JÔNATAS PULZ

Abordagem alternativa para cálculo regulatório de perdas não-técnicas do sistema de distribuição e técnicas de *machine learning* para detecção de fraude na baixa tensão

São Paulo

2022

JÔNATAS PULZ

Abordagem alternativa para cálculo regulatório de perdas não-técnicas do sistema de distribuição e técnicas de *machine learning* para detecção de fraude na baixa tensão

Versão Corrigida

Dissertação apresentada ao Programa de Pós-Graduação de Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para obtenção do título de Mestre em Ciências

Área de Concentração: Sistemas de Potência

Orientador: Prof. Dr. Carlos Frederico Meschini Almeida

São Paulo

2022

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 12 de Setembro de 2022

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Pulz, Jonatas
Abordagem alternativa para cálculo regulatório de perdas não-técnicas do sistema de distribuição e técnicas de machine learning para detecção de fraude na baixa tensão / J. Pulz -- versão corr. -- São Paulo, 2022.
254 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.Aprendizado de máquina 2.Enérgia elétrica 3.Furto I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

“O coração do homem pode fazer planos, mas a resposta certa dos lábios vem do Senhor”

Provérbios 16:1 – Bíblia Sagrada

DEDICATÓRIA

Dedico a presente dissertação a Deus, o único Deus, criador dos céus e da terra e escritor da Bíblia Sagrada, que é e sempre será a fonte de toda sabedoria. Agradeço a Ele pela sua misericórdia, graça e sabedoria que me foi dada.

Como a quem muito é dado, muito é pedido, eu não poderia deixar de concluir este trabalho com a maior dedicação possível, pois toda glória por este trabalho é de Deus. DEle é o Reino, o poder e a glória para sempre, amém.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, pois sem Ele esse trabalho jamais seria possível, Ele é a fonte de toda sabedoria.

Agradeço ao meio orientador, Carlos Frederico, por toda a ajuda e paciência na jornada até a finalização deste trabalho, que não foi simples ou fácil.

Agradeço a minha família e, em especial, a minha irmã Michele, que sempre me incentivou a não desistir e a finalizar este trabalho.

Agradeço a todos da Daimon, principalmente ao Dr. André Meffe, Dr. Alden Antunes e Dr. Carlos Barioni; e em especial ao Renan Muller e ao Denis Antonelli, sendo que estes me ajudaram muito com seus conhecimentos, e sem a ajuda deles eu não poderia ter concluído este trabalho.

RESUMO

Pulz, Jonatas. **Abordagem alternativa para cálculo regulatório de perdas não-técnicas do sistema de distribuição e técnicas de *machine learning* para detecção de fraude na baixa tensão.** 2022. 254 f. Dissertação (Mestrado em Ciências e Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2022.

As perdas não técnicas são um problema significativo em países subdesenvolvidos, decorrente, principalmente, de fraudes em medidores e furtos de energia. Para mitigá-las, as distribuidoras realizam inspeções em unidades consumidoras suspeitas. O custo operacional para se realizar essas inspeções é alto e só pode ser justificado por um retorno através da descoberta de fraudes. Para aumentar a precisão na descoberta de fraudes, modelos de *machine learning* podem ser utilizados. Este trabalho propõe modelos de detecção de fraudes utilizando os tipos de modelos mais atuais e que vem se destacando como bons classificadores. Além disso, este trabalho propõe uma metodologia de cálculo regulatório de perdas mais realista que leve em consideração esse rico banco de dados criado através das inspeções realizadas pelas distribuidoras e o compara com a metodologia regulatória atual numa área piloto da distribuidora Enel de São Paulo.

Palavras-chave: *Analytics*; Detecção de fraude; Furto de energia elétrica, *Machine Learning*; Perdas de energia; Perdas não-técnicas; Perdas regulatórias; Perdas técnicas.

ABSTRACT

Pulz, Jonatas. **Alternative approach for regulatory non-technical losses calculation of distribution system and machine learning techniques for fraud detection at low voltage.** 2022. 254 p. Dissertation (Master in Science and Electrical Engineering) – Polytechnic School, University of Sao Paulo, Sao Paulo, 2022.

Non-technical losses are a significative problem in developing countries, mainly due to tampering in meters and energy theft. To mitigate them, utilities carry out inspections in suspected consumers. The operational cost to carry out these inspections is high and can only be justified by a return through fraud discovery. To increase the precision of fraud detection, machine learning models can be used. This work presents fraud detection models using the most recent types of models that have been showing good results as classifiers. In addition, this work proposes a more realistic regulatory loss calculation methodology that takes into account this rich database created through inspections carried out by utilities and compares it with the current regulatory methodology in a pilot area of the utility Enel of São Paulo.

Keywords: Analytics; Electricity theft; Energy Losses; Fraud detection, Machine Learning; Non-technical Losses; Regulatory Losses; Technical Losses.

LISTA DE ILUSTRAÇÕES

Figura 1 - Divisão territorial por setores censitários de uma região de São Paulo 46
Figura 2 - Conceituação de um modelo com saída binária 47
Figura 3 - Ilustração de uma validação cruzada.....49
Figura 4 - Exemplo de árvore de decisão.....52
Figura 5 - Exemplo de Random Forest com 4 árvores de decisão.....53
Figura 6 - Predição de um modelo usando *boosting* 55
Figura 7 - Área piloto escolhida (em rosa) 68
Figura 8 - Inspeções (pontos verdes) realizadas na área piloto em 2016 69
Figura 9 - Login no SQL Server.....70
Figura 10 - Criação do banco "InspectionEnel2016" 70
Figura 11 - Criação do banco “BDGDEnel2019” no SQL Server.....72
Figura 12 - Instalação do GDAL 74
Figura 13 - Retorno do comando para criação da BDGD no SQL Server 74
Figura 14 - Histograma do número de fraudes por retângulo.....76
Figura 15 - Histograma do número de fraudes por retângulo sem valor extremo 76
Figura 16 - Histograma em duas dimensões do número de fraudes.....77
Figura 17 - *Kriging* normalizado do histograma de fraudes.....78
Figura 18 - Comparação entre número de fraudes e *Kriging* 79
Figura 19 - IPF sobreposto ao mapa da região de interesse (linha preta delimita a área piloto) 80
Figura 20 - Segmentos de média tensão (linha vermelha) na área piloto 83
Figura 21 - Janela inicial do ProgGeoPerdas 1.0.0.10 84
Figura 22 - Configuração do ProgGeoPerdas com relação ao banco de dados 88
Figura 23 - Configuração do ProgGeoPerdas para cálculo de perdas nos alimentadores.....89
Figura 24 - Configuração de Banco de Dados do novo ProgGeoPerdas 94
Figura 25 - Relação entre IQR do IPF do alimentador e variação relativa absoluta de perdas totais entre as metodologias.....107
Figura 26 - Abertura do ODBC Data Sources 169
Figura 27 - Criação da conexão SQL Server.....170
Figura 28 - Configuração da conexão ODBC com SQL Server.....171
Figura 29 - Criação da conexão no QGIS, primeiro passo.....172

Figura 30 - Criação da conexão no QGIS, segundo passo 173
Figura 31 - Inclusão da área piloto no QGIS 174
Figura 32 - Inclusão dos segmentos de média tensão no QGIS 175
Figura 33 - Processing Toolbox no QGIS.....176
Figura 34 - Janela para cálculo da interseção entre ssmt e área piloto 177
Figura 35 - Exportação da intersecção entre ssdmt e área piloto 178
Figura 36 - Janela para cálculo da diferença entre ssmt e área piloto 179
Figura 37 - Como criar o *schema* “sde” através do SSMS 209
Figura 38 - Criação do *schema* “sde” no SSMS 210
Figura 39 - Menu para criar login 227
Figura 40 - Como criar login.....228
Figura 41 - Configuração necessária para criar login.....229

LISTA DE TABELAS

Tabela 1 - Variáveis utilizadas pela ANEEL 37
Tabela 2 - Base de Perdas 2019.....40
Tabela 3 - Matriz de confusão 48
Tabela 4 - Resultados da validação cruzada (sem super ou subamostragem) 59
Tabela 5 - Resultados da validação cruzada com subamostragem 60
Tabela 6 – Resultados do teste do melhor modelo 61
Tabela 7 - Contagem de curvas de carga alocadas a unidades consumidoras de baixa tensão 85
Tabela 8 - Resultados inconsistentes de perdas com dados da BDGD (amostra de 8 alimentadores).....90
Tabela 9 - Relação de energia entre cargas e declaradas para os alimentadores com dados da BDGD (amostra de 8 alimentadores).....91
Tabela 10 - Variação de energia entre dados de energia da BDGD e dados de energia da 5ª RTP (amostra de 8 alimentadores) 92
Tabela 11 - Relação de energia entre cargas e declaradas para os alimentadores com dados da 5ª RTP (amostra de 8 alimentadores).....93
Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia.....98

LISTA DE SIGLAS E ABREVIATURAS

ANEEL	Agência Nacional de Energia Elétrica
AMI	*Advanced Metering Infrastructure*
ASRO	Áreas com Severas Restrições à Operação
BDGD	Base de Dados Geográfica da Distribuidora
BT	Baixa Tensão
EI	Energia Injetada
IQR	*Inter Quartile Range*
IPF	Índice Potencial de Fraude
LTR	*Long Term Release*
PD	Perdas na Distribuição
PNT	Perdas Não-Técnicas
PPD	Percentual de perdas na distribuição
PPNT	Percentual de perdas não técnicas
PPT	Percentual de perdas técnicas
PT	Perdas Técnicas
RTP	Revisão Tarifária Periódica
SVM	*Support Vector Machines*
SSMS	SQL Server Management Studio
XGBoost	*Extreme Gradient Boosting*

SUMÁRIO

1. INTRODUÇÃO 21
1.1. Objetivo 24
1.2. Organização do Trabalho 24
2. REVISÃO BIBLIOGRÁFICA 27
3. CONTEXTUALIZAÇÃO DAS PERDAS 33
3.1. Cálculo de Perdas Técnicas e Não Técnicas Regulatórias 34
3.1.1. Parâmetros adotados no cálculo 35
3.2. Metas de Perdas Não Técnicas 35
3.2.1. Detalhamento do estabelecimento das metas de PNT 36
3.3. Potencial de ganho com mitigação ou revisão de cálculo das PNT 39
4. MITIGAÇÃO DAS PNT: INSPEÇÕES E *MACHINE LEARNING* 43
4.1. Dados utilizados 43
4.1.1. Obtenção e cruzamento com dados do IBGE 45
4.2. Fundamentos de machine learning 46
4.3. Tipos de modelos utilizados 50
4.3.1. Regressão Logística 50
4.3.2. *Random Forest* 51
4.3.3. *Extra Trees* 54
4.3.4. XGBoost 54
4.4. Variáveis utilizadas 56
4.5. Treinamento dos modelos e resultados 58
5. NOVA METODOLOGIA DE CÁLCULO PARA PNT 63
5.1. Detalhamento da metodologia atual de cálculo de PNT e PT 64
5.2. Detalhamento da metodologia proposta para cálculo de PNT e PT 66
5.2.1. Cálculo do Índice Potencial de Fraude 66

5.3. Implementação da metodologia proposta e resultados 67

5.3.1. Resultados de perdas com a metodologia regulatória e com a metodologia proposta.....80

6. CONCLUSÕES.....109

7. PUBLICAÇÕES REALIZADAS.....111

8. FUTUROS TRABALHOS 113

REFERÊNCIAS BIBLIOGRÁFICAS.....115

Apêndice A - *Notebook* Python para treinamento dos modelos de *machine learning* 119

Apêndice B - Módulo Python adicional para treinamento dos modelos de *machine learning*.....141

Apêndice C - Área piloto do estudo em formato GeoJSON.....155

Apêndice D - *Notebook* Python para obtenção do IPF para cada consumidor da BDGD.....161

Apêndice E - Passo-a-passo no QGIS para obtenção dos alimentadores da área piloto 169

Apêndice F - Alocação das curvas de carga.....181

Apêndice G - Passo-a-passo para obtenção do banco para cálculo de perdas através da metodologia atual regulatória a partir da BDGD 203

Apêndice H - Passo-a-passo para obtenção do banco para cálculo de perdas através da metodologia atual regulatória a partir dos dados da 5ª Revisão Tarifária Periódica 231

Apêndice I - Passo-a-passo para obtenção do banco para cálculo de perdas através da metodologia proposta 237

Apêndice J - Notebook Python para obtenção da análise entre IQR e diferenças para perdas entre metodologias 247

1. INTRODUÇÃO

As perdas de energia elétrica na distribuição não se referem apenas às perdas provenientes de efeitos físicos, como, por exemplo, as do efeito Joule. As perdas de energia elétrica na distribuição se referem a toda a energia mensurada como entrada no sistema da distribuidora subtraída de toda a energia mensurada como saída (consumo mensurado dos clientes de qualquer nível de tensão, o que pode incluir também outras distribuidoras) (ANEEL, 2015).

De acordo com dados fornecidos (ANEEL, 2021b) pela Agência Nacional de Energia Elétrica (ANEEL), o órgão responsável por regular o setor elétrico brasileiro, as perdas na distribuição reconhecidas pela agência em 2019 representam 12,41% (65,090,295 MWh) de toda a energia injetada das 53 principais distribuidoras. Considerando um preço médio de aquisição da distribuidora de 200 R$/MWh esse montante de energia perdida representou aproximadamente 13 bilhões de reais em 2019.

Como mencionado anteriormente, as perdas são a diferença entre toda a energia medida como entrada subtraída de toda a energia medida como saída, essa são as Perdas na Distribuição (PD). As perdas na distribuição são divididas pela ANEEL em duas partes:

- Perdas Técnicas (PT): perdas provenientes de efeitos físicos;
- Perdas Não-Técnicas (PNT): Perdas na Distribuição subtraídas das Perdas Técnicas.

Enquanto as PT provêm de efeitos físicos, como o efeito Joule, as PNT provêm de furtos de energia, fraudes nos medidores ou falhas da própria distribuidora no processo de leitura e faturamento do correspondente consumo de seus clientes (ANEEL, 2015), e essas PNT são um problema muito acentuado em países em desenvolvimento como o Brasil, podendo chegar a 40% da energia total distribuída em países como Brasil, Índia, Malásia e Líbano (GLAUNER, MEIRA, VALTCHEV, STATE, & BETTINGER, 2017). Para obter os ganhos de produtividade e eficiência no que diz respeito às PNT da distribuidora, ela deve melhorar seus processos gerenciais para mitigar os erros de leitura e faturamento, além de inspecionar

unidades consumidoras suspeitas de possuírem adulteração em seu processo de medição de consumo.

A ANEEL define metas para redução das PNT de cada distribuidora que levam em consideração a dificuldade socioeconômica da área de atuação da distribuidora (mais detalhes em 3.2), mas caso essa não alcance a meta, perde financeiramente.

As distribuidoras fazem inspeções nas unidades consumidoras para encontrar as adulterações nos medidores e o furto de energia, usando modelos para detecção de fraude ou a experiência dos inspetores, ou ambos, com o objetivo de mitigar as PNT. Tal ação de inspecionar é custosa pela mão de obra envolvida, portanto é desejável que cada uma das inspeções seja direcionada para unidades consumidoras com alta probabilidade de possuir fraude, para obter o retorno desse custo com a diminuição da PNT proveniente da fraude. Isso é possível ao se utilizar técnicas de *machine learning*, em que se utilizam modelos para detecção de fraude nas unidades consumidoras do sistema de distribuição de energia elétrica. Todas essas inspeções geram um grande e rico conjunto de dados que pode ser usado para aprimorar os modelos de *machine learning*. Portanto, além dos ganhos para a distribuidora, o uso de modelos de *machine learning* pode contribuir para a sociedade, evitando o desperdício de energia, já que uma fraude ou um roubo de energia elimina, parcialmente ou totalmente, o sinal econômico que pode frear o consumo inconsciente.

As PD, tanto PT quanto PTN, são inclusas na tarifa de energia paga por cada unidade consumidora. A ANEEL, ao calcular as perdas de energia da distribuidora que compõe a tarifa, considera um algoritmo de fluxo de potência que distribui as PNT entre todos os consumidores, proporcionalmente ao seu consumo (ANEEL, 2018), como se existissem cargas fictícias acopladas a cada consumidor que representam as PNT e essas perdas estivessem distribuídas por todos os consumidores, sem exceção, proporcionalmente aos seus consumos. Essa abordagem não é realista, uma vez que a situação cultural, social e econômica de um país influencia as PNT, também de forma geográfica (YURTSEVEN, 2015), com enfoque para as PNT geradas por fraudes na medição e furtos de energia.

Uma abordagem regulatória mais realista para o cálculo de perdas pode beneficiar a concessionária em termos de planejamento, manutenção, uma vez que o resultado da simulação do fluxo de potência mostra uma visão mais realista das partes sobrecarregadas da rede, e das PT e PNT em cada área geográfica. Ou seja, como a alocação das PNT para cada consumidor é uma etapa prévia da simulação do fluxo de potência, alocando as PNT para grandes consumidores - que, em geral, ficam em áreas ricas - proporcionalmente ao seu consumo podem resultar em superestimação das PT simuladas em sua área; enquanto a alocação de PNT para áreas pobres proporcionalmente ao consumo de cada consumidor podem resultar em perdas técnicas simuladas subestimadas. Em (YURTSEVEN, 2015), o autor mostra uma proporcionalidade negativa por regiões entre o consumo ilegal de energia elétrica e o aumento da renda nos países em desenvolvimento.

Diante do exposto, este trabalho apresenta o uso de *machine learning* para criação de modelos para indicação potencial de fraude em unidades consumidoras de baixa tensão do sistema de distribuição; e uma nova metodologia para o cálculo regulatório de PNT que inclui o uso do banco de dados de inspeções da distribuidora e que aloca PNT nos locais ondem foram encontradas mais fraudes.

Como benefícios potenciais deste trabalho, pode-se enumerar:

- Uso dos modelos de indicação potencial de fraude para aumentar a acurácia da detecção de fraudes por parte das equipes de inspeção. Além de diminuir o custo operacional da distribuidora com equipes de inspeção, a mitigação das PNT pode gerar um ganho para a distribuidora devido a trajetória de perdas definida pela ANEEL;
- Uma metodologia revisada de cálculo regulatório das PNT, que tenda a utilizar dados mais realistas, pode trazer ganho para a sociedade como um todo. Isso ocorre porque, se os valores de PD forem maiores, significa que falta remuneração para a distribuidora, e se os valores de PD forem menores, significa que os consumidores pagam acima do que deveriam. Se essa revisão for, por exemplo, 1% do valor mencionado de PD em 2019, significa que 130 milhões de reais seriam realocados.

1.1. Objetivo

Como primeiro objetivo deste trabalho, ele tem por apresentar modelos que indiquem potenciais fraudes em unidades consumidoras de baixa tensão no sistema de distribuição para aumentar a acurácia das equipes de inspeção da distribuidora, obtidos através de técnicas de *machine learning*. Isso pode gerar a diminuição de custo operacional e mitigação das PNT, ambos resultando em ganhos para a distribuidora e para os consumidores, pois as perdas fazem parte da tarifa de energia.

Como segundo objetivo deste trabalho, ele tem por apresentar uma nova metodologia de cálculo das PNT, por conseguinte das PD, que utilize uma abordagem mais realista com relação a atual. Essa nova metodologia utiliza o banco de dados de inspeções das unidades consumidoras da distribuidora para concentrar PNT nas áreas onde foram encontradas mais fraudes.

O trabalho também detalha toda a forma para se obter os modelos de detecção de fraude e também toda a forma para obtenção do cálculo regulatório de PNT, permitindo, portanto, que qualquer pessoa interessada possa reproduzir os resultados obtidos neste trabalho.

1.2. Organização do Trabalho

Este trabalho é dividido nos seguintes capítulos:

- O Capítulo 2 apresenta os trabalhos atuais relacionados aos objetivos deste;
- O Capítulo 3 apresenta a contextualização das perdas, seus valores estimados atualmente e ganhos potenciais com este trabalho;
- O Capítulo 4 apresenta os modelos de indicação potencial de fraude, com as bases teóricas necessárias, suas construções, utilizações e resultados;
- O Capítulo 5 apresenta a nova metodologia para cálculo das PNT, e por conseguinte das PT, e um comparativo com a metodologia atual;
- O Capítulo 6 apresenta as conclusões deste trabalho;

- O Capítulo 7 apresenta os trabalhos publicados;
- O Capítulo 8 apresenta os possíveis desdobramentos futuros deste trabalho.

2. REVISÃO BIBLIOGRÁFICA

Nesta seção são apresentados os estudos de outros trabalhos correlatos com este, seja no tema de perdas não técnicas ou de detecção de fraude nos sistemas de distribuição.

Primeiramente, são revisados trabalhos que relacionam aspectos socioeconômicos com as perdas não técnicas.

Em (YURTSEVEN, 2015), é apresentada a relação entre furto de energia e indicadores socioeconômicos na Turquia através de análises econométricas e certas ações, baseando-se nos achados, são indicadas. No trabalho, o objetivo é tentar entender os fatores subjacentes ao furto de energia para indicar ações que previnam o furto antes que ele ocorra. Nele é mostrado que educação, renda, capital social (a vontade de disponibilizar recursos a outros sem qualquer compensação explícita), razão de população rural, índice de temperatura e produção agrícola são importantes determinantes para o furto de energia. O trabalho mostra que há uma relação negativa entre a renda e educação com relação ao furto de energia. Além disso, o trabalho também mostra uma relação positiva entre o preço da energia e o furto de energia e uma relação negativa para o capital social (mostrando que as pessoas evitam ter ações que não são socialmente aceitas). O autor sugere, para a prevenção de fraude, a criação de uma tarifa social, devido a relação entre renda, preço da energia com relação ao furto de energia; sugere também o investimento em educação que inclua o tema do problema de furto de energia nos currículos do ensino.

Em (SAINI, 2017), é realizada uma revisão bibliográfica sobre a relação entre aspectos sociais/comportamentais e furto de energia. Um dos estudos revisados mostra a alta taxa de furto de energia na Jamaica, se comparado com outros países, e que a mesma causa incremento no preço da energia e baixa qualidade no serviço prestado. Outro estudo revisado, referente ao Paquistão onde a maioria da população é pobre, mostra que o preço da energia influencia diretamente o furto de energia. Outro estudo revisado mostra que existe relação entre furto de energia e alto nível de corrupção nos países analisados, que o furto de energia aumenta o preço da energia e diminui a receita na venda de energia e que, para mitigar o furto

de energia deveriam ser utilizados medidores à prova de adulteração, deveriam ser realizadas inspeções e monitoramento e, em alguns casos, deveria haver uma mudança regulatória. Outro estudo revisado discute o furto de energia em termos de aspectos sociais, econômicos, regionais, políticos, educacionais, criminais e de corrupção na Índia e sugere que a forma de punição para os consumidores fraudadores precisa ser revista. Outro estudo revisado investiga, na Índia, a ineficiência de métodos tecnológicos e a eficiência de métodos psicossociais para a redução do furto de energia. Outro estudo revisado mostra que o furto de energia é feito, deliberadamente, por estratégias políticas no estado de Uttar Pradesh da Índia, notando o aumento de furto de energia em períodos anteriores às eleições. Outro estudo revisado, de 2016, é o primeiro a ser conduzido utilizando dados de toda a Índia e mostra o impacto negativo de bons indicadores socioeconômicos no furto de energia. Outro estudo revisado, sobre a Uganda, mostra os benefícios da implementação de um sistema pré-pago de energia na redução do furto de energia, que foi implementado com sucesso em Ruanda. Outro estudo revisado, com relação às favelas do Brasil, mostra que o uso ilegal de energia está relacionado com a baixa renda, uso incorreto de eletrodomésticos, serviço de energia abaixo do padrão mínimo e uso de energia para propósitos comerciais em residenciais e que o uso ilegal de energia pode ser mitigado com a conscientização, melhoria em equipamento, promoção de medidas de uso consciente da energia, além de medição e manutenção com qualidade. Outro estudo revisado mostra a relação entre a falta de eletricidade e o furto de energia como um dos responsáveis. Outros estudos foram revisados, mas os principais que contém relação com este trabalho foram mencionados.

Em (HUBACK, 2018), são propostas medidas para reduzir as perdas não técnicas em Áreas com Severas Restrições à Operação (ASRO), ou seja, áreas controladas por criminosos que limitam a operação e a supervisão da distribuidora. A autora utiliza a experiência de distribuidoras de outros países (Colômbia, Peru, Jamaica, Filipinas) para propor medidas. No trabalho, é também analisado o caso específico da Light, que possui grande parte de sua área classificada como ASRO e que atua no combate às perdas não técnicas através de ações convencionais de inspeção, instalação de redes blindadas e sistema de medição centralizada, mas essas ações são de difícil implementação em ASRO. A autora sugere como medidas propositivas

para as distribuidoras brasileiras com relação a suas perdas não técnicas, com base nas experiências de outros países: tarifa social (tarifa mais baixa para pessoas com baixa renda); medidas de aproximação e melhora da imagem da distribuidora, o que pode permitir a operação dos funcionários nas comunidades; medidas tecnológicas, como redes blindadas, medidores eletrônicos, sistema de medição centralizada, mas difíceis de serem feitas em ASRO; reconhecimento regulatório diferenciado para as ASRO, pois o modelo atual para definição do nível de perdas das distribuidoras não leva isso em consideração, e é importante para reduzir as perdas econômicas das distribuidoras; sistemas de pré-pagamento; medição nos transformadores, para permitir a localização mais precisa das perdas não técnicas; convênio com a procuradoria; denúncia de eletrotraficantes em veículos de imprensa.

Os estudos revisados acima mostram que questões socioeconômicas têm relação com o furto de energia elétrica, portanto justifica-se a utilização de dados socioeconômicos em um modelo para detecção de fraude. Os próximos estudos revisados concernem a modelos para a detecção de fraude nas unidades consumidoras.

Em (VIEGAS, ESTEVES, MELÍCIO, MENDES, & VIEIRA, 2017), é realizada uma extensa revisão da literatura com relação a possíveis soluções para a detecção de perdas não técnicas. As soluções associadas às perdas não técnicas são divididas em soluções teóricas (análises de relação de perdas não técnicas com outros fatores de forma agregada), soluções baseadas em *hardware* (instalação de equipamentos específicos) e soluções não baseadas em *hardware* (técnicas de análises de dados). Nele, foram analisados 103 estudos, de 2000 a 2016. Os autores mostram que o interesse pelo assunto de perdas não técnicas se tornou mais acentuado a partir de 2006. Dentre as soluções não baseadas em *hardware*, essas são divididas, com relação a suas técnicas, entre classificação, estimação, teoria dos jogos e outras abordagens. Os autores explicam que a principal vantagem das soluções não baseadas em *hardware*, que é o foco deste trabalho, é o seu baixo custo de investimento se comparado com soluções baseadas em *hardware*. As técnicas mais utilizadas por essas soluções eram (até o ano de 2016): *Support Vector Machines* (SVM), *Artificial Neural Networks* e *Decision Trees* (árvores de decisão) como classificadores para indicação de fraude que utilizam dados dos

consumidores para treinamento do classificador. Os autores também mostram que vários trabalhos utilizam a caracterização da curva de carga do consumidor, mas para isso é necessário possuir *smart meters* na rede que colham essa informação com granularidade melhor ou igual a 1 hora, que não é o caso deste trabalho que utiliza o consumo faturado mensalmente.

Em (HU, et al., 2019), propõe-se um modelo de *deep learning* (redes neurais) com dados de uma *Advanced Metering Infrastructure* (AMI) – que seria uma estrutura de rede de distribuidora com *smart meters* –, para detectar fraudes. Esse artigo apresenta relevância, pois foi citado por 31 outros artigos. Os autores explicam que os modelos convencionais de SVM ou árvore de decisão não conseguem lidar com a alta dimensionalidade de variáveis que provém de uma AMI, por exemplo, no caso de *smart meters* que obtenham dados numa resolução de 15 minutos existem 672 variáveis para caracterizar uma unidade consumidora semanalmente, e esse número é muito alto para um modelo convencional de SVM ou árvore de decisão e *deep learning* (redes neurais). Além disso, os autores utilizam uma técnica de aprendizado de máquina semi-supervisionado para utilizar tanto dados classificados (com a resposta de fraude encontrada ou não) quanto dados não classificados (sem a resposta de fraude encontrada ou não), isso é interessante pois não é possível inspecionar todos os consumidores e não levar em consideração esses dados seria desperdício, segundo os autores.

Em (YAN & WEN, 2021), utiliza o modelo *Extreme Gradient Boosting* (XGBoost) com dados de uma AMI para criar um modelo de detecção de fraude. Os autores explicam que uma AMI, por causa dos *smart meters*, está exposta a novos tipos de métodos de furto de energia através de ataque por *hackers*. Os autores separam os métodos para combate de furto de energia em 3 categorias: abordagens baseadas em estados, em que é necessário instrumentos de medição ao longo da rede, porém a estimação de estados só pode ser usada no nível da uma subestação e não no nível do usuário final; modelos baseados em teoria dos jogos, em que é modelado um jogo entre fraudadores e a distribuidora, mas é difícil de se encontrar uma equação adequada; e por fim, modelo baseados em inteligência artificial, em que é utilizado *machine learning* para que o modelo aprenda o perfil de um fraudador e de um não fraudador. Os autores explicam que rede neurais (*deep learning*) estão

propensas a *overfitting*, que ocorre quando o modelo prevê muito bem os dados utilizados para treinamento, mas não consegue prever bem dados que não foram utilizados para treinamento; eles explicam também que modelos como SVM mostram baixa performance quando os dados possuem ruído e por isso propõem o uso do XGBoost, que é um modelo mais recente baseado em árvores de decisão e no método do gradiente. Os autores mostram que o XGBoost apresenta melhor desempenho que todos os outros modelos, como, por exemplo, SVM, redes neurais e árvores de decisão.

Em (MASSAFERRO, MARTINO, & FERNÁNDEZ, 2020), é proposto um modelo que maximize o retorno econômico na detecção de perdas não técnicas, em vez de um modelo com as melhores métricas de acurácia ou outras métricas padrões de modelos de classificação. O estudo é realizado com dados do Uruguai e propõe um modelo de *machine learning* baseado em dados de consumo e nos custos para a distribuidora com inspeções para produzir uma lista ótima de consumidores a serem inspecionados, indicando consumidores potencialmente fraudadores de tal forma que o retorno econômico para a distribuidora seja o maior possível. O estudo utiliza *Random Forest*, SVM e redes neurais como modelos base para alcançar seus objetivos. O estudo utiliza dados mensais de consumo em um histórico de 3 anos para entrada para os modelos.

Os estudos revisados acima mostram que, até 2016, os principais modelos utilizados para detecção de fraude eram SVM e árvores de decisão, e que avanços foram feitos com a utilização de redes neurais de uma forma semi-supervisionada e por fim o XGBoost tem se mostrado um dos melhores modelos para utilização devido a sua performance superior com relação a outros tipos de modelos. Por fim, neste capítulo, são revisados estudos que proponham uma metodologia para estimativa das perdas não técnicas.

Em (DONADEL, et al., 2009), é proposta uma metodologia para aprimorar o cálculo as perdas técnicas através de estimativas das perdas não técnicas, sendo que essa estimativa leva em consideração o histórico de consumo e os resultados das inspeções. Os autores estimam o número mínimo de inspeções, por região, para representar com rigor estatístico a proporção de fraudes da região, se o número de inspeções estiver abaixo do mínimo, eles então agrupam regiões até que se obtenha

o valor mínimo de inspeções necessárias, formando um *cluster*. Com o número mínimo de inspeções por *cluster*, é calculada a proporção de consumidores irregulares daquele cluster, e, portanto, o número de consumidores irregulares será essa proporção multiplicada pelo número de consumidores do *cluster*. Dado o número F_j estimado de consumidores irregulares para o *cluster* j, são selecionados os F_j consumidores com consumo médio mais próximo da média do *cluster* para terem seus consumos ajustados. O ajuste no consumo dos consumidores selecionados, que representará as perdas não técnicas daquele *cluster*, é feito analisando-se o consumo médio antes e após a inspeção com irregularidade encontrada para cada *cluster*.

Em (ROSSONI, et al., 2015), é utilizado detecção de anomalia para se determinar, com relação ao faturamento de energia, os consumidores que são potencialmente fraudadores. Detecção de anomalia objetiva encontrar padrões em dados que não estão em conforme com o esperado. O resultado da detecção da anomalia é uma razão de consumidores suspeitos em cada transformador. Essa razão é utilizada como parte do cálculo do peso na estimação de estados através de uma técnica chamada *Weighted Least Squares State Estimation*. Portanto, esse estudo não utiliza dados de inspeções, mas utilizada a técnica de detecção de anomalia em conjunto com estimação de estados para determinar as perdas não técnicas e técnicas em cada transformador.

Em (HENRIQUES, CORRÊA, FORTES, BORBA, & FERREIRA, 2020), mostra-se que é possível utilizar sensores de temperatura instalados ao longo da rede de distribuição em conjunto com equações de dissipação de calor para estimar as perdas técnicas nos cabos e conexões da rede. Esses sensores são integrados a uma *smart grid* que lê os dados dos sensores e estima as perdas técnicas. Estimando as perdas técnicas e utilizando uma equação de balanço de energia é possível chegar a uma estimativa de perdas não técnicas ao longo da rede, possibilitando uma forma mais acurada de se detectar pontos onde deve haver fraudes ou furtos de energia. Essa solução proposta utiliza, portanto, *hardwares* adicionais.

3. CONTEXTUALIZAÇÃO DAS PERDAS

Neste capítulo, é apresentado o contexto das perdas de energia elétrica na distribuição no Brasil.

Como definido pela ANEEL em (ANEEL, 2015):

- Perdas na Distribuição (PD) são a diferença entre a energia injetada na rede da distribuidora e o total de energia vendida e entregue;
- Perdas Técnicas (PT) compõem a parcela da PD inerente ao processo de transporte, de transformação de tensão e de medição de energia. Ou seja, as PT são compostas pelas perdas oriundas dos efeitos físicos: perdas Joule nos condutores; perdas no núcleo, por histerese e por correntes de Foucault nos transformadores; e qualquer outra perda por efeito físico que ocorra durante o percurso da energia desde a entrada na distribuidora até o consumidor;
- Perdas Não Técnicas (PNT) compõem a parcela da PD que não seja de origem física, ou seja, todas as demais perdas associadas à distribuição de energia elétrica, tais como furtos de energia, erros de medição, erros no processo de faturamento, unidades consumidoras sem equipamento de medição etc. É calculada como a diferença entre PD e PT.

Convém detalhar outras definições e conceitos que serão utilizados neste trabalho e estão presentes em (ANEEL, 2015):

- Energia Injetada (EI): somatório de toda energia que entra na rede da distribuidora, seja a energia que entra pela fronteira (por exemplo a que vem da transmissão) ou que é gerada localmente;
- Percentual de perdas na distribuição (PPD): percentual das PD em relação à EI;
- Percentual de perdas técnicas (PPT): percentual das PT em relação à EI;
- Percentual de perdas não técnicas (PPNT): percentual das PNT em relação à EI.

As PT e PNT que compõem parte da tarifa do consumidor final são chamadas de regulatórias, e no caso das PNT, existem metas estabelecidas pela ANEEL. As PT, PNT e as metas dessas são calculadas por métodos definidos pela ANEEL com certas especificidades, as quais são dissertadas nas próximas seções. Por fim, há uma seção para a discussão de potenciais ganhos com a mitigação das PNT por parte da distribuidora ou uma metodologia regulatória revisada para cálculo das PNT.

3.1. Cálculo de Perdas Técnicas e Não Técnicas Regulatórias

A metodologia de cálculo das perdas na distribuição é definida em (ANEEL, 2018) e detalhada nesta seção.

Para o cálculo das perdas, são necessários os dados físicos e de energia da rede de distribuição, e eles são enviados pela distribuidora a ANEEL, de forma recorrente, no formato definido pela ANEEL de um banco de dados chamado Base de Dados Geográfica da Distribuidora (BDGD). Na BDGD, se encontram todas as informações referentes a cabos, rede, transformadores, reguladores, chaves e medidores, além de informações de medição de energia em alimentadores e consumidores.

Na BDGD, as informações de medição são mensais, mesmo no nível mais granular que é o dos consumidores. Portanto, é necessário o estabelecimento de curvas de carga para os consumidores, e sua forma de obtenção é definida pela ANEEL em (ANEEL, 2014b), sendo que tal detalhamento foge ao escopo desta seção. É suficiente saber que existe uma atribuição de curva de carga através de um código para cada consumidor da distribuidora na BDGD e seu consumo é dado de forma mensal também na BDGD.

Na metodologia atual utilizada pela ANEEL, é atribuída uma carga virtual que representa a PNT para cada uma das unidades consumidoras de forma proporcional ao consumo de cada uma dessas unidades. Depois é utilizado um método iterativo que aumenta ou diminui essa carga virtual até que a soma dos consumos reais, das cargas virtuais de PNT e as PT calculadas pelo fluxo de potência através do software OpenDSS (EPRI, 2022) esteja próximo do valor declarado de energia

consumida no alimentador que supre as unidades consumidoras. Nota-se que a forma de atribuir as PNT influencia diretamente as PT, já que esta é dependente da primeira. Portanto, as PD são influenciadas diretamente pela metodologia adotada para atribuição de PNT.

3.1.1. Parâmetros adotados no cálculo

- Para as cargas de média de baixa tensão, é adotado o fator de potência de 0,92.
- Elementos de compensação de reativos são desprezados quando instalados na rede de média ou baixa tensão.

3.2. Metas de Perdas Não Técnicas

A metodologia para obtenção das metas de PNT Regulatória é detalhada em (ANEEL, 2015) e nesta seção é dissertado sobre ela.

Idealmente, todas as distribuidoras, deveriam ter PNT nulas, porém, como afirmado em (ANEEL, 2013), é consensual a ideia de que as características do meio em que a concessionária está inserida podem influenciar dramaticamente o seu resultado no combate às perdas não técnicas. (ANEEL, 2013) também afirma que a experiência tem demonstrado que combater perdas em algumas áreas é bem mais desafiador que em outras, e que isso está associado a questões socioeconômicas da região como: presença de criminalidade; acentuada desigualdade social; precariedade de infraestrutura, ausência de presença do Estado, dentre outras.

Diante do exposto, o estabelecimento das metas de PNT pela ANEEL para cada distribuidora leva em consideração as características socioeconômicas da região que a distribuidora atende. Através de uma metodologia que é descrita a seguir, a ANEEL compara distribuidoras que atendem regiões com características socioeconômicas semelhantes e se uma delas possui valores de PNT menores que as outras, então ela se torna um *benchmark* – referência – para as outras com valores de PNT maiores, estabelecendo-se assim uma meta.

3.2.1. Detalhamento do estabelecimento das metas de PNT

As metas de PNT são estabelecidas em termos percentuais com relação ao mercado de baixa tensão da distribuidora conforme a equação:

$$P_{nt} = \frac{PD - EI \cdot \frac{PPT}{100}}{M_{bt}} \cdot 100 \qquad (3.1)$$

Onde:

- P_{nt} é o percentual de perdas não técnicas sobre o mercado de baixa tensão;
- PD é a Perdas na Distribuição, já definida;
- EI é a Energia Injetada, já definida;
- PPT é o Percentual de Perdas Técnicas, obtido como descrito na seção 3.1;
- M_{bt} é o mercado de baixa tensão da distribuidora.

Para a comparação entre distribuidoras, a ANEEL criou o chamado índice de complexidade socioeconômica. (ANEEL, 2014a) afirma que a principal função do índice é viabilizar a análise comparativa entre as distribuidoras no que tange ao grau de dificuldade para se combater as perdas não técnicas; e permitir estabelecer metas ou limites de redução de perdas não técnicas que respeitem a realidade socioeconômica de cada distribuidora com relação a sua área de atendimento. O índice é uma combinação linear de variáveis socioeconômicas que busca explicar a parcela de perdas não técnicas que não está relacionada à gestão da distribuidora, mas à complexidade socioeconômica da região atendida por ela. Todo o detalhamento da criação do índice, como é feito até a data de escrita deste trabalho, é dado em (ANEEL, 2008) e (ANEEL, 2014a) e é tratado a seguir.

Para a obtenção do índice de complexidade socioeconômica, parte-se do princípio de que as perdas de uma distribuidora poderiam ser decompostas da forma como mostra a equação abaixo.

$$P_{nt} = C + X\beta + IG \qquad (3.2)$$

Onde:

- $X\beta$ se refere a PNT provenientes de características socioeconômicas da área atendida pela distribuidora;
- IG se refere a PNT provenientes da ineficiência gerencial da distribuidora;
- C se refere a um conjunto de variáveis específicas da distribuidora que influenciam suas PNT não consideradas (e não observáveis) nos demais termos.

O termo $X\beta$ é o produto de duas matrizes, conforme abaixo

$$X\beta = [x_1 \quad x_2 \quad \cdots \quad x_n] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} \qquad (3.3)$$

Onde:

- $x_1, x_2 \ldots, x_n$ são variáveis socioeconômicas;
- $\beta_1, \beta_2 \ldots, \beta_n$ são coeficientes obtidos através de uma análise de regressão, em que $x_1, x_2 \ldots, x_n$ são as variáveis independentes e P_{nt} é a variável dependente.

As variáveis socioeconômicas utilizadas pela ANEEL, como detalhado na “Tabela 1: Resumo das Variáveis Analisadas” em (ANEEL, 2008), são reproduzidas na Tabela 1.

Tabela 1 - Variáveis utilizadas pela ANEEL

(continua)

Dimensão	Variável	Fonte
Violência	• Óbitos por Agressão	SIM/DATASUS
Escolaridade	• Taxa de Analfabetismo • Proporção da população acima de 15 anos de idade com até um ano de escolaridade (excluindo analfabetos) • 1 a 3 anos de escolaridade • 4 a 7 anos de escolaridade • com mais que 7 anos de escolaridade	PNAD/IBGE e CENSO/IBGE

Tabela 1 - Variáveis utilizadas pela ANEEL (conclusão)

Dimensão	Variável	Fonte
Renda	• PIB total per capita • PIB Industrial per capita • PIB Serviços per capita • PIB Comercial per capita • PIB Rural per capita	IBGE
Desigualdade	• Proporção da população que ganha até meio salário-mínimo • Desemprego	PNAD/IBGE e CENSO/IBGE
Infraestrutura	• Percentual de Domicílios com Cobertura de Abastecimento de • Água • Percentual de Domicílios com Cobertura de Abastecimento de • Esgoto Sanitário	PNAD/IBGE e CENSO/IBGE
Demografia	• População total • Número de consumidores • Densidade demográfica	IBGE e SAMP-AMP/ANEEL
Judiciário	• Relação entre processos distribuídos e Julgados nos TRF por • estado • Percentual de processos acumulados nos TRF por estado • Percentual médio de processos acumulados nos TRF por estado • no período 2001–2005	STF
Empresas	• Relação entre mercado de baixa tensão e mercado total • Relação entre mercado livre e Energia Injetada • Tarifa média na baixa tensão • Área de concessão	ANEEL
Informalidade	• Razão entre Empregados Formais e Empregados • Razão entre Empregadores e Empregadores mais • Trabalhadores por conta própria • Percentual de Pessoas que Vivem em Domicílios Subnormais • Percentual de Cheques Devolvidos	PNAD/IBGE e CENSO/IBGE
Temperatura	• Temperatura Média • Temperatura Máxima	*Weather Underground*
Outras	• IASC • Perdas no setor de Água • Percentual de Domicílios que possuem Ar-Condicionado • Percentual de Domicílios que possuem Ar-Condicionado e • possuem renda familiar até 1 Salário-Mínimo	SNIS, PNAD/IBGE CENSO/IBGE e ANEEL

Fonte: (ANEEL, 2008)

É importante notar que a própria ANEEL entende que as PNT têm relação com as características socioeconômicas da região em que cada distribuidora atua e pode-se extrapolar esse entendimento, entendendo que dentro da própria área de atuação da distribuidora há diferenças entre as PNT; ou seja, para uma dada distribuidora existirão áreas com mais PNT que outras, a depender em parte da característica socioeconômica de cada área. Esse entendimento será utilizado tanto para a nova metodologia de cálculo de PNT quanto para os modelos de *machine learning* propostos neste estudo.

3.3. Potencial de ganho com mitigação ou revisão de cálculo das PNT

Como explicado anteriormente, as PD são diretamente influenciadas pela metodologia adotada para atribuição das PNT. Um dos objetivos deste estudo é propor uma nova metodologia de cálculos das PNT para a Baixa Tensão (BT). Essa nova metodologia pode fazer com que os valores calculados de PD fossem maiores, significando que as distribuidoras têm um valor de PD maior que o atual, implicando em perda financeira para as empresas, ou que as PD fossem menores, significando que a sociedade está pagando por um valor maior do que deveria.

Foi acessada a base de perdas disponível em (ANEEL, 2021b), que possui os valores de perdas de vários anos até 2019 e reproduzidas apenas as colunas “Distribuidora”, “EI - Energia Injetada sem A1”, “BT -Mercado Baixa Tensão” e “PTot-Perda Total” do ano de 2019 na Tabela 2. A coluna “Perda Estimada BT” foi calculada de forma proporcional a coluna “BT -Mercado Baixa Tensão” com relação a coluna “EI - Energia Injetada sem A1”.

Tabela 2 - Base de Perdas 2019

(continua)

Distribuidora	EI - Energia Injetada sem A1 (MWh)	BT -Mercado Baixa Tensão (MWh)	PTot- Perda Total (MWh)	Perda Estimada BT (MWh)
Amazonas	10.997.363	3.279.105	4.852.609	1.446.912
Enel RJ	14.919.524	6.844.731	3.368.196	1.545.250
EDP SP	16.798.760	5.726.667	1.359.989	463.618
Roraima	1.262.664	706.643	321.839	180.115
CEA	1.948.139	764.394	878.999	344.894
CEAL	5.029.280	2.245.758	1.294.519	578.050
CEB	7.683.544	3.930.904	1.083.122	554.125
CEEE	9.677.033	4.840.670	1.668.468	834.605
Celesc	27.513.854	10.694.688	2.340.501	909.757
Enel GO	16.370.002	8.382.941	2.021.463	1.035.174
Celpa	12.321.494	5.629.998	3.704.538	1.692.696
Celpe	17.234.499	7.915.915	2.989.966	1.373.310
Energisa Tocantins	2.795.270	1.723.999	368.060	227.003
Cemar	7.846.280	4.957.363	1.411.098	891.547
Energisa MT	10.958.463	5.569.448	1.643.569	835.315
Cemig	51.954.975	19.148.928	7.048.872	2.597.987
Equatorial Piauí	5.007.578	2.771.524	1.216.839	673.479
Ceron	4.523.001	2.284.309	1.280.772	646.844
CHESP	144.617	95.914	15.571	10.327
Cocel	328.791	130.844	11.088	4.413
Coelba	25.062.120	12.242.771	3.833.108	1.872.462
Enel CE	14.185.562	7.370.698	2.004.613	1.041.580
Cooperaliança	218.375	110.367	-2.038	-1.030
Copel	32.886.144	14.540.586	2.294.426	1.014.479
Cosern	6.423.302	3.472.237	641.390	346.715
CPFL Paulista	36.328.834	14.990.273	3.361.588	1.387.083
CPFL Piratininga	15.434.294	5.519.237	1.174.184	419.883
DEMEI	148.241	107.543	16.440	11.927
DME-PC	566.159	203.623	24.576	8.839

Tabela 2 - Base de Perdas 2019 (conclusão)

Distribuidora	EI - Energia Injetada sem A1 (MWh)	BT -Mercado Baixa Tensão (MWh)	PTot- Perda Total (MWh)	Perda Estimada BT (MWh)
Energisa Borborema	753.725	408.800	52.023	28.216
EFLJC	19.323	11.835	898	550
EFLUL	103.991	24.334	4.259	997
Elektro	19.154.018	8.021.570	1.491.478	624.621
Eletroacre	1.337.439	813.621	256.661	156.138
Eletrocar	207.728	116.098	19.486	10.891
Eletropaulo	47.804.843	24.415.150	4.509.064	2.302.894
ELFSM	640.097	443.696	63.921	44.308
Energisa MG	1.754.032	1.060.695	196.247	118.674
Energisa MS	6.576.343	3.487.723	865.966	459.260
Energisa NF	382.464	252.813	15.521	10.259
Energisa PB	5.339.408	2.930.583	738.228	405.183
EDP ES	11.572.304	4.974.602	1.474.050	633.652
Energisa SE	3.427.166	1.750.430	399.014	203.797
Forcel	43.240	22.778	-	-
Hidropan	89.828	57.106	5.574	3.543
Iguaçu Energia	304.642	138.762	27.612	12.577
Light	37.432.256	13.583.810	9.781.364	3.549.564
MUX Energia	79.351	35.214	4.156	1.844
Sulgipe	478.105	222.628	56.254	26.195
Nova Palma	84.398	56.044	7.819	5.192
Energisa Sul Sudeste	4.880.194	2.595.615	318.161	169.219
CPFL Nova Santa Cruz	3.339.932	1.471.104	262.421	115.586
RGE Sul	22.230.330	9.046.576	2.120.062	862.754
Total				32.693.274

Fonte: (ANEEL, 2021b) e autor

Nota-se que uma variação de 1% para mais ou para menos das perdas na BT, representam aproximadamente 326.933 MWh, e considerando um valor de compra de energia de 200 R$/MWh, seriam 65 milhões de reais que deveriam ser das distribuidoras ou da sociedade.

4. MITIGAÇÃO DAS PNT: INSPEÇÕES E *MACHINE LEARNING*

Para mitigar as PNT, as distribuidoras enviam equipes de inspeção para verificar unidades consumidoras potencialmente fraudadoras. Isso é registrado, com o resultado de uma fraude descoberta ou não a cada inspeção, gerando um rico banco de dados que pode ser analisado.

O custo para envio dessas equipes só pode ser justificado se houver precisão no encontro de fraudes, caso contrário o trabalho de uma equipe que não encontra fraudes se torna um custo sem retorno. As equipes de inspeção utilizam sua expertise para encontrar fraudes. Mas, para aumentar mais a precisão da descoberta de fraudes, torna-se interessante a utilização de uma inteligência que direcione as equipes, podendo-se aplicar técnicas de *machine learning* (aprendizado de máquina) que buscam direcionar as equipes para locais com mais chance de descoberta de fraude.

Neste capítulo são apresentadas técnicas de *machine learning* para criação de um modelo que indique as unidades consumidoras com maior chance de serem fraudadoras para o direcionamento de equipes de inspeção, de modo a justificar o custo do envio dessas equipes.

Os modelos propostos neste trabalho são supervisionados e precisam ser treinados com dados classificados, ou seja, com dados que contenham a resposta de fraude encontrada ou não. Apesar de (HU, et al., 2019) utilizar uma técnica semi-supervisionada, ou seja, que utiliza dados classificados e não classificados, grande parte dos estudos revisados em (VIEGAS, ESTEVES, MELÍCIO, MENDES, & VIEIRA, 2017) e o modelo em (YAN & WEN, 2021) utilizam uma abordagem supervisionada. O modelo mais avançado, no momento de escrita deste trabalho, é o XGBoost (detalhado nas próximas seções) e tem sido utilizado, como em (YAN & WEN, 2021), para detecção de fraude com alta performance.

4.1. Dados utilizados

Para a criação de um modelo de *machine learning*, é necessário possuir dados que serão a entrada para o treinamento e aprendizado do modelo.

Neste trabalho, foram utilizados os seguintes dados reais da distribuidora Enel de São Paulo (antiga Eletropaulo):

- Dados de inspeções (disponíveis de janeiro de 2016 a agosto de 2017);
- Dados de cadastro (disponíveis em julho de 2017)
- Dados com histórico de 12 meses de faturamento de energia anteriores à época da inspeção (disponíveis de agosto de 2016 a julho de 2017);
- Dados com histórico de 12 meses de inadimplência anteriores à época da inspeção (disponíveis de agosto de 2016 a julho de 2017);
- Dados socioeconômicos do Censo de 2010 da região de atendimento da distribuidora na granularidade de setores censitários.

Todos os dados mencionados são privados, exceto os dados socioeconômicos do Censo de 2010. Apesar dos dados de inspeções estarem disponíveis para este trabalho desde janeiro de 2016, só foram utilizados a partir de agosto de 2016, pois é a data a partir da qual os dados de consumo e inadimplência estão disponíveis.

É importante salientar que os estudos em (HU, et al., 2019) e (YAN & WEN, 2021) utilizam dados de consumo de uma AMI, ou seja, de *smart meters* que contém uma alta resolução de captura de dados, enquanto que os dados disponíveis para este trabalho são dados mensais do consumo faturado pela distribuidora. Neste trabalho, procurou-se extrair o máximo do perfil de consumo com base nos dados mensais, conforme detalhado na seção sobre as variáveis utilizadas, mas seriam esperados modelos com melhores performances se houvesse dados de consumo numa resolução maior que a mensal, porém a instalação massiva de *smart meters* ainda não é a realidade brasileira.

Os dados de cadastro são de julho de 2017 e sua utilização é justificada por se assumir que esses dados praticamente não mudam com o tempo, como a classe da unidade consumidora (comercial, residencial, residencial baixa renda etc.), número de fases, etc.

A utilização de dados socioeconômicos é justificada pelo fato de a própria ANEEL reconhecer que existe relação entre PNT e indicadores socioeconômicos, conforme

mostrado na seção 3.2.1. Apesar de os dados serem antigos, de 2010, assume-se, neste trabalho, que se houve melhoras nos indicadores socioeconômicos, essas melhoras ocorreram em toda a região, e, portanto, o interesse se dá nas diferenças relativas de indicadores dentro da região. Mais detalhes sobre a forma de obtenção dos dados do IBGE e seu cruzamento com os dados de inspeção são dados na seção 4.1.1.

4.1.1. Obtenção e cruzamento com dados do IBGE

Os dados do IBGE são divididos em duas partes: os setores censitários (disponível em (IBGE, 2022b)) e os resultados socioeconômicos agregados por setores censitários (disponível em (IBGE, 2022a)).

Os dados socioeconômicos disponibilizados pelo IBGE, são disponibilizados por setor censitário, que é “[...] a unidade territorial de controle cadastral da coleta, constituída por áreas contíguas, respeitando-se os limites divisão político-administrativa, do quadro urbano e rural legal e de outras estruturas territoriais de interesse, além dos parâmetros de dimensão mais adequados à operação de coleta” (IBGE, 2010). Portanto, nada mais é do que uma região (normalmente compreende alguns quarteirões em áreas populosas) de onde se pode obter características socioeconômicas com rigor estatístico, e tal região também respeita os limites políticos (limite de distritos, cidades, etc.). Na Figura 1, é apresentada a divisão territorial por setores censitários de uma região do município de São Paulo; nela nota-se a alta granularidade, portanto, pode-se obter informações socioeconômicas com alta granularidade a partir dos dados dos setores censitários.

Figura 1 - Divisão territorial por setores censitários de uma região de São Paulo

Fonte: Autor

Para se cruzar os dados de inspeções com os dados socioeconômicos, utiliza-se as coordenadas das unidades consumidoras com inspeções e é realizada uma operação de intersecção com os setores censitários (a partir também das coordenadas desses). Assim, cada unidade consumidora inspecionada é associada a um setor censitário e, portanto, às informações socioeconômicas desse.

4.2. Fundamentos de machine learning

Convém detalhar alguns dos fundamentos de *machine learning* para que se possa entender como o modelo para detecção de fraude é criado.

Um modelo de *machine learning* para detecção de fraude, é um modelo que possui como entrada variáveis (x) numéricas e retorna um valor binário em sua saída (y), conforme Figura 2.

Figura 2 - Conceituação de um modelo com saída binária

Fonte: Autor

O modelo precisa ser treinado com um conjunto de dados com y conhecido, que é 1 para fraude encontrada e 0 para fraude não encontrada. Esse conjunto de dados é obtido a partir dos dados de inspeções, já que se conhece o resultado da inspeção. Portanto, cria-se as variáveis para cada inspeção realizada, e sabendo a saída esperada para cada inspeção, o modelo é treinado e pode-se utilizá-lo para descobrir quais outras unidades consumidoras têm potencial para fraude.

Normalmente, divide-se o conjunto de dados em uma parte para treinamento e uma parte para teste (*split train test*), o conjunto de treinamento é utilizado para treinar o modelo e o conjunto de teste é utilizado para obter as métricas esperadas para o modelo. Faz-se isso para colocar o modelo à prova com relação à parte reservada para teste, já que são inspeções desconhecidas pelo modelo, tendo-se mais confiança nas métricas no conjunto de teste do que no conjunto de treino.

Com relação às métricas, convém antes introduzir o conceito de matriz de confusão que é utilizada para se derivar métricas com relação à qualidade de previsão do modelo, ela é apresentada na Tabela 3.

Tabela 3 - Matriz de confusão

		Valor Real	
		1	**0**
Valor Predito	**1**	Verdadeiro Positivo	Falso Positivo
	0	Falso Negativo	Verdadeiro Negativo

Fonte: Autor

Verdadeiro Positivo ocorre quando o modelo acerta a previsão de uma inspeção como fraude. Verdadeiro Negativo ocorre quando o modelo acerta uma previsão de uma inspeção como não fraude. Falso Positivo ocorre quando o modelo erra a previsão de uma inspeção como fraude. Falso Negativo ocorre quando o modelo erra a previsão de uma inspeção como não fraude. Cada previsão feita pelo modelo pode ser classificada em uma, e somente uma, dessas quatro possibilidades.

Todas as previsões feitas pelo modelo são contabilizadas em cada uma das classificações da matriz de confusão e pode-se computar as seguintes métricas:

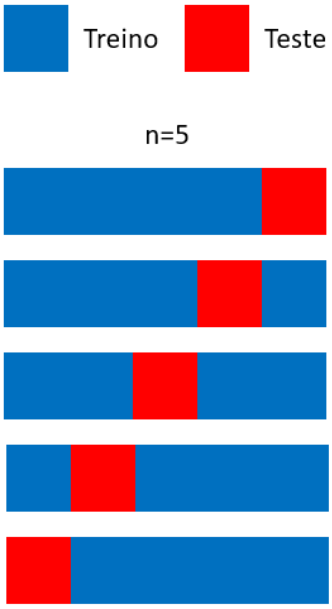
- Precisão: número de Verdadeiros Positivos dividido pelo número de previstos como 1;
- Potência: número de Verdadeiros Positivos dividido pelo número daqueles que têm como valor real 1;

Existem outras métricas que podem ser obtidas com a manipulação algébrica resultante do número de casos em cada uma das 4 possibilidades da matriz de confusão, mas os dois citados acima são os de maior interesse, pois são aqueles que se deseja maximizar em um modelo que detecta fraude. A precisão deve ser o mais próximo de 1, pois isso traz mais retorno para o custo de se enviar uma equipe de inspeção para a unidade consumidora; e a potência também deveria ser o mais próximo de 1, para não deixar de mitigar o máximo possível de perdas não técnicas.

Uma métrica muito utilizada é a combinação da precisão e da potência através de uma média harmônica, chamada de métrica F_1. Essa métrica é a de principal interesse neste trabalho para os modelos de detecção de fraude, pois leva em consideração a precisão e a potência ao mesmo tempo.

Convém agora explicar como é feito o treinamento do modelo para se obter métricas durante o treinamento. O treinamento é realizado através de uma validação cruzada. Uma validação cruzada significa nada mais do que se dividir o conjunto de treinamento (apenas o conjunto de treinamento, o conjunto de teste continua em separado para testes futuros) em n partes, realizar o treinamento do modelo com $n-1$ partes e calcular métricas de interesse para a parte que ficou de fora do treinamento (parte de teste da validação cruzada); depois escolhe-se outro conjunto de $n-1$ partes para realizar o treinamento e esse processo se repete n vezes, sendo que em cada uma das n vezes, cada uma das partes é separada para se obter métricas de interesse. Uma ilustração desse processo para $n=5$ é apresentada na Figura 3. Cada uma desses n partes é chamada de *fold*.

Figura 3 - Ilustração de uma validação cruzada



Fonte: Autor

Esse procedimento de validação cruzada foi utilizado neste trabalho para obtenção das métricas precisão e potência durante o treinamento. Ainda se tem o conjunto de dados de teste, separados inicialmente, do qual se pode obter mais métricas para avaliação do modelo.

Convém falar também sobre a otimização dos parâmetros dos modelos. Cada tipo de modelo (regressão logística, árvores de decisão etc.), que são detalhados na

próxima seção, possuem parâmetros que precisão ser escolhidos *a priori*, ou seja, antes de realizar o treinamento. Existem duas formas de se obter os melhores parâmetros. A primeira forma, seria utilizar a força bruta e testar todas as combinações possíveis entre os parâmetros, dado um intervalo de busca para cada um deles, e escolher aqueles que resultam na melhor métrica na validação cruzada; essa forma é chamada de *grid search*. A segunda forma, utilizada neste trabalho por ser mais rápida, é utilizando uma otimização bayesiana, através da biblioteca "hyperopt" do Python, não convém detalhar a forma de funcionamento por ser demasiadamente complexa para inclusão neste trabalho, mas mais detalhes podem ser encontrados em (BERGSTRA, YAMINS, & COX, 2013). Neste trabalho, a biblioteca "hyperopt" foi utilizada durante a validação cruzada para maximizar a métrica F_1 para se encontrar os melhores parâmetros para cada um dos modelos.

4.3. Tipos de modelos utilizados

Os modelos utilizados neste trabalho foram:

- Regressão Logística;
- *Random Forest*;
- *Extra Trees*;
- XGBoost.

A seguir, nas próximas seções, os modelos são explicados de uma maneira geral, sem entrar nos pormenores de cada um.

4.3.1. Regressão Logística

A regressão logística tem como objetivo obter um modelo que venha predizer valores binários a partir da combinação linear de um conjunto de variáveis independentes. Em verdade, a regressão logística prediz uma probabilidade (valores

entre 0 e 1), sendo 0 sem dúvida a saída binária 0, e a probabilidade 1 sem dúvida a saída binária 1.

A regressão logística difere de uma regressão linear comum no sentido de haver uma função sigmoide pela qual a combinação linear das variáveis independentes é passada. Uma função sigmoide é uma função com forma de “S”, tendo uma saturação inferior e uma saturação superior. A função sigmoide mais utilizada é a

$$S_x = \frac{1}{1 + e^{-x}}, \tag{4.1}$$

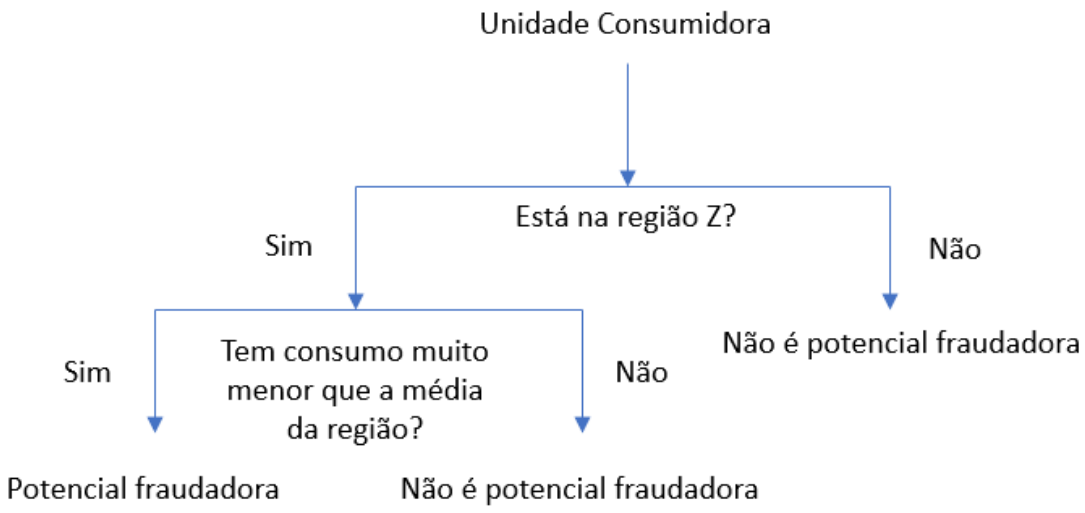
em que x seria a combinação linear das variáveis independentes, cada uma com seus coeficientes a serem estimados pela regressão logística.

Um procedimento estatístico é utilizado para se achar os melhores parâmetros para a combinação linear das variáveis independentes, e desse ponto em diante, obtém-se um modelo que tem como entrada essas variáveis e como saída uma probabilidade. Utiliza-se o ponto de corte de probabilidade 0,5, de modo que, quando maior ou igual a 0,5, a saída será 1, e 0, caso contrário. Maiores detalhes sobre regressão logística podem ser obtidos em (P. & P., 1999).

4.3.2. *Random Forest*

Para entender o *Random Forest*, antes é necessário entender árvores de decisão, já que é uma parte constituinte do *Random Forest*. Uma árvore de decisão, nada mais é do que um conjunto de regras utilizado para classificar um conjunto de dados com base em seus atributos. Por exemplo, suponha que tenhamos a seguinte árvore de decisão, completamente hipotética, na Figura 4, que tenta classificar uma unidade consumidora em potencial fraudadora ou não:

Figura 4 - Exemplo de árvore de decisão



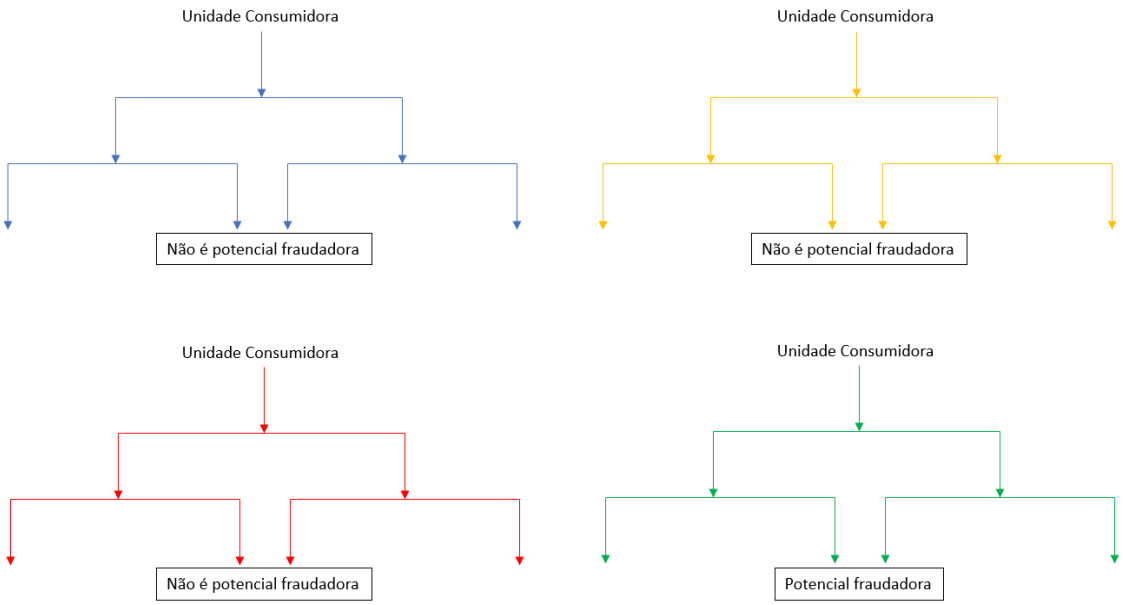
Fonte: Autor

Essa árvore tenta classificar uma unidade consumidora, baseando-se em seus atributos (que também podemos chamar de variáveis), em potencial fraudadora ou não. Ela possui dois nós de decisão: o primeiro se baseia na região da unidade consumidora e o segundo se baseia na média de consumo com relação à região. A função da árvore de decisão é, portanto, classificar um conjunto de dados com base em seus valores, através de um conjunto de regras para cada uma das variáveis do conjunto de dados.

Antes de continuar com a explicação do *Random Forest*, é importante salientar que durante o treinamento de uma árvore de decisão, para cada etapa de divisão de um de nó, ela irá selecionar a variável de entrada para divisão que melhor separa o conjunto de dados com relação à variável reposta no seu melhor ponto de corte. Por exemplo, supondo que durante o treinamento de uma árvore de decisão, a primeira variável que melhor separa os fraudadores dos não fraudadores é o consumo médio, então, o primeiro nó da árvore irá separar os dados em dois conjuntos de dados com relação a um ponto de corte do consumo médio, e esse ponto de corte é aquele que melhor separa os conjuntos.

No *Random Forest*, tem-se várias árvores de decisão que classificam individualmente uma entrada de variáveis. Remetendo-se ao caso deste trabalho, tem-se várias árvores de decisão que irão classificar uma unidade consumidora em potencial fraudadora ou não. Cada árvore tem como o poder de um “voto”, e a classificação com mais “votos” define a classificação final para aquela unidade consumidora. Na Figura 5, é apresentado um exemplo de *Random Forest* com 4 árvores de decisão (os detalhes de cada árvore são omitidos), em que 3 delas “votam” que a unidade consumidora não é potencial fraudadora e 1 “vota” no contrário, portanto o resultado é que a unidade consumidora não é fraudadora.

Figura 5 - Exemplo de Random Forest com 4 árvores de decisão



Fonte: Autor

O princípio do *Random Forest* é que muitas de árvores de decisão não relacionadas, operando através dessa forma de “votação”, terá um desempenho melhor que uma única árvore de decisão. Para isso, portanto, o erro de cada uma das árvores de decisão precisa ter baixa correlação com o erro das outras.

Para criar essas árvores com baixa correlação entre si, o *Random Forest* lança mão de duas técnicas que usam o fato de uma árvore de decisão ser altamente sensível aos dados utilizados para seu treinamento e às variáveis utilizadas para criar a separação de um nó. A primeira técnica, chamada de *bootstrap aggregation*, faz com que cada árvore do *Random Forest* pegue uma amostra aleatória do conjunto

de dados de treinamento e coloque essa amostra de volta, podendo obter novamente a própria amostra. Isso faz com que as árvores tenham uma diferença significativa entre si, já que são sensíveis aos dados de entrada. A outra técnica é considerar em cada árvore, a cada passo de divisão de um nó, que um subconjunto aleatório de variáveis é utilizado para criar a divisão de um nó, novamente, utilizando-se o fato de que as árvores são sensíveis à variável escolhida para divisão do nó. Essas duas técnicas em conjunto fazem com que árvores bem diferentes entre si sejam criadas, que é um dos objetivos do *Random Forest*. Mais detalhes podem ser encontrados em (BREIMAN, 2001).

4.3.3. *Extra Trees*

Extra Trees vem de *Extremely Randomized Trees*. Esse modelo é como o *Random Forest*, utiliza um grande conjunto de árvores de decisão para a classificação final. Porém, há diferenças na forma como as árvores são construídas.

A primeira diferença é que, na construção da árvore durante o treinamento, todas as amostras são utilizadas para o treinamento sem que sejam colocadas de volta, então uma árvore não passa pela mesma amostra mais de uma vez. A segunda diferença é que, durante a divisão de um nó, ainda é considerado um subconjunto aleatório das variáveis e a variável que melhor divide o nó, porém o ponto de corte dessa variável que melhor divide o nó é escolhido de maneira aleatória e não como o melhor ponto de corte.

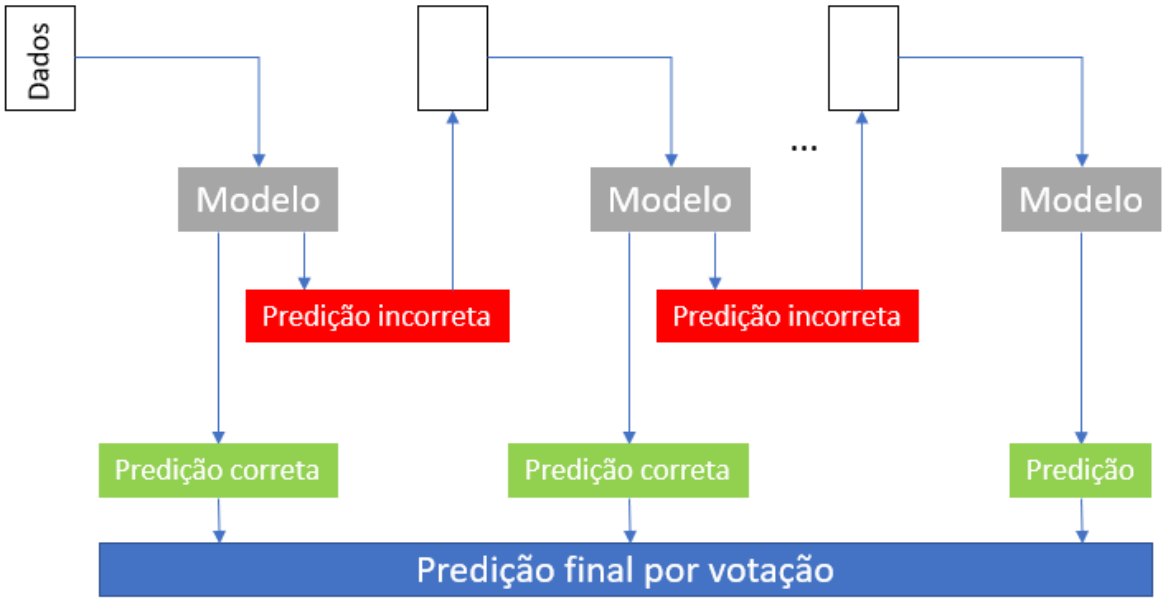
Essas diferenças têm o objetivo de gerar árvores bem distintas entre si, o mesmo objetivo do *Random Forest*, porém com um grau maior de aleatoriedade devido aos pontos de cortes dos nós serem aleatórios e não os pontos ótimos.

4.3.4. XGBoost

XGBoost vem de *Extreme Gradient Boosting*. É um dos modelos mais recentes e que vem ganhando espaço com um classificador de grande performance.

Para entender o funcionamento do XGBoost é necessário entender primeiro que o funcionamento do algoritmo *boosting*. *Boosting* é uma técnica para agregar modelos de forma sequencial, ao contrário do *Random Forest* que agrega de forma paralela. No *boosting*, é criado um modelo para tentar predizer os dados que o modelo anterior errou, de uma forma sequencial, e a saída final será a votação de todos as saídas corretas dos modelos, uma ilustração de um modelo com *boosting* é mostrada na Figura 6.

Figura 6 - Predição de um modelo usando *boosting*



Fonte: Autor

Há o caso particular de *boosting*, chamado de *gradient boosting*, em que os erros são minimizados usando o método do gradiente, ou seja, a cada adição de um novo modelo ao *boosting*, este é adicionado de forma a minimizar o erro pelo método do gradiente, e o processo se repete até que seja atingido o número máximo de modelos ou que o erro esteja abaixo de um limiar. O XGBoost utiliza o método de *gradient boosting* adicionando árvores de decisão até que o número máximo de árvores seja atingido ou que o erro esteja abaixo de um limiar. Mais detalhes podem ser obtidos em (CHEN & C., 2016).

4.4. Variáveis utilizadas

Com relação às variáveis, convém detalhar como utilizar variáveis de cadastro, que são variáveis categóricas, ou seja, variáveis que não têm um valor numérico intrínseco em si mesmas, mas que devem ser transformadas em valores numéricos para que possam ser usadas como entrada para os modelos.

Para variáveis categóricas, é utilizado o processo de criação de *dummies*, ou seja, são criadas variáveis para cada categoria de uma variável categórica. Por exemplo, supondo que se deseja utilizar a variável de número de fases no modelo, sabe-se que as possibilidades dessa variável categórica são monofásico, bifásico e trifásico. Nesse caso haverá 3 variáveis numéricas uma que é 1 quando a unidade consumidora é monofásica, caso contrário 0; outra que é 1 quando é bifásica, caso contrário 0; e por fim outra que é 1 quando é trifásica, caso contrário 0; e somente uma das 3 variáveis pode ser 1 para uma mesma entrada.

As seguintes variáveis foram criadas para caracterizar cada uma das inspeções e utilizadas com entrada para o treinamento e teste dos modelos:

1. Média dos últimos 12 meses de consumo de energia na época da inspeção;
2. Quantil 0,10 dos últimos 12 meses de consumo de energia na época da inspeção;
3. Quantil 0,25 dos últimos 12 meses de consumo de energia na época da inspeção;
4. Quantil 0,75 dos últimos 12 meses de consumo de energia na época da inspeção;
5. Quantil 0,90 dos últimos 12 meses de consumo de energia na época da inspeção;
6. Valor 1 se unidade consumidora esteve inadimplente nos últimos 3 meses na época da inspeção, caso contrário valor 0;
7. Valor 1 se unidade consumidora esteve inadimplente nos últimos 6 meses na época da inspeção, caso contrário valor 0;

8. Valor 1 se unidade consumidora esteve inadimplente nos últimos 12 meses na época da inspeção, caso contrário valor 0;
9. Máximo degrau de consumo em kWh (diferença positiva entre um mês e o próximo);
10. Mínimo degrau de consumo em kWh (diferença negativa entre um mês e o próximo);
11. Amplitude de consumo em kWh (diferença entre o consumo máximo e mínimo);
12. Longitude em graus arredondada para a segunda casa decimal (para levar em consideração a região);
13. Latitude em graus arredondada para a segunda casa decimal (para levar em consideração a região);
14. Razão entre a média de consumo da unidade e a média de consumo da região;
15. Variáveis *dummies* da categoria do consumidor (Comercial, Consumo próprio, Iluminação pública, Industrial, Poder público, Residencial-B Optante, Residencial-Baixa Renda, Residencial-Baixa Renda BPC, Residencial-Plena, Rural e Serviço Público)
16. Variáveis *dummies* do número de fases do consumidor (monofásico, bifásico, trifásico);
17. Média de moradores por domicílio;
18. Rendimento mensal médio dos responsáveis por domicílios;
19. Razão de domicílios alugados;
20. Razão de domicílios com abastecimento de água;
21. Razão de domicílios com coleta de esgoto;
22. Razão de domicílios com coleta de lixo;
23. Razão de pessoas alfabetizadas com 10 ou mais anos.

As variáveis 12 e 13 foram utilizadas para se levar em consideração a região em que a unidade consumidora está e o arredondamento na segunda casa decimal foi

utilizado, pois equivale a um raio de aproximadamente 1km, podendo ser verificado através do software QGIS (QGIS, 2022). A variável 14 é calculada utilizando-se essas coordenadas arredondadas para se definir a região da unidade consumidora.

4.5. Treinamento dos modelos e resultados

O *notebook* Python utilizado para a criação das variáveis e treinamento dos modelos está disponível no Apêndice A.

Os dados de inspeções foram separados em um conjunto para testes e um conjunto para treino. Foi definida a data de referência de maio de 2017 como a data máxima para treino, portanto os meses de agosto de 2016 a maio de 2017 (10 meses) foram utilizados para treinamento dos modelos e os meses de junho e julho de 2017 foram utilizados para teste. É importante salientar que, no caso do teste do modelo, deseja-se utilizar o modelo como ele seria utilizado por uma distribuidora em um caso real, sendo assim, como a data inicial de teste é de junho de 2017, os dados de consumo só estariam disponíveis até o mês da data inicial. Por isso, para geração dos dados de teste, as variáveis de consumo tiveram a data de referência de junho de 2017, tanto para a criação das variáveis das inspeções do mês de julho, quanto para o próprio mês de junho; já as outras variáveis são agnósticas à data de referência – dados socioeconômicos são sempre de 2010.

A validação cruzada usada foi de 10 partes, e ela foi utilizada para otimizar os parâmetros dos modelos através da biblioteca Python “hyperopt”, com o objetivo de maximizar a média dos F_1 (média harmônica da precisão e potência) obtidos em cada uma das partes da validação cruzada.

Inicialmente, não foi utilizado nenhuma subamostragem ou superamostragem com relação à variável alvo – “fraude encontrada” (1) ou “fraude não encontrada” (0). Nesse caso, o número de inspeções disponíveis é de 195.665. Os dados disponíveis para treinamento possuem uma proporção muito maior da variável alvo “fraude não encontrada” (0), sendo que apenas 5% dos dados de treino apresentam como variável alvo o valor “fraude encontrada” (1). Isso é chamado de desbalanceamento do conjunto de dados e gera baixa performance nos modelos quando não tratado

(LEMAÎTRE, NOGUEIRA, & ARIDAS, 2016). Mais adiante, nesta seção, é mostrado a importância de se realizar uma subamostragem ou superamostragem.

Além disso, os modelos foram treinados com conjuntos crescentes de variáveis, para se verificar se a adição de novas variáveis traria ganho na performance dos modelos. Os conjuntos de variáveis utilizados – seguindo numeração apresentada na seção 4.4 – foram: variáveis de 1 a 8; variáveis de 1 a 14; variáveis de 1 a 16 e variáveis de 1 a 23.

Os resultados da validação cruzada dos modelos, com otimização dos parâmetros (maximização do F_1), sem subamostragem ou superamostragem são mostrados na Tabela 4.

Tabela 4 - Resultados da validação cruzada (sem super ou subamostragem)

Modelo	**Variáveis**	**Validação cruzada**					
		Precisão (*fold* de teste)	**Precisão (*folds* de treino)**	**Potência (*fold* de teste)**	**Potência (*folds* de treino)**	**F_1 (*fold* de teste)**	**F_1 (*folds* de treino)**
Regressão Logística	1 a 8	0,056	0,060	0,006	0,006	0,011	0,011
Random Forest	1 a 8	0,163	0,976	0,007	0,110	0,013	0,198
Extra Trees	1 a 8	0,154	1,000	0,001	0,042	0,002	0,080
XGBoost	1 a 8	0,094	0,982	0,032	0,746	0,048	0,848
Regressão Logística	1 a 14	0,000	0,021	0,000	0,000	0,000	0,000
Random Forest	1 a 14	0,338	0,993	0,019	0,292	0,036	0,452
Extra Trees	1 a 14	0,230	0,999	0,006	0,180	0,012	0,305
XGBoost	1 a 14	0,157	0,977	0,061	0,890	0,088	0,931
Regressão Logística	1 a 16	0,000	0,021	0,000	0,000	0,000	0,000
Random Forest	1 a 16	0,349	0,995	0,019	0,288	0,036	0,447
Extra Trees	1 a 16	0,266	0,999	0,008	0,187	0,016	0,315
XGBoost	1 a 16	0,153	0,981	0,066	0,892	0,092	0,934
Regressão Logística	1 a 23	0,078	0,129	0,000	0,000	0,001	0,001
Random Forest	1 a 23	0,372	0,997	0,021	0,386	0,039	0,556
Extra Trees	1 a 23	0,273	0,999	0,011	0,219	0,022	0,359
XGBoost	1 a 23	0,177	0,979	0,082	0,953	0,112	0,966

Fonte: Autor

Nota-se, como esperado, uma baixa performance dos modelos, o melhor F_1 no teste da validação cruzada foi de 0,112, um valor baixo, devido ao desbalanceamento do conjunto de dados. Para tratar esse problema, foi utilizada uma subamostragem no conjunto de dados, fazendo com que o número de inspeções com “fraude

encontrada” (1) e “fraude não encontrada” (0) fossem iguais. Não foi utilizada superamostragem, pois a utilização de uma superamostragem que replica dados cria resultados não confiáveis na validação cruzada, já que seria altamente provável que a mesma amostra estaria presente nos *folds* de treino e no *fold* de teste; e a utilização de uma superamostragem que interpola amostragem não é aplicável devido à presença de variáveis categóricas.

Com a subamostragem, o número de inspeções disponíveis para a validação cruzada é de 19.452. Os resultados dessa validação cruzada são apresentados na Tabela 5.

Tabela 5 - Resultados da validação cruzada com subamostragem

Modelo	**Variáveis**	**Validação cruzada**					
		Precisão (*fold* de teste)	**Precisão (*folds* de treino)**	**Potência (*fold* de teste)**	**Potência (*folds* de treino)**	**F_1 (*fold* de teste)**	**F_1 (*folds* de treino)**
Regressão Logística	1 a 8	0,561	0,561	0,447	0,446	0,497	0,497
Random Forest	1 a 8	0,588	0,595	0,724	0,730	0,649	0,655
Extra Trees	1 a 8	0,591	0,630	0,712	0,759	0,646	0,689
XGBoost	1 a 8	0,571	0,576	0,772	0,778	0,656	0,662
Regressão Logística	1 a 14	0,560	0,561	0,441	0,442	0,493	0,495
Random Forest	1 a 14	0,643	0,748	0,697	0,800	0,669	0,773
Extra Trees	1 a 14	0,637	0,897	0,690	0,938	0,663	0,917
XGBoost	1 a 14	0,650	0,701	0,689	0,741	0,669	0,721
Regressão Logística	1 a 16	0,560	0,561	0,441	0,442	0,493	0,495
Random Forest	1 a 16	0,642	0,778	0,711	0,864	0,675	0,819
Extra Trees	1 a 16	0,635	0,792	0,711	0,887	0,671	0,837
XGBoost	1 a 16	0,659	0,779	0,683	0,807	0,670	0,793
Regressão Logística	1 a 23	0,697	0,698	0,105	0,107	0,183	0,185
Random Forest	1 a 23	0,657	0,961	0,705	0,994	0,680	0,978
Extra Trees	1 a 23	0,651	0,896	0,717	0,970	0,683	0,932
XGBoost	1 a 23	0,660	0,737	0,691	0,774	0,675	0,755

Fonte: Autor

Nota-se a melhora significativa na performance dos modelos com o tratamento do desbalanceamento do conjunto de dados através da subamostragem. O melhor F_1 foi de 0,683, que não é um valor baixo. Nota-se também a melhora da performance dos modelos conforme são adicionadas mais variáveis.

A partir daí, deve-se escolher um modelo para ser utilizado com os dados separados para teste e verificar suas métricas quanto a esses dados. Obviamente, um dentre os modelos com variáveis de 1 a 23 é o que deve ser escolhido. Cuidado deve ser tomado ao escolher um dentre esses modelos, já que, nesse caso, escolher aquele com maior F_1 no teste da validação cruzada não seria uma boa opção, pois nota-se um F_1 muito alto no treinamento da validação cruzado (valores maiores que 0,9), o que é um indício de *overfitting*, que ocorre quando o modelo se torna muito ajustado aos dados de treino, mas esse aprendizado extremo das características dos dados de treino inclui características que não estão presentes em todo o universo de dados – que inclui os dados de teste. Por esse motivo, o melhor modelo é aquele que possua o valor de F_1 no teste da validação cruzada o mais alto possível, mas que não possua um valor de F_1 no treinamento da validação cruzada muito mais alto que o valor anterior. Logo, o modelo escolhido é o XGBoost que possui um F_1 no teste da validação cruzada de 0,675 e um F_1 no treinamento da validação cruzada de 0,755.

O modelo escolhido foi utilizado nos dados de teste, que são 45.425 inspeções. Na Tabela 6 são mostrados os resultados do teste do melhor modelo. Nota-se uma piora na performance do modelo quando utilizado nos dados de teste se comparado a sua performance na validação cruzada, isso deve ocorrer devido à utilização da subamostragem, que faz perder muitos dados que poderiam ser utilizados no treinamento do modelo. Como já mencionado, utilizar uma superamostragem não era um bom caminho a se seguir.

Tabela 6 – Resultados do teste do melhor modelo

Precisão	**Potência**	**F_1**
0,067	0,646	0,121

Fonte: Autor

O modelo, além de ter como saída o valor 0 ou 1, pode gerar como saída a probabilidade de o cliente ser um fraudador – valor real entre 0,0 e 1,0. O modelo, por padrão, considera um potencial cliente fraudador – saída 1 – quando a probabilidade é maior que 0,5. Apesar de sua baixa performance geral nos dados de teste, deseja-se analisar se as probabilidades de fraude geradas pelo modelo nos dados de teste possuem utilidade para o direcionamento de equipes de inspeção de

uma distribuidora. Para isso, foram geradas as probabilidades de fraude das 45.425 inspeções utilizadas para teste e estas foram ordenadas de forma decrescente quanto a essa probabilidade. Lembrando que esse número de inspeções foi realizado em dois meses, entende-se, que, numa situação real, a distribuidora poderia realizar a metade disso em um mês, ou seja, 22.713 inspeções. Logo, considerando que as primeiras 22.713 inspeções ordenadas de forma decrescente pela probabilidade de fraude – dada pelo melhor modelo – seriam inspecionadas pela distribuidora, a precisão das equipes de inspeção seria o número real de fraudes nessas 22.713 inspeções. Realizando esse procedimento nos dados de teste, tem-se que a precisão das equipes de inspeções seria de 5,6%, enquanto a precisão obtida na realidade pela distribuidora – número de fraudes das 45.425 inspeções – foi de 3,7%, ou seja, um incremento de 51% na precisão das equipes de inspeções se o direcionamento delas fosse dado pela maior probabilidade de fraude gerada pelo melhor modelo obtido.

5. NOVA METODOLOGIA DE CÁLCULO PARA PNT

A metodologia proposta nesta dissertação para o cálculo das PNT, e consequentemente das Perdas Não-Técnicas, busca utilizar uma abordagem mais realista, usando dados de inspeções realizadas nas unidades consumidoras da BT. As distribuidoras realizam inspeções em unidades consumidoras e mantém um banco de dados ativo com os resultados dessas inspeções: não encontrada nenhuma regularidade, encontrado defeito no medidor, encontrada uma adulteração no medidor.

A abordagem mais realista pode beneficiar a distribuidora em termos de manutenção e planejamento e em termos de uma tarifa mais adequada à realidade da área de atuação da distribuidora, já que a atribuição de Perdas Não-Técnicas altera o resultado de Perdas Técnicas, pois a atribuição de Perdas Não-Técnicas é um passo anterior ao cálculo de fluxo de potência que fornece o resultado de Perdas Técnicas.

Na forma regulatória atual de cálculo de perdas, a ANEEL atribui cargas virtuais a cada unidade consumidora de forma proporcional ao consumo da mesma, então em áreas mais ricas, onde o consumo em geral é maior, terá mais PT e PNT do que em áreas mais pobres, onde o consumo em geral é menor. Porém, isso é contrário ao entendimento geral e como mostra (YURTSEVEN, 2015), onde é mostrado uma proporcionalidade negativa entre consumo ilegal de energia elétrica e renda.

Em (DONADEL, et al., 2009), foi proposta uma forma de estimar as perdas não técnicas levando-se em consideração os dados de inspeções. Este trabalho difere da técnica do estudo citado e já detalhado no capítulo 2. No estudo citado, os autores estimam o número mínimo de inspeções, por região, para representar com rigor estatístico a proporção de fraudes da região, se o número de inspeções estiver abaixo do mínimo, eles então agrupam regiões até que se obtenha o valor mínimo de inspeções necessárias, formando um *cluster*. Em cada *cluster* é calculada a proporção de consumidores irregulares daquele *cluster*, como a divisão entre o número de irregularidades e o número de inspeções do *cluster*, e, portanto, o número de consumidores irregulares será essa proporção multiplicada pelo número de consumidores do *cluster*. Dado o número F_j estimado de consumidores

irregulares para o *cluster* j, são selecionados os F_j consumidores com consumo médio mais próximo da média do consumo do *cluster* para terem seus consumos ajustados. O ajuste no consumo dos consumidores selecionados, que representará as perdas não técnicas daquele *cluster*, é feito analisando-se o consumo médio antes e após a inspeção com irregularidade encontrada para cada *cluster*.

Já a abordagem proposta em (HENRIQUES, CORRÊA, FORTES, BORBA, & FERREIRA, 2020) propõe a instalação de sensores de temperatura ao longo da rede para estimativa de perdas técnicas e a partir do balanço de energia a estimativa das perdas não técnicas ao longo da rede, indicando, portanto, possíveis pontos da rede onde deve haver fraude ou roubo de energia.

A abordagem proposta não requer nenhum *hardware* adicional, ela utiliza os dados de inspeções realizadas pela distribuidora em conjunto com a metodologia *Kriging* (P. & P., 1999) para extrapolar para todas as regiões um certo peso de potencial de fraude no consumo de energia elétrica, assumindo que existe uma correlação espacial entre possíveis unidades consumidoras fraudadoras. Esse peso obtido pelo método *Kriging* é utilizado para distribuir as cargas virtuais de PNT, onde o peso for maior, maior será a carga virtual de PNT. Importante salientar que a proposta de uso desse peso é apenas para a baixa tensão, onde, em geral, se concentram a maior parte das perdas não técnicas.

5.1. Detalhamento da metodologia atual de cálculo de PNT e PT

A ANEEL utiliza a Base de Dados Geográfica da Distribuidora (BDGD) desde 2017 (ANEEL, 2016) como uma de suas fontes para os cálculos de perdas técnicas e não-técnicas. Reitera-se que nesta dissertação todos os dados utilizados para os cálculos são reais da distribuidora Enel de São Paulo (antiga AES Eletropaulo). A BDGD contém informações como:

- Informações dos consumidores, incluindo o consumo mensal utilizado para o cálculo do fluxo de potência;
- Informações dos segmentos de média e baixa tensão como impedância e outras informações relevantes para o cálculo do fluxo de potência;

- Informações sobre reguladores e chaves relevante para o cálculo do fluxo de potência;
- Informações relevantes sobre subestações e transformadores para o cálculo do fluxo de potência;
- Informações da energia medida nos alimentadores, necessário para o cálculo do fluxo de potência.

Através da Lei de Acesso à Informação (Fala.BR, 2020), é possível obter acesso às BDGD de todas as distribuidoras do Brasil. Com as informações disponíveis na BDGD, depois de preparados os dados, é possível utilizar o software OpenDSS (EPRI, 2022) para realizar um cálculo de fluxo de potência. Em verdade, é necessário ter a curva de carga de cada unidade consumidora, mas isso já é feito pela ANEEL através de uma metodologia na qual não é o interesse desta dissertação se aprofundar.

Para a realização do cálculo de fluxo de potência, a ANEEL disponibiliza arquivos SQL, um executável chamado ProgGeoPerdas em (ANEEL, 2021a). Em posse desses arquivos e da BDGD de interesse, pode-se realizar o fluxo de potência e obter os dados de PD, PNT e PT de cada alimentador da distribuidora referente a BDGD utilizada.

O arquivo SQL disponibilizado pela ANEEL é utilizado para criar um novo banco de dados chamado GeoPerdas e o programa executável ProgGeoPerdas é executado para executar o fluxo de potência e o método iterativo que atribui cargas virtuais a cada unidade consumidora de forma proporcional ao consumo de cada unidade consumidora para representar as PNT e aumenta ou diminui o valor dessas cargas (ainda de forma proporcional ao consumo de cada uma) até que a soma das energias realmente consumidas mais a soma das cargas virtuais mais a soma das PT seja próximo do valor declarado pela distribuidora como energia medida no alimentador que atende essas unidades consumidoras. O ProgGeoPerdas utiliza o OpenDSS para realizar o fluxo de potência, e seu código fonte em C# também se encontra disponível em (ANEEL, 2021a).

O processo de cálculo, analisando o código fonte do ProgGeoPerdas, poderia ser resumido nos seguintes passos:

I. Considere inicialmente as cargas virtuais que representam as PNT todas como com valor 0;
II. Execute o fluxo de potência para o alimentador em questão;
III. Calcule a diferença entre a energia medida no alimentador e a soma de energia consumida pelas unidades consumidoras mais as perdas técnicas;
IV. Utilize a diferença calculada no passo III para distribuir cargas virtuais a todas as unidades consumidoras, para representar as PNT, de forma proporcional à energia consumida por cada unidade consumidora;
V. Volte ao passo II se a diferença calculada no passo IV for maior que certa tolerância, senão finalize as iterações.

Quando a iteração é finalizada, tem-se como PNT a soma das cargas virtuais atribuídas a cada unidade consumidora e a PT é o resultado do fluxo de potência. A PD é a soma de PT mais PNT.

5.2. Detalhamento da metodologia proposta para cálculo de PNT e PT

A metodologia proposta apenas altera o passo IV da metodologia regulatória utilizada pela ANEEL, apenas para a BT. Em vez de se acoplar uma carga virtual a cada unidade consumidora que represente a PNT com valor proporcional ao seu consumo, acopla-se uma carga virtual proporcional ao consumo e a um fator de potencial de fraude da região em que se localiza a unidade consumidora, doravante chamado de Índice Potencial de Fraude (IPF).

5.2.1. Cálculo do Índice Potencial de Fraude

A obtenção do IPF é feita utilizando os dados georreferenciados das inspeções realizadas pela distribuidora em conjunto com a metodologia *Kriging* (CHILÈS & DESASSIS, 2018). Nesta seção, será detalhada a obtenção de tal índice.

Primeiramente, utiliza-se os dados georreferenciados de inspeções com fraudes encontradas. Então, cria-se um histograma de duas dimensões com as fraudes

encontradas, ou seja, é associado a cada quadrilátero do histograma um número igual ao número de fraudes dentro desse quadrilátero. O número de caixas (*bins*) do histograma é arbitrário. Esse histograma em duas dimensões, ou mapa de quadriláteros, é passado para o método *Kriging* que interpola toda a região para atribuir um valor a qualquer ponto no mapa.

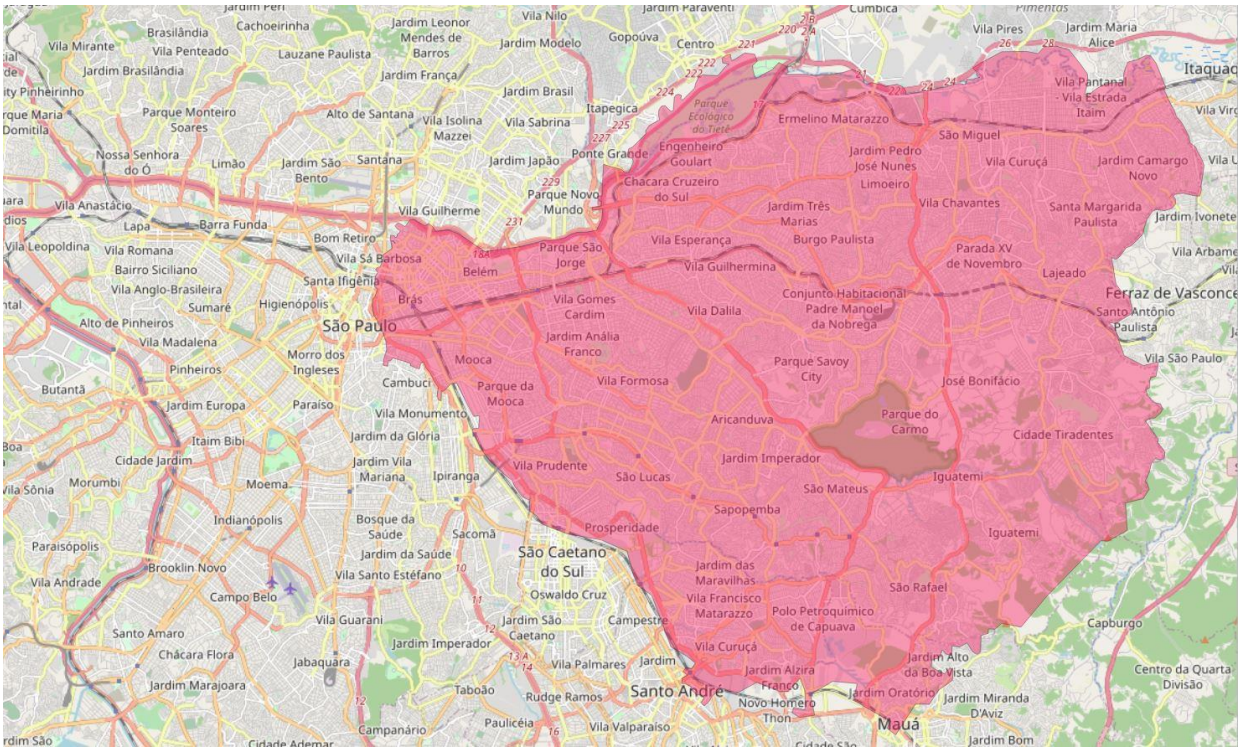
Cruzando a coordenada de cada unidade consumidora com esse mapa resultante do *Kriging*, obtém-se o IPF de cada unidade consumidora.

5.3. Implementação da metodologia proposta e resultados

Foi escolhida uma área piloto para se comparar os resultados da metodologia regulatória com a metodologia proposta. A área se refere à área leste do município de São Paulo, pois obteve-se acesso a dados privados das inspeções realizadas em 2016 pela distribuidora que na época se chamava AES Eletropaulo (chamada de Enel na data de escrita desta dissertação), foram 10.363 fraudes encontradas com dados válidos (número de instalação consistente), em toda a área atendida pela Enel São Paulo. Na Figura 7 é mostrada a área piloto escolhida, a figura foi criada utilizando o software QGIS (QGIS, 2022). Essa área piloto é apresentada no formato GeoJSON (WIKIPEDIA, 2022) no Apêndice C.

A BDGD utilizada no estudo é de 2019, pois passou-se alguns anos desde que a ANEEL exigiu que esses dados fossem entregues nesse formato, e, portanto, as distribuidoras tiveram que se adaptar a esse novo formato, logo a BDGD de 2019 é mais confiável quanto a qualidade dos dados. Nota-se a diferença de data entre os dados de inspeções (2016) e da BDGD (2019), mas assume-se que os locais de fraude, por estarem relacionados a indicadores socioeconômicos, continuam sendo aproximadamente os mesmos.

Figura 7 - Área piloto escolhida (em rosa)

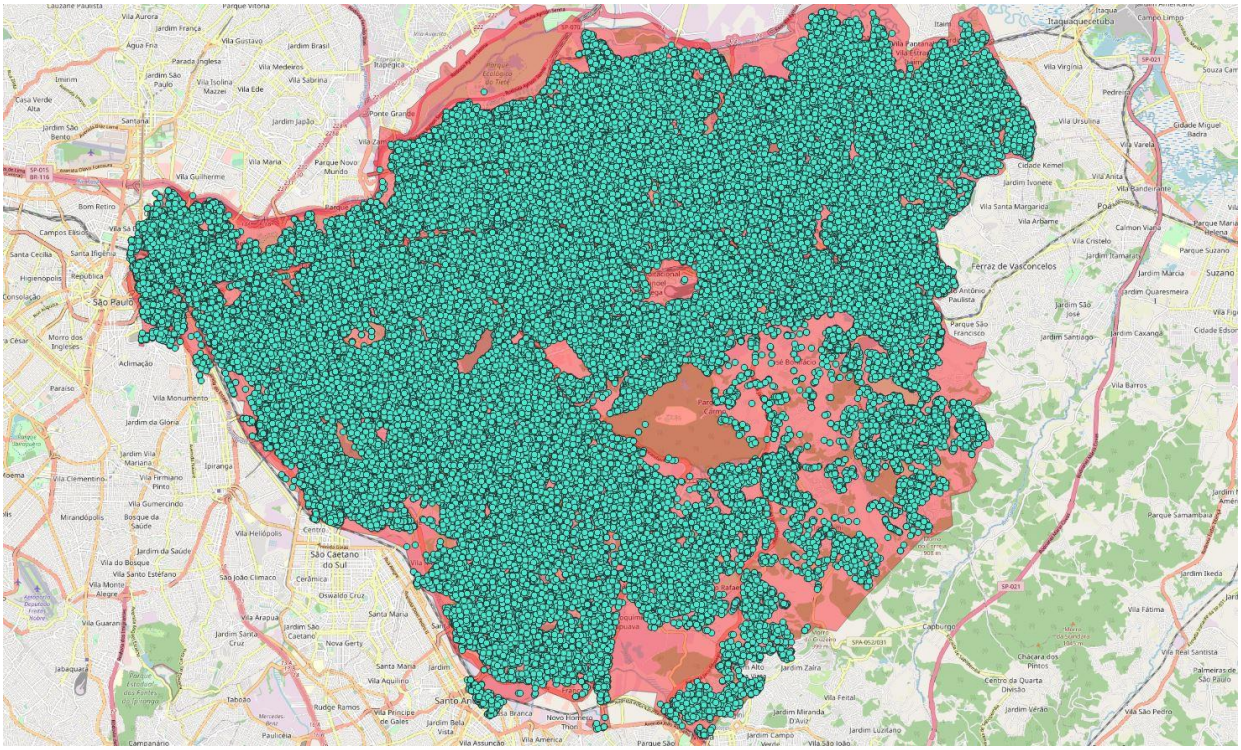


Fonte: Autor

Essa área possui 588 alimentadores e aproximadamente 2 milhões de unidades consumidoras de baixa tensão.

A área escolhida possui inspeções realizadas em 2016 em quase sua totalidade, como pode ser visto na Figura 8.

Figura 8 - Inspeções (pontos verdes) realizadas na área piloto em 2016

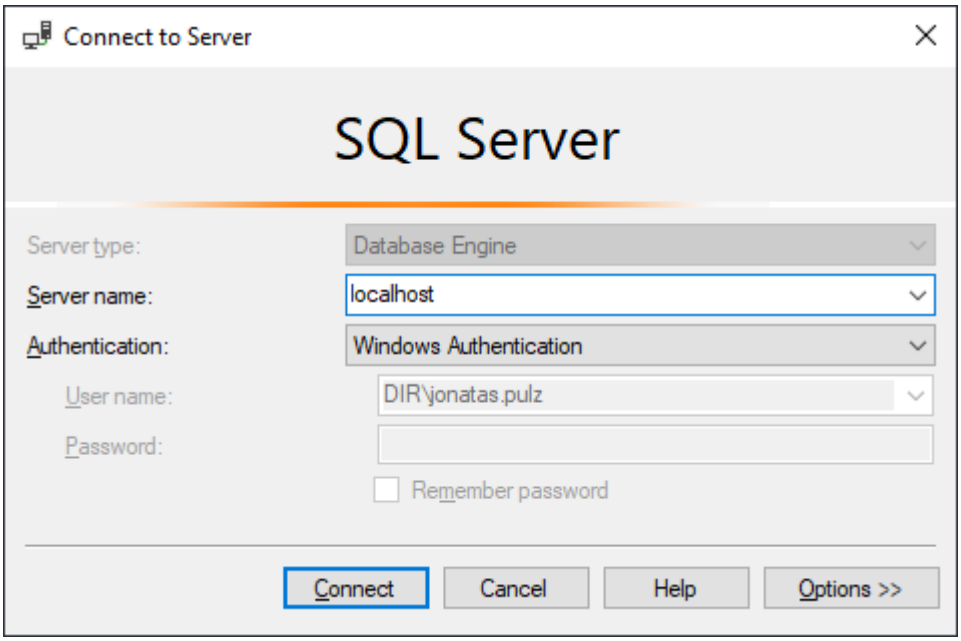


Fonte: Autor

É importante salientar que o ambiente utilizado para todo o processo foi um ambiente Windows 10. Foram realizados os seguintes passos para a criação do mapa que contém os IPF:

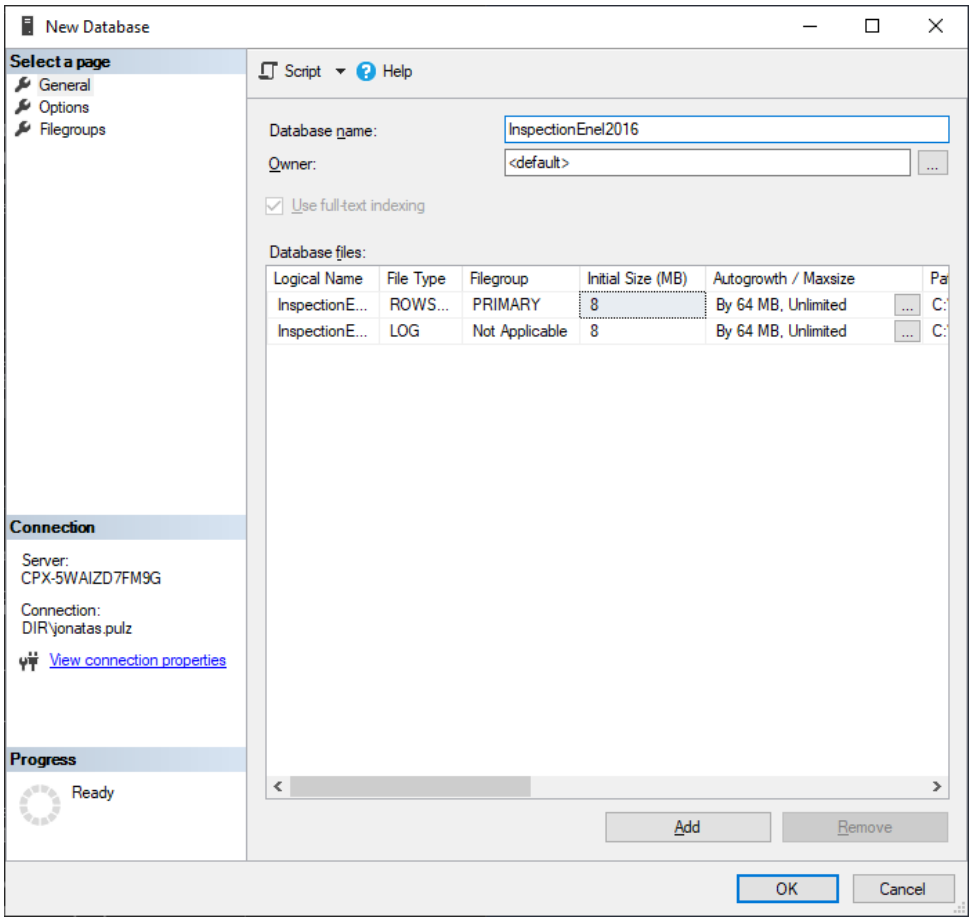
1. Foi instalado o SQL Server 2019 Developer (MICROSOFT, 2022a);
2. Foi instalado o SQL Server Management Studio (SSMS) (MICROSOFT, 2022b), versão 18.11;
3. Foi criado um banco de dados chamado “InspectionEnel2016” no SQL Server instalado que contém as informações de inspeções (coluna “hit_f” que indica fraude encontrada quando o valor é 1 e colunas que representem as coordenadas de longitude e latitude). Para isso, os seguintes passos foram necessários:
    a. Abrir o SSMS, executar a conexão conforme Figura 9 e depois criar o banco conforme Figura 10;

Figura 9 - Login no SQL Server



Fonte: Autor

Figura 10 - Criação do banco "InspectionEnel2016"

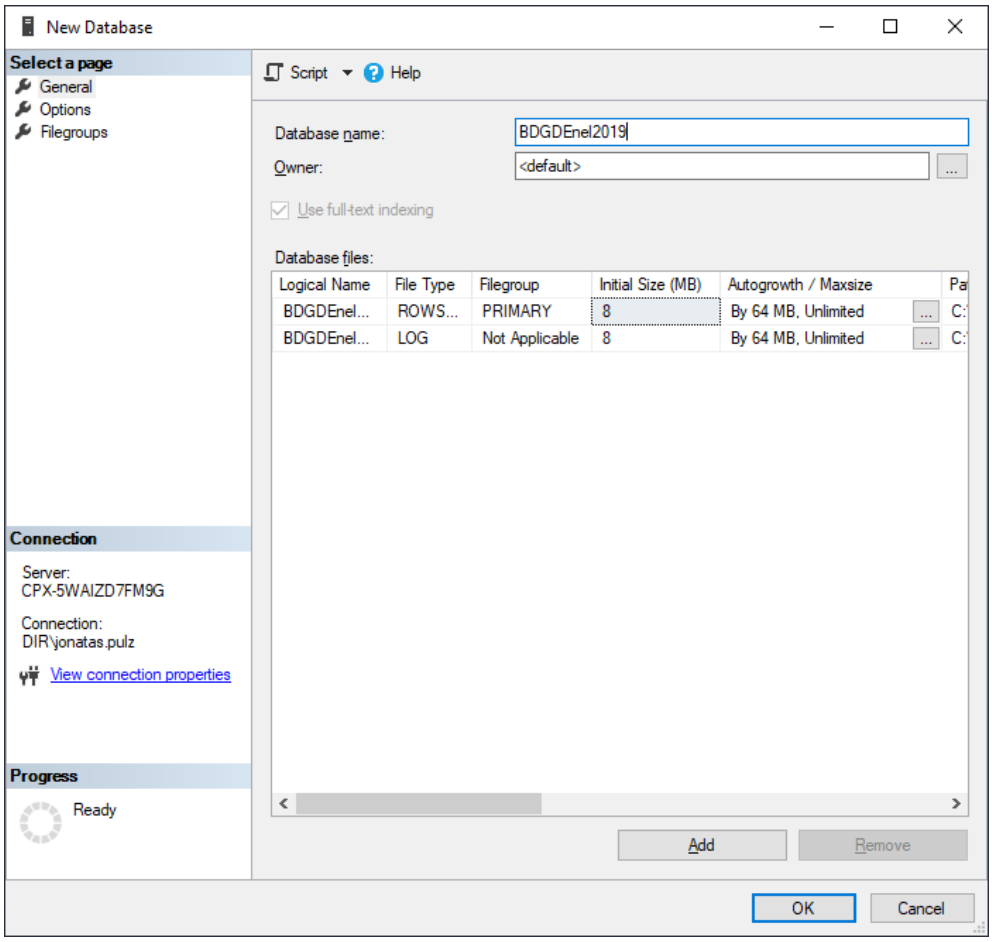


Fonte: Autor

b. No banco “InspectionEnel2016”, é necessário criar uma tabela chamada “InspectionLV”, populada por um arquivo em formato CSV ou Excel que contenha a coordenada e a informação da fraude encontrada em cada uma dessas coordenadas. Essa tabela contém os dados das inspeções de 2016 realizadas pela AES Eletropaulo. Nela, existem a coluna “hit_f” (quando igual a 1, indica a detecção de fraude na medição) e colunas de latitude e longitude que representam as coordenadas;

4. É necessário criar um banco de dados no SQL Server da BDGD em questão. Para isso, deve-se seguir os seguintes passos:
    a. Abrir o SSMS, executar a conexão conforme Figura 9 e depois criar o banco “BDGDEnel2019” conforme Figura 11;

Figura 11 - Criação do banco “BDGDEnel2019” no SQL Server



Fonte: Autor

b. Ter obtido a BDGD da Enel (antiga AES Eletropaulo) através da Lei de Acesso à Informação (Fala.BR, 2020), e entre os arquivos disponibilizados pela ANEEL, estarão os arquivos compactados “ELETROPAULO_390_2019-12-31_M10_20200130-0726.gdb.7z.00X”, em que ‘X’ é um número inteiro, deve-se descompactar esse arquivo através do software 7-Zip (7-Zip, 2022), o que irá gerar uma pasta chamada “ELETROPAULO_390_2019-12-31_M10_20200130-0726.gdb”; que contém toda a BDGD. Essa pasta precisa ter transformada em um banco SQL Server;

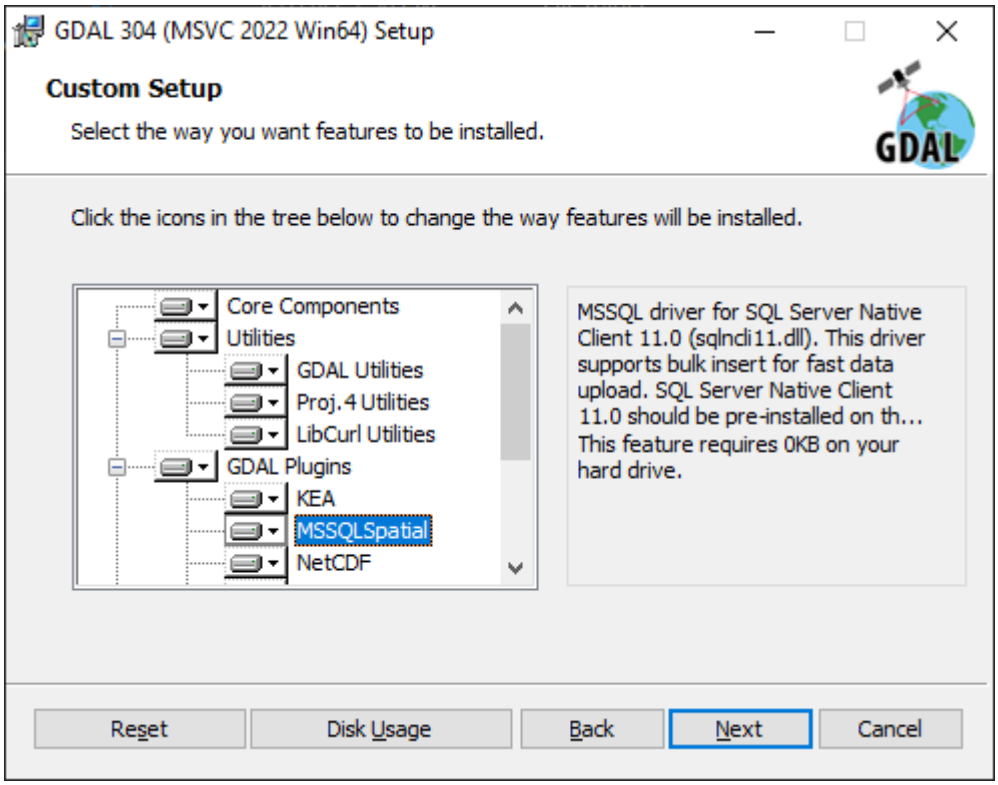
c. Para transformar a pasta descompactada da BDGD em um banco SQL Server, é necessário instalar o software GDAL (GDAL, 2022). A instalação precisa ser completa (incluindo o Plugin “MSSQLSpatial”), conforme Figura 12. Depois de instalado, deve-se configurar as seguintes variáveis de ambiente no Windows: GDAL_DATA:

"C:\Program Files\GDAL\gdal-data"; GDAL_DRIVER_PATH: "C:\Program Files\GDAL\gdalplugins" e PROJ_LIB "C:\Program Files\GDAL\projlib". Depois de configuradas as variáveis de ambientes, deve-se abrir um CMD e executar o seguinte comando a partir da pasta raiz onde está a pasta da BDGD:

“C:\Program Files\GDAL\ogr2ogr.exe” --config MSSQLSPATIAL_USE_BCP TRUE --config MSSQLSPATIAL_BCP_SIZE 60000 -f MSSQLSpatial "MSSQL:server=localhost;database=BDGDEnel2019;trusted_connection=yes;DRIVER={SQL Server Native Client 11.0};" "ELETROPAULO_390_2019-12-31_M10_20200130-0726.gdb" -gt 60000 -progress -unsetFid

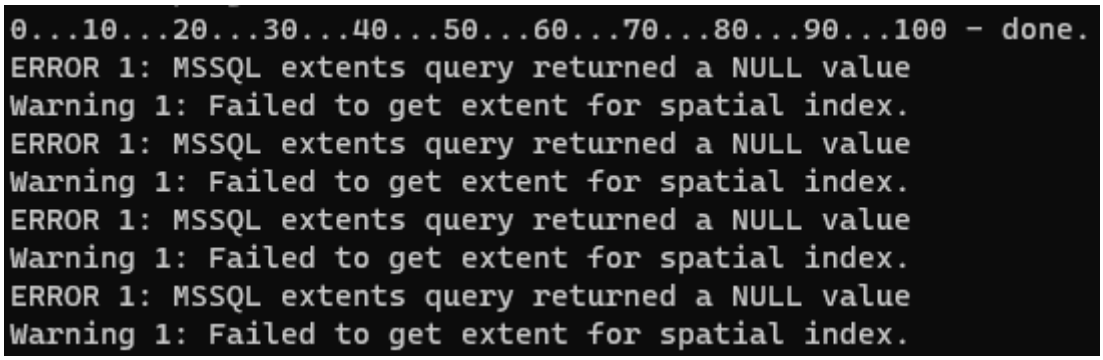
d. O processo irá apresentar o resultado no terminal CMD como na Figura 13. Os erros de criação de indexação espacial podem ser ignorados;

Figura 12 - Instalação do GDAL



Fonte: Autor

Figura 13 - Retorno do comando para criação da BDGD no SQL Server



```
0...10...20...30...40...50...60...70...80...90...100 - done.
ERROR 1: MSSQL extents query returned a NULL value
Warning 1: Failed to get extent for spatial index.
ERROR 1: MSSQL extents query returned a NULL value
Warning 1: Failed to get extent for spatial index.
ERROR 1: MSSQL extents query returned a NULL value
Warning 1: Failed to get extent for spatial index.
ERROR 1: MSSQL extents query returned a NULL value
Warning 1: Failed to get extent for spatial index.
```

Fonte: Autor

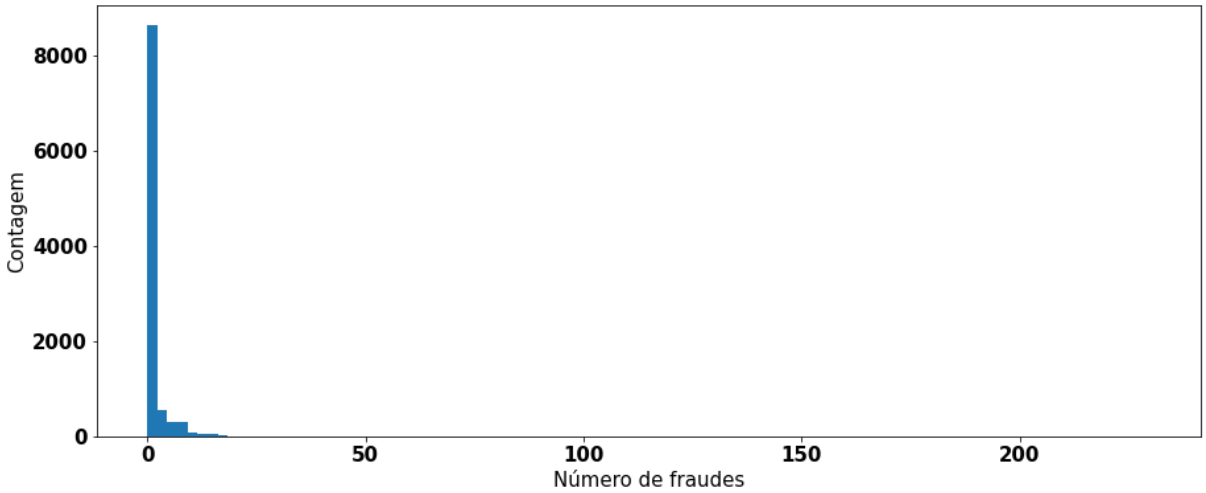
5. Então, é executado um Python Notebook que lê os dados das fraudes e suas coordenadas e realiza o *Kriging* para obtenção do IPF para cada unidade consumidora presente na BDGD. Esse Python Notebook se encontra no Apêndice D, transformado em formato Word através do software Pandoc (PANDOC, 2022). A execução do *notebook* cria o arquivo *shapefile* “ipf_kriging_500.shp”, contendo o código identificador de cada instalação e seu respectivo IPF.

A seguir, nesta seção, são apresentados os resultados da obtenção do IPF.

Com os dados de fraudes georreferenciados, foi feito o histograma em duas dimensões com a contagem do número de fraudes em cada caixa do histograma (um retângulo espacial), usando uma divisão em 100 partes na longitude e 100 partes na latitude, ou seja, 10.000 retângulos na área onde houve alguma fraude em 2016 no município de São Paulo. Como cada retângulo representa aproximadamente 850 metros, não há quebra de sigilo dos dados privados para se identificar unidades consumidoras fraudadoras através das figuras expostas nesse trabalho.

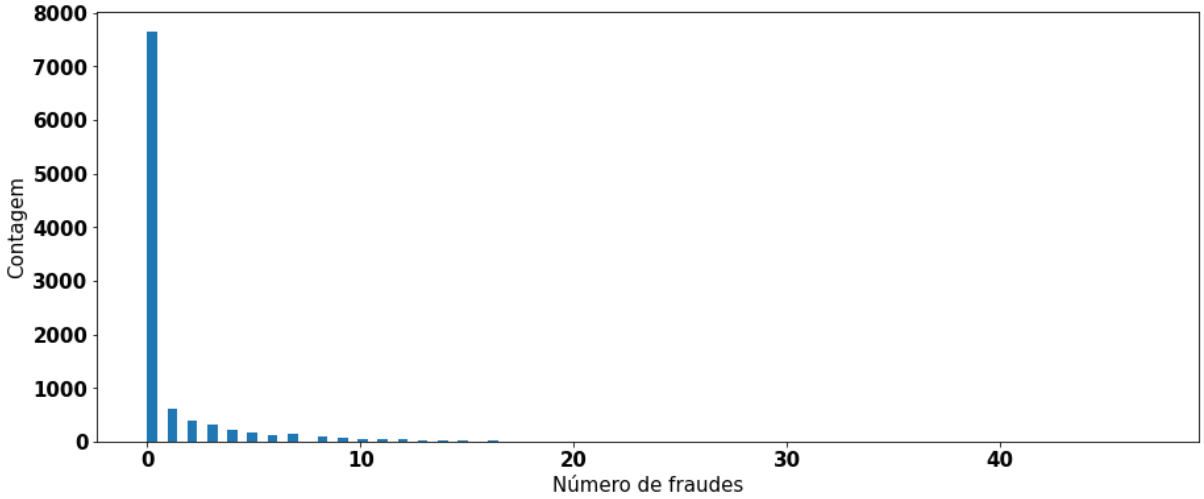
Nota-se que cada retângulo do histograma em duas dimensões possui o número de fraudes encontradas, mas há um valor extremo de 230 fraudes em um único retângulo, como mostrado na Figura 14. Esse valor foi removido e ao retângulo que o continha, foi atribuído o segundo maior valor de fraudes por retângulo de 47. O novo histograma de número de fraudes por retângulo é apresentado na Figura 15. A remoção desse valor extremo é necessária, pois o *Kriging* com esse valor extremo incluído apresenta inconsistências para a obtenção do IPF.

Figura 14 - Histograma do número de fraudes por retângulo



Fonte: Autor

Figura 15 - Histograma do número de fraudes por retângulo sem valor extremo

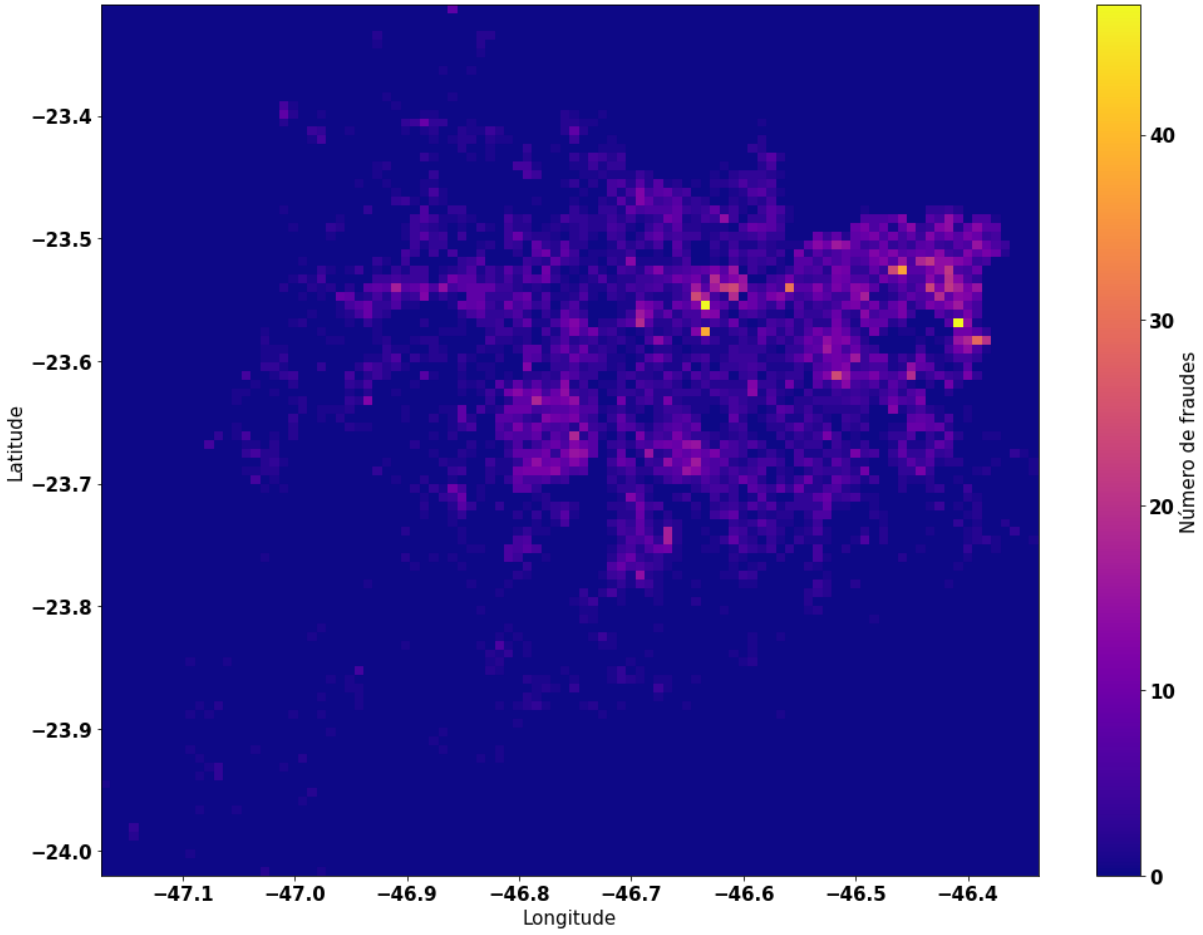


Fonte: Autor

O histograma de duas dimensões com a contagem do número de fraudes, com o valor extremo removido, é apresentado na Figura 16. O IPF, resultante do *Kriging* do histograma de duas dimensões com posterior normalização subtraindo o valor mínimo e dividindo pelo valor máximo com a subtração do valor mínimo, é mostrado na Figura 17. O *Kriging* foi realizado com 500 divisões em cada eixo, ou seja, 5 vezes maior que as divisões utilizadas no histograma, isso é feito para se obter uma melhor interpolação, como é mostrado mais a frente nesta seção. Esse resultado do *Kriging*, por conter mais de 250.000 linhas, não foi colocado como apêndice nesta

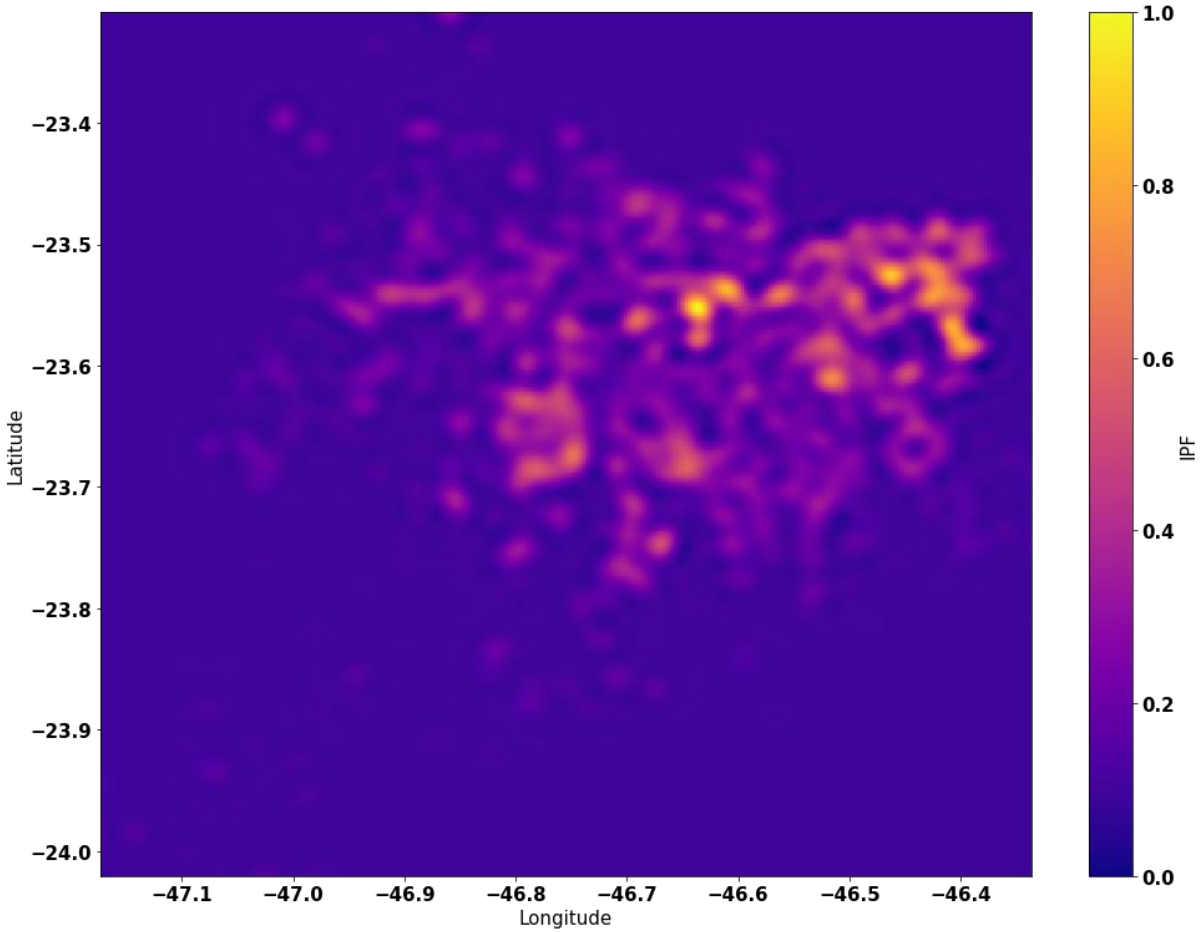
dissertação, mas pode ser obtido através do *link* disponível em (PULZ, 2022) – arquivo “kriging_500.shp”.

Figura 16 - Histograma em duas dimensões do número de fraudes



Fonte: Autor

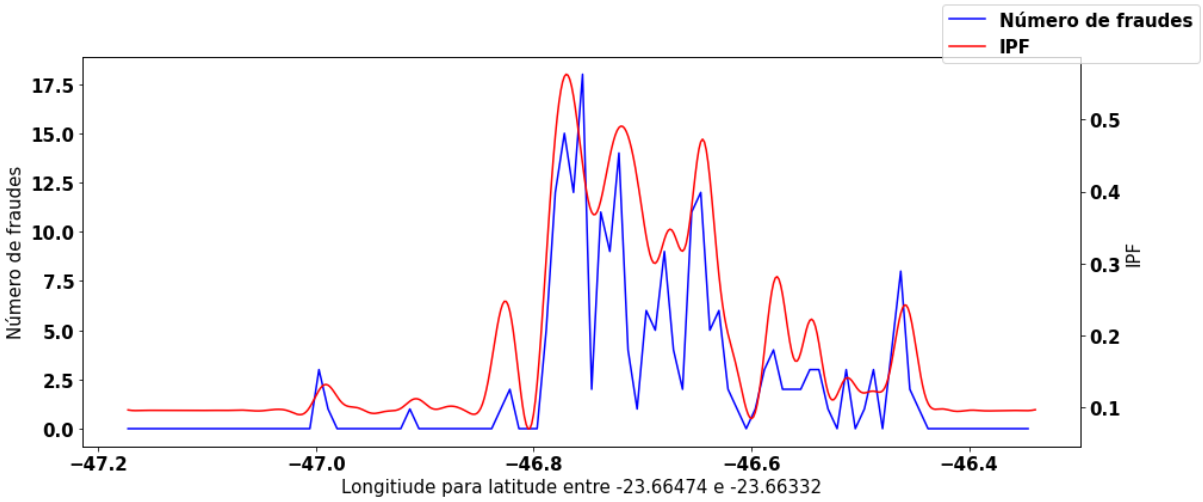
Figura 17 - *Kriging* normalizado do histograma de fraudes



Fonte: Autor

O *Kriging* utilizado foi o gaussiano e nota-se a clara interpolação realizada para a obtenção do IPF na Figura 18.

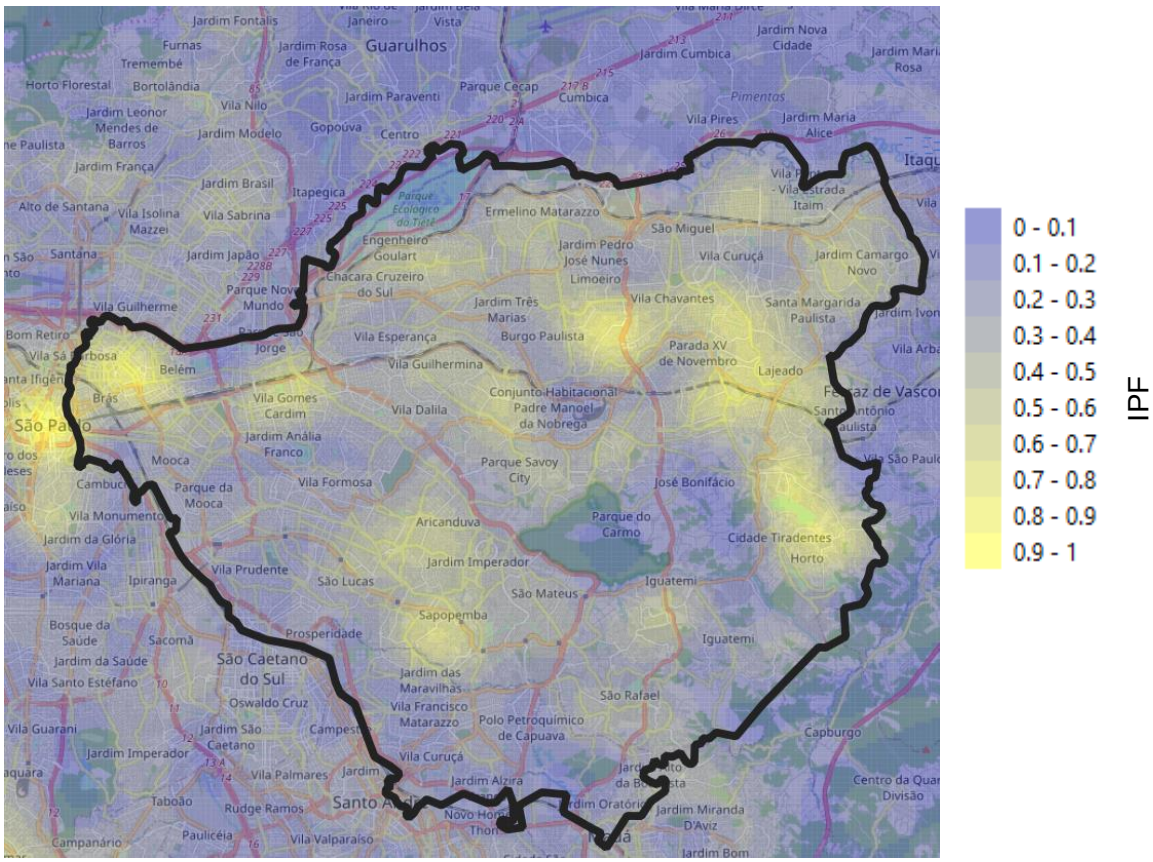
Figura 18 - Comparação entre número de fraudes e *Kriging*



Fonte: Autor

O mapa de IPF sobreposto ao mapa da região de interesse é mostrado na Figura 19, esse mapa foi criado através do software QGIS (QGIS, 2022) com os resultados da execução do *Notebook* Python. Nota-se que há uma clara distinção entre regiões com maior potencial de fraude e outras com baixo potencial de fraude, deixando de levar em consideração regiões não habitadas como o Parque do Carmo.

Figura 19 - IPF sobreposto ao mapa da região de interesse (linha preta delimita a área piloto)



Fonte: Autor

Agora que há um IPF para cada unidade consumidora, pode-se executar o fluxo de potência através do ProgGeoPerdas modificado para se obter os valores de perdas através da nova metodologia proposta. A seguir, na subseção subsequente, são apresentados os valores do fluxo de potência com a metodologia regulatória atualmente utilizada, quando esta dissertação foi elaborada, e com a nova metodologia proposta, detalhando as alterações feitas nos códigos do ProgGeoPerdas e scripts SQL.

5.3.1. Resultados de perdas com a metodologia regulatória e com a metodologia proposta

Primeiramente, é necessário determinar os alimentadores da distribuidora que estão na área de interesse. Isso pode ser feito através do software QGIS (QGIS, 2022). O passo-a-passo para isso se encontra no Apêndice E. Seguindo esse passo-a-passo, obtém-se que os seguintes alimentadores serão utilizados no estudo (em ordem

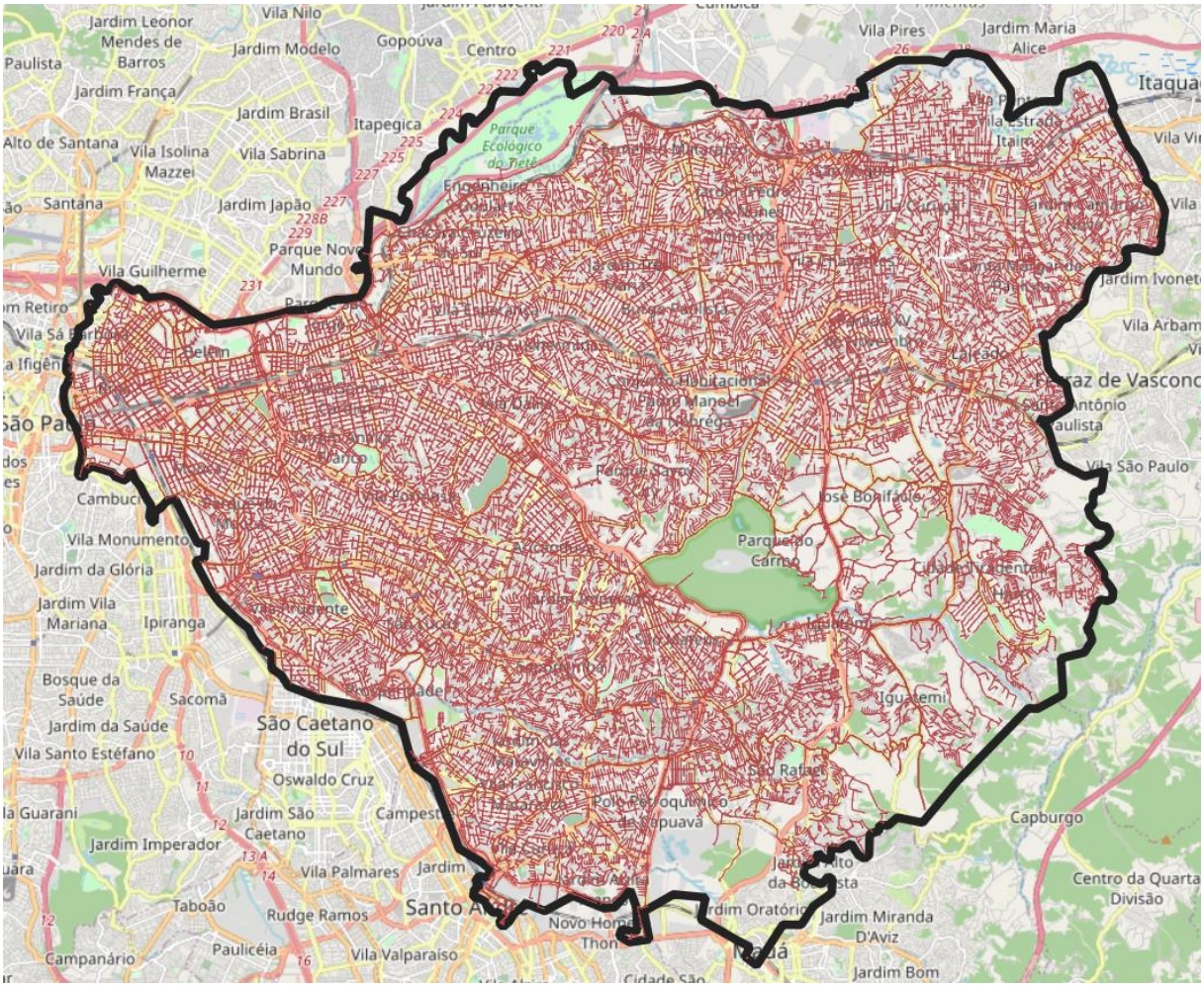
alfabética, total de 469), pois seus segmentos de média tensão estão inteiramente na área piloto:

BAI0100; BAI0101; BAI0102; BAI0103; BAI0104; BAI0105; BAI0106; BAI0107; BAI0108; BAI0109; BAI0110; BAI0111; BAI0112; BAR0100; BAR0101; BAR0102; BAR0103; BAR0104; BAR0105; BAR0106; BAR0107; BAR0108; BAR0109; BAR0110; BAR0111; BAR0112; BAR0113; BAR0114; BAR0115; BRA0100; BRA0101; BRA0102; BRA0103; BRA0104; BRA0105; BRA0106; BRA0107; BRA0108; BRA0109; BRA0110; BRA0111; BRA0112; BRA0113; BRA0114; BRA0115; CAM0001; CAM0003; CAM0012; CAM0016; CAM0107; CAP0101; CAP0103; CAT0100; CAT0101; CAT0102; CAT0103; CAT0104; CAT0105; CAT0106; CAT0107; CAT0108; CAT0109; CAT0110; CAT0111; CAT0112; CAT0113; CAT0114; CAT0115; CLA0100; CLA0101; CLA0102; CLA0103; CLA0106; CLA0107; CLA0108; CLA0109; CLA0110; CLA0111; CLA0112; CLA0113; CRA0100; CRA0101; CRA0102; CRA0103; CRA0104; CRA0105; CRA0106; CRA0107; CRA0108; CRA0109; CRA0110; CRA0111; CRA0112; CRA0113; CRA0114; CRA0115; DBAI-EQ-TR1; DBAI-EQ-TR2; DBAI-EQ-TR3; DBAI-EQ-TR4; DBAR-EQ-TR1; DBAR-EQ-TR2; DBRA-EQ-TR1; DBRA-EQ-TR2; DCAI-EQ-TR1; DCAI-EQ-TR2; DCAI-EQ-TR3; DCAM-EQ-TR1; DCAM-EQ-TR2; DCAM-EQ-TR5; DCAM-EQ-TR6; DCAM-EQ-TR7; DCAM-EQ-TR8; DCAM-EQ-TR9; DCAP-EQ-TR1; DCAP-EQ-TR2; DCAP-EQ-TR3; DCAP-EQ-TR4; DCAT-EQ-TR1; DCAT-EQ-TR2; DCLA-EQ-TR1; DCLA-EQ-TR2; DCRA-EQ-TR1; DCRA-EQ-TR2; DERM-EQ-TR1; DERM-EQ-TR2; DGCA-EQ-TR1; DGCA-EQ-TR2; DGCA-EQ-TR3; DGCA-EQ-TR4; DGCA-EQ-TR5; DGNA-EQ-TR1; DGNA-EQ-TR2; DGNA-EQ-TR3; DHIP-EQ-TR1; DHIP-EQ-TR2; DHIP-EQ-TR3; DHIP-EQ-TR4; DITN-EQ-TR1; DITN-EQ-TR2; DITN-EQ-TR3; DITN-EQ-TR4; DITN-EQ-TR5; DITR-EQ-TR1; DITR-EQ-TR2; DMAT-EQ-TR1; DMAT-EQ-TR2; DMAT-EQ-TR3; DMAT-EQ-TR4; DMAT-EQ-TR5; DMAU-EQ-TR1; DMAU-EQ-TR2; DMOO-EQ-TR1; DMOO-EQ-TR2; DMPA-EQ-TR1; DMPA-EQ-TR2; DMPA-EQ-TR3; DMPA-EQ-TR4; DMSA-EQ-TR1; DMSA-EQ-TR2; DMSA-EQ-TR3; DNAC-EQ-TR1; DNAC-EQ-TR2; DNAC-EQ-TR3; DNAC-EQ-TR4; DNAC-EQ-TR6; DORA-EQ-TR1; DORA-EQ-TR2; DPEC-EQ-TR1; DPEN-EQ-TR1; DPEN-EQ-TR2; DPSO-EQ-TR1; DPSO-EQ-TR2; DPSO-EQ-TR3; DPSO-EQ-TR4; DPSO-EQ-TR5; DPSO-EQ-TR6; DSLE-EQ-TR1; DSLE-EQ-TR2; DSND-EQ-TR1; DSND-EQ-TR2; DTIR-EQ-TR1; DTIR-EQ-TR2; DTIR-EQ-TR3; DTTI-EQ-TR1; DTTI-EQ-TR2; DUTI-EQ-TR1; DUTI-EQ-TR2; DUTI-EQ-TR3; DUTI-EQ-TR4; DVEM-EQ-TR1; DVEM-EQ-TR2; DVEM-EQ-TR3; DVEM-EQ-TR4; DVFO-EQ-TR1; DVFO-EQ-TR2; DVPR-EQ-TR1; DVPR-EQ-TR2; DVTA-EQ-TR1; DVTA-EQ-TR2; DVTA-EQ-TR4; DVTA-EQ-TR5; ERM0100; ERM0101; ERM0102; ERM0103; ERM0104; ERM0105; ERM0106; ERM0107; ERM0108; ERM0109; ERM0110; ERM0111; GCA0100; GCA0102; GCA0103; GCA0104; GCA0105; GCA0106; GNA0100; GNA0101; GNA0102; GNA0103; GNA0104; GNA0105; GNA0106; GNA0107; GNA0108; GNA0109; GNA0110; GNA0111; GNA0112; GNA0113; GNA0114; GNA0115; HIP0100; HIP0101; HIP0102; HIP0103; HIP0104; HIP0105; HIP0106; HIP0107; HIP0108; HIP0109; HIP0110; HIP0111; HIP0112; HIP0113; HIP0114; ITN0100; ITN0102; ITN0103; ITN0104; ITN0105; ITN0106; ITN0107; ITN0108; ITN0109; ITN0110; ITN0111; ITQ1303; ITR0100; ITR0101; ITR0102; ITR0103; ITR0104; ITR0105; ITR0106; ITR0107; ITR0108; ITR0109; ITR0110; ITR0111; ITR0112; ITR0113; ITR0114; ITR0115; MAT0100; MAT0101; MAT0102; MAT0103; MAT0104; MAT0105; MAT0106; MAT0107; MAT0108; MAT0109; MAT0110; MAT0111; MAT0112; MAT0113; MAT0114; MOO0101; MOO0102;

MOO0103; MOO0104; MOO0105; MOO0106; MOO0107; MOO0108; MPA0101; MPA0102; MPA0103; MPA0104; MPA0105; MPA0106; MPA0107; MPA0108; MPA0109; MPA0110; MPA0111; MPA0112; MRE0228; MRE0229; MRE0998; MSA0100; MSA0101; MSA0102; MSA0103; MSA0104; MSA0106; MSA0107; MSA0108; MSA0109; MSA0110; MSA0111; MSA0112; MSA0113; MSA0114; MSA0115; MSA0116; NAC0101; NAC0102; NAC0103; NAC0104; NAC0105; NAC0106; NAC0107; NAC0108; NAC0109; NAC0110; NAC0111; NAC0112; NAC0113; NAC0301; NAC0302; ORA0101; ORA0102; ORA0103; ORA0104; ORA0105; PEC0102; PEN0100; PEN0101; PEN0102; PEN0103; PEN0104; PEN0105; PEN0106; PEN0107; PEN0108; PEN0109; PEN0110; PEN0111; PEN0112; PEN0113; PEN0114; PEN0115; PSO0003; PSO0005; PSO0007; PSO0015; PSO0017; PSO0021; SLE0102; SLE0103; SND0100; SND0102; SND0115; TIR0100; TIR0101; TIR0102; TIR0103; TIR0104; TIR0105; TIR0106; TIR0107; TIR0108; TIR0109; TIR0110; TIR0111; TTI0100; TTI0101; TTI0102; TTI0103; TTI0104; TTI0105; TTI0106; TTI0107; TTI0108; TTI0109; TTI0110; TTI0111; TTI0112; TTI0113; TTI0114; TTI0115; UTI0100; UTI0102; UTI0103; UTI0104; UTI0105; UTI0107; UTI0108; UTI0110; VEM0100; VEM0101; VEM0102; VEM0103; VEM0104; VEM0105; VEM0106; VEM0107; VEM0108; VEM0109; VEM0110; VEM0111; VEM0112; VFO0100; VFO0101; VFO0102; VFO0103; VFO0104; VFO0105; VFO0106; VFO0107; VFO0108; VFO0109; VFO0110; VFO0111; VFO0112; VFO0113; VFO0114; VFO0115; VPR0100; VPR0101; VPR0102; VPR0103; VPR0104; VPR0105; VPR0106; VPR0107; VPR0108; VPR0109; VPR0110; VPR0111; VPR0112; VPR0113; VPR0114; VPR0115; VTA0100; VTA0102; VTA0103; VTA0104; VTA0105; VTA0106; VTA0107

Na Figura 20, são mostrados os segmentos de média tensão dos 469 alimentadores selecionados sobre o mapa da área piloto.

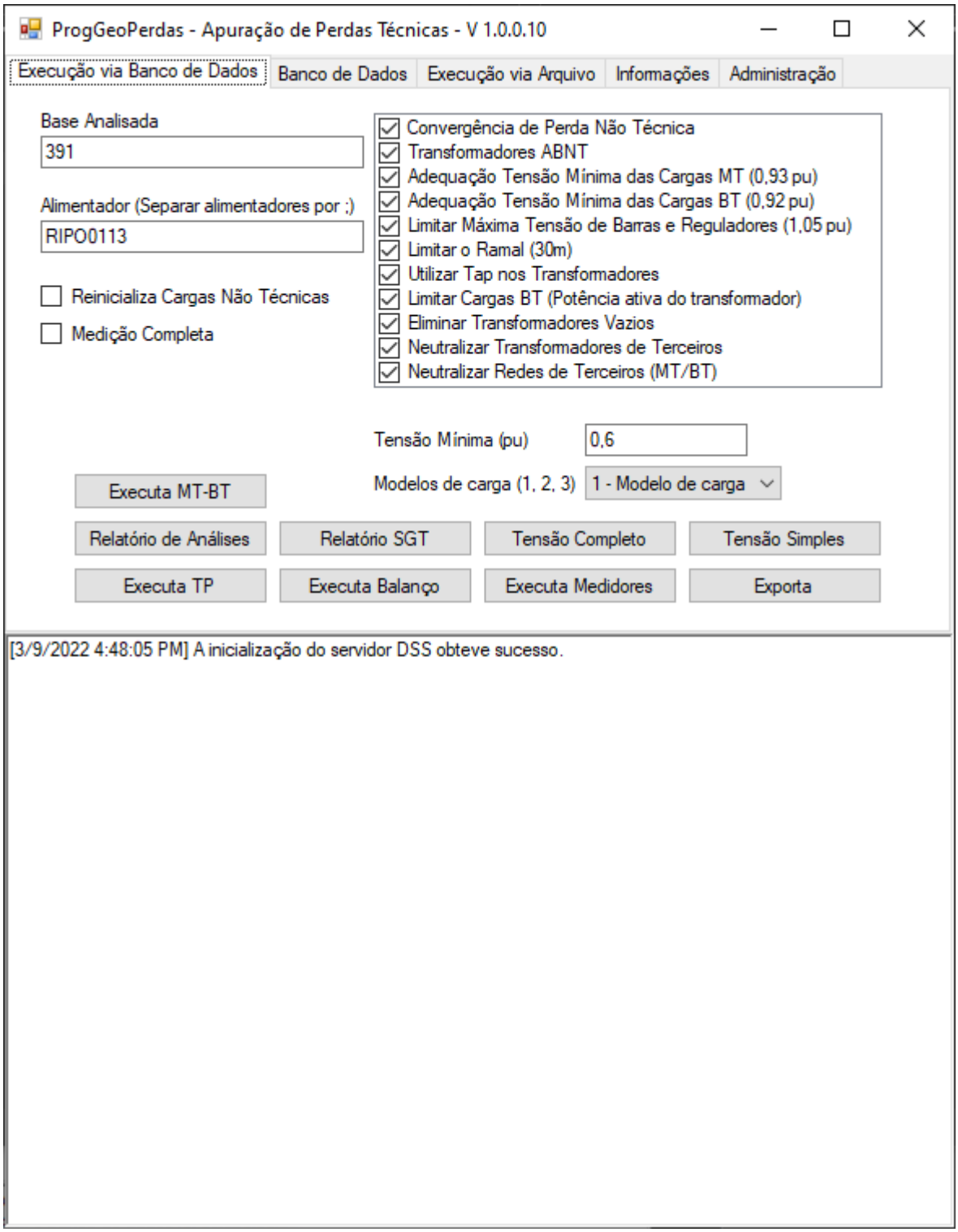
Figura 20 - Segmentos de média tensão (linha vermelha) na área piloto



Fonte: Autor

Para executar o ProgGeoPerdas, é necessário instalar o OpenDSS (EPRI, 2022), a versão utilizada no estudo foi a 7.6.5.91. A versão do ProgGeoPerdas utilizada é a 1.0.0.10. O arquivo para se executar o ProgGeoPerdas está disponível após se descompactar o arquivo disponível em (ANEEL, 2021a) no caminho: “Manual_BDGD_e_ANEXOS\Simulação Perdas Técnicas\ProgGeoPerdas10010\SETUP\ProgGeoPerdas.application”. Ao se iniciar o ProgGeoPerdas, sua janela é apresentada como mostrado na Figura 21. Se o ProgGeoPerdas reconhecer a instalação do OpenDSS, então é mostrado a mensagem “A inicialização do servidor DSS obteve sucesso.”, como mostrado na mesma figura.

Figura 21 - Janela inicial do ProgGeoPerdas 1.0.0.10



Fonte: Autor

Um passo importante é também realizar a alocação de curvas de carga a cada unidade consumidora ou ponto de iluminação pública. A BDGD obtida através da Lei de Acesso à Informação (Fala.BR, 2020) não está com as curvas corretamente

alocadas. Isso é notado no campo “tip_cc” na tabela “ucbt” através da Tabela 7, resultado da query

```
SELECT
        count(1) AS Quantidade
        ,tip_cc
FROM BDGDEnel2019.dbo.ucbt
GROUP BY
        tip_cc
ORDER BY
        tip_cc
```

Tabela 7 - Contagem de curvas de carga alocadas a unidades consumidoras de baixa tensão

(continua)

Quantidade	"tip_cc"
2.241.445	
10.356	COM-Tipo 1
30.741	COM-Tipo 10
11.434	COM-Tipo 11
13.174	COM-Tipo 12
8.904	COM-Tipo 13
28.244	COM-Tipo 14
31.694	COM-Tipo 15
37.314	COM-Tipo 2
14.599	COM-Tipo 3
21.847	COM-Tipo 4
15.260	COM-Tipo 5
33.371	COM-Tipo 6
22.224	COM-Tipo 7
22.564	COM-Tipo 8
22.258	COM-Tipo 9
2.101	IND-Tipo 1
506	IND-Tipo 10
968	IND-Tipo 11
280	IND-Tipo 12
573	IND-Tipo 13
1.615	IND-Tipo 14
1.805	IND-Tipo 15
1.218	IND-Tipo 2
3.546	IND-Tipo 3
1.093	IND-Tipo 4
1.077	IND-Tipo 5
1.268	IND-Tipo 6
3.395	IND-Tipo 7
1.578	IND-Tipo 8

Tabela 7 - Contagem de curvas de carga alocadas a unidades consumidoras de baixa tensão (conclusão)

Quantidade	"tip_cc"
2.111	IND-Tipo 9
1.391	IP-Tipo 1
43	IP-Tipo 2
2.833	IP-Tipo 3
1.837	MT-Tipo 1
33	MT-Tipo 13
1	MT-Tipo 15
3.209	MT-Tipo 2
274	MT-Tipo 3
12	MT-Tipo 4
396	MT-Tipo 5
235	MT-Tipo 6
606.286	RES-Tipo 1
582.093	RES-Tipo 10
179.363	RES-Tipo 11
419.216	RES-Tipo 12
242.046	RES-Tipo 13
548.609	RES-Tipo 14
353.374	RES-Tipo 15
102.649	RES-Tipo 2
260.618	RES-Tipo 3
787.910	RES-Tipo 4
105.267	RES-Tipo 5
541.082	RES-Tipo 6
165.300	RES-Tipo 7
668.258	RES-Tipo 8
57.623	RES-Tipo 9
19	RUR-Tipo 1
210	RUR-Tipo 10
13	RUR-Tipo 2
3	RUR-Tipo 3
11	RUR-Tipo 4
4	RUR-Tipo 5
26	RUR-Tipo 6
27	RUR-Tipo 7
9	RUR-Tipo 8
11	RUR-Tipo 9

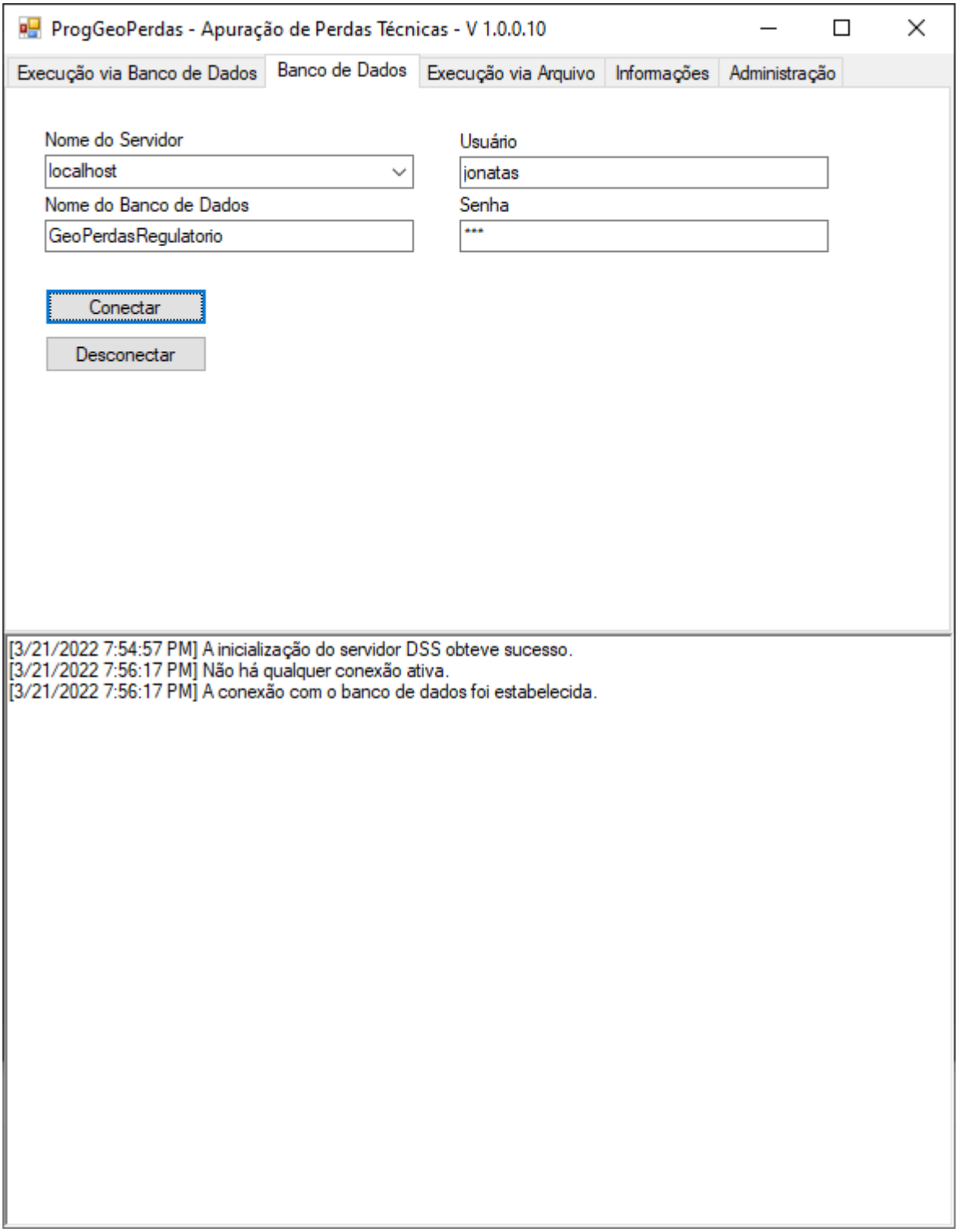
Fonte: Autor

As curvas de carga para cada tipo de carga são obtidas através de uma metodologia definida pela ANEEL, e não é o objetivo desta dissertação entrar em seus detalhes. As curvas resultantes dessa metodologia obtidas em 2019 estão disponíveis em (ANEEL, 2019). Os detalhes para alocação dessas curvas a cada um dos consumidores são apresentados no Apêndice F.

Para se conseguir criar o banco de dados que é utilizado pelo ProgGeoPerdas para o cálculo de perdas regulatórias, deve-se seguir os passos descritos no Apêndice G, este banco se chama “GeoPerdasRegulatorio”.

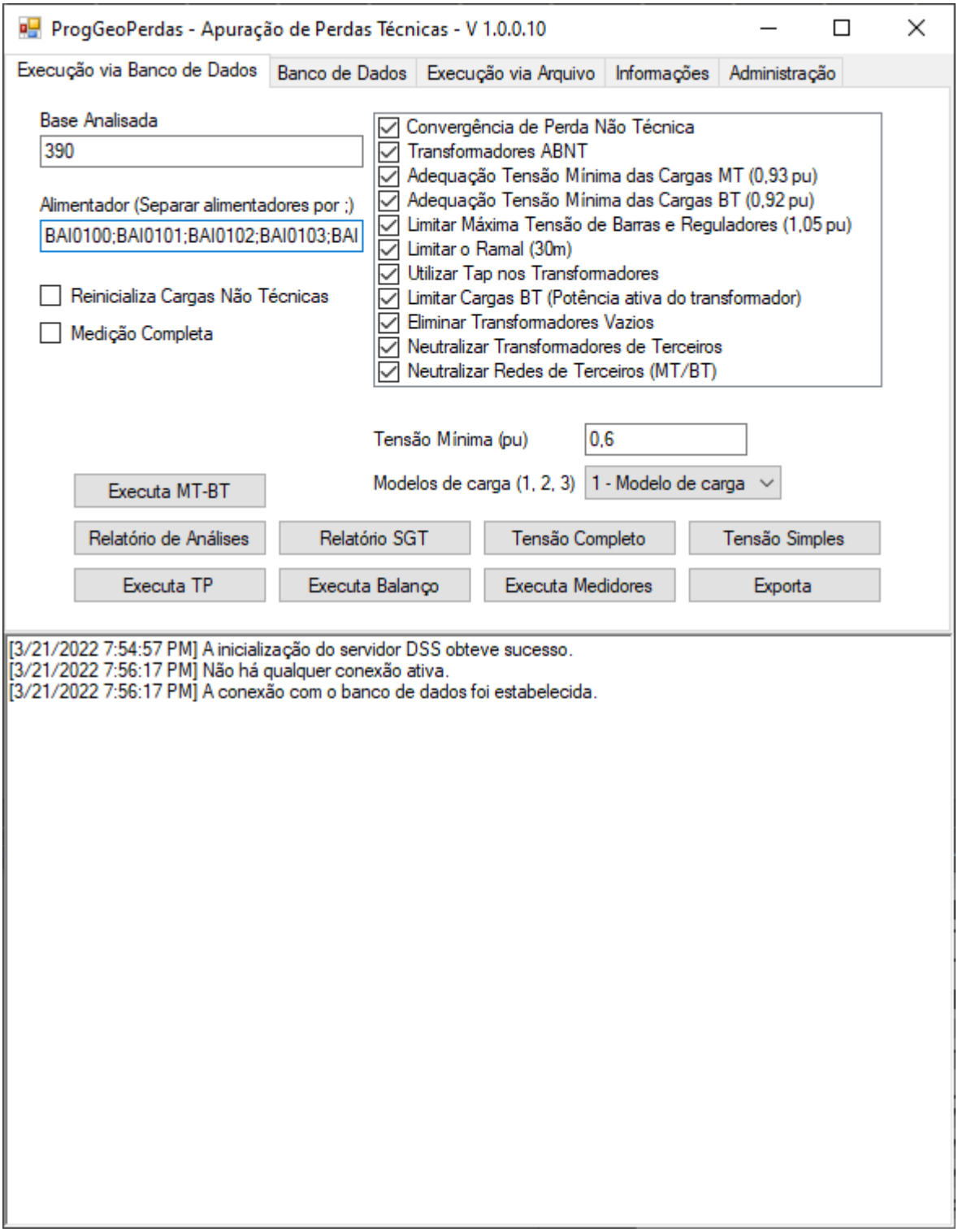
Dado que os passos do apêndice mencionado foram seguidos, o que inclui a criação de um usuário no servidor SQL Server, deve-se abrir o ProgGeoPerdas e o configurar conforme Figura 22 e Figura 23. Os alimentadores selecionados devem ser postos no campo “Alimentador”, separados por “;” sem espaços.

Figura 22 - Configuração do ProgGeoPerdas com relação ao banco de dados



Fonte: Autor

Figura 23 - Configuração do ProgGeoPerdas para cálculo de perdas nos alimentadores



Fonte: Autor

Desse ponto em diante, clica-se em “Executa MT-BT” e é aguardado a finalização do cálculo, que levou 17 horas em um sistema i7, com 32GB de RAM e SSD. O

ProgGeoPerdas não realiza, até o momento de elaboração desta dissertação, o cálculo de perdas para alimentadores que possuam malhas, e isso é apontado através do campo “CircAtipResultante” na tabela “390CircMT” do banco “GeoPerdasRegulatorio”, sendo o valor de “CircAtipResultante” igual a 0 para que o cálculo possa ser executado. Considerando esse filtro, o cálculo de perdas é realizado somente para 295 alimentadores dos 496 mencionados anteriormente, mas alguns tiveram problemas durante a execução, resultando em um cálculo de perdas realizado para 267 alimentadores dos 295. Os resultados foram inconsistentes, pois vários alimentadores que apresentaram PNT negativas ou PD muito altas. Uma amostra com os primeiros alimentadores em ordem alfabética é mostrada na Tabela 8.

Tabela 8 - Resultados inconsistentes de perdas com dados da BDGD (amostra de 8 alimentadores)

Alimentador	**Energia Injetada (kWh)**	**PT (kWh)**	**PNT (kWh)**	**PT (%)**	**PNT (%)**
BAI0102	5,650,570.94	109,584.06	-2,202,110.60	1.9	-39.0
BAI0103	0.00	0.00	-970,811.65	100.0	-214496607989.0
BAI0104	9,226,872.13	186,567.31	3,739,014.81	2.0	40.5
BAI0105	0.00	0.00	2,124,602.38	100.0	290460501358.4
BAI0107	9,634,028.84	300,600.03	2,041,838.18	3.1	21.2
BAI0108	17,920,902.74	195,332.11	5,145,077.21	1.1	28.7
BAI0110	5,210,234.23	99,050.51	4,953,985.46	1.9	95.1
BAI0111	0.00	0.00	6,303,010.72	100.0	858703258377.3

Fonte: Autor

A investigação desses resultados inconsistentes levou a analisar qual seria a relação, mês a mês, entre as energias das cargas MT somadas das energias das cargas BT sobre a energia medida no alimentador, como mostrado para os mesmos alimentadores na Tabela 9. Nota-se na tabela que vários alimentadores possuem razão de energia para com as cargas muito baixa, aproximadamente 70%, ou muito alta, próxima de 100% ou até mesmo maior que 100%, o que indica um erro na energia declarada no alimentador ou nas cargas conectadas a tal alimentador.

Tabela 9 - Relação de energia entre cargas e declaradas para os alimentadores com dados da BDGD (amostra de 8 alimentadores)

Alimentador	**Relação de energia das cargas para a energia declarada para o alimentador no mês (%)**											
	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**
BAI0102	115	126	107	101	98	111	109	116	108	112	112	114
BAI0103	94	111	110	99	98	107	104	111	106	106	118	115
BAI0104	70	86	81	81	78	78	84	83	80	81	88	81
BAI0105	75	88	84	77	77	84	73	86	46	114	169	87
BAI0107	77	89	85	87	82	85	86	86	86	91	89	88
BAI0108	81	92	83	84	83	88	88	90	87	78	90	86
BAI0110	75	92	83	83	79	84	83	84	88	88	91	90
BAI0111	72	83	81	78	76	80	80	82	82	83	80	86

Fonte: Autor

Conclui-se que há inconsistências nos dados da BDGD, provavelmente no valor dos pontos de acoplamento que levam cargas a se conectarem nos alimentadores incorretos. Isso é concluído, pois pela Lei de Acesso à Informação (Fala.BR, 2020), há um pedido de acesso ao banco de dados utilizado para o cálculo de perdas da Enel São Paulo (processo nº 48500.000249/2019-29) – dados se referem a 2017 – para a Revisão Tarifária Periódica (RTP), a 5ª RTP, e eles foram utilizados para comparação conforme Tabela 10 com relação aos valores declarados de energia para os alimentadores mês a mês. Nessa tabela, nota-se uma variação de energia mês a mês não maior que 5,7%, portanto a BDGD deve ter dados inconsistentes com relação às conexões das cargas nos alimentadores e por isso gera resultados inconsistentes de perdas obtidas através do uso do ProgGeoPerdas.

Tabela 10 - Variação de energia entre dados de energia da BDGD e dados de energia da 5ª RTP (amostra de 8 alimentadores)

Alimentador	**Variação entre energia dos alimentadores dos dados da BDGD para os dados da 5ª RTP (%)**											
	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**
BAI0102	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0103	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0104	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0105	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0107	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0108	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0110	5,7	-02	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3
BAI0111	5,7	-0,2	-2,5	5,4	2,0	-1,3	0,0	-1,3	-0,2	3.0	2.3	-1.3

Fonte: Autor

Portanto, descartou-se a utilização da BDGD para o cálculo de perdas, e utilizou-se o banco de dados da 5ª Revisão Tarifária Periódica, dados de 2017, que aparenta mais zelo em sua consistência, como pode ser visto pela relação entre as energias das cargas conectadas ao alimentador e a energia declarada no alimentador na Tabela 11, que apresenta valores que variam entre 84% e 93%. O passo-a-passo para construção do banco de dados no SQL Server para utilização do ProgGeoPerdas a partir desses dados da 5ª Revisão Tarifária Periódica encontra-se detalhado no Apêndice H. A preparação do banco de dados e o cálculo de perdas levaram 5 dias para serem executados em um sistema i7, com 32GB de RAM e SSD.

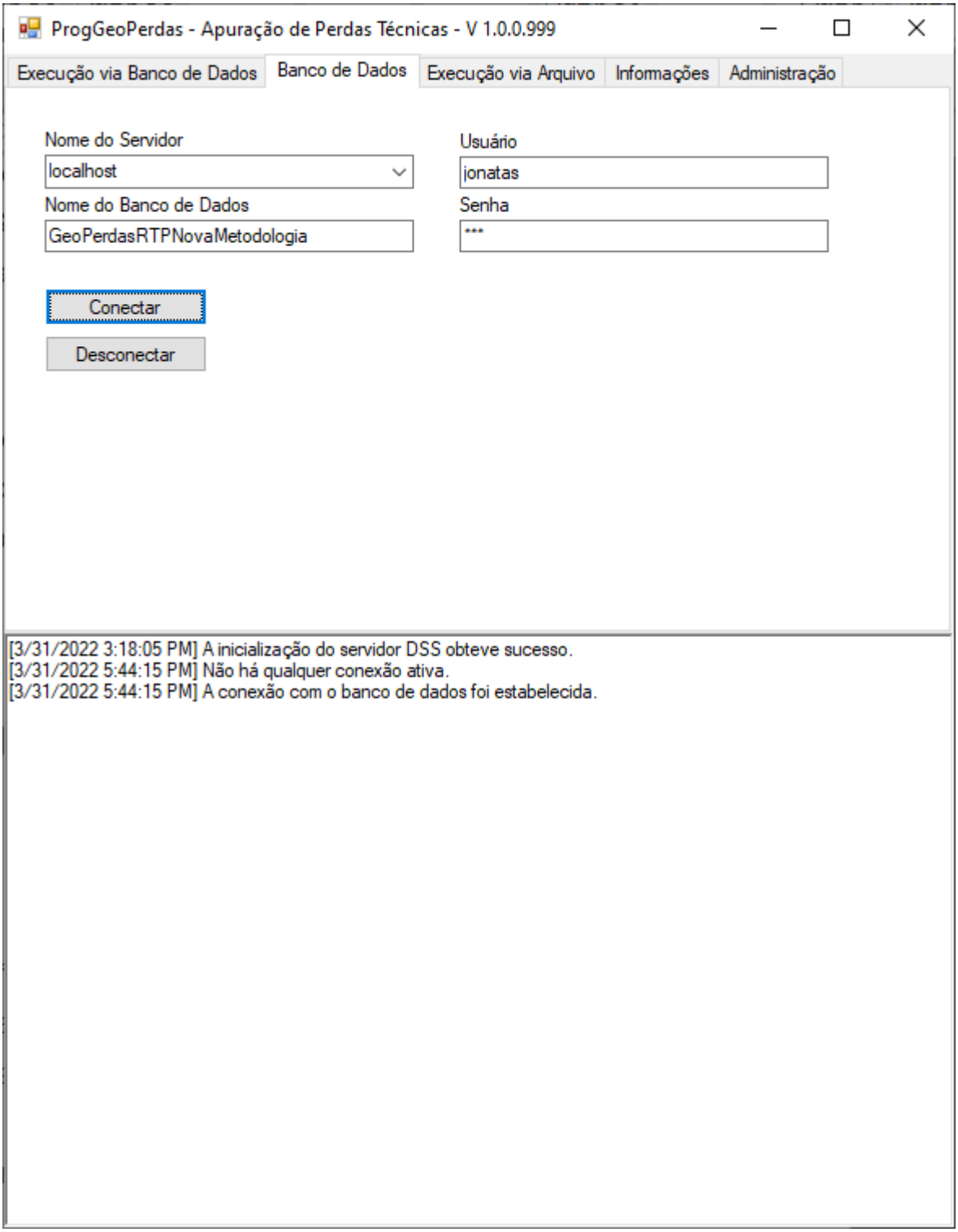
Tabela 11 - Relação de energia entre cargas e declaradas para os alimentadores com dados da 5ª RTP (amostra de 8 alimentadores)

Alimentador	**Relação de energia das cargas para a energia declarada para o alimentador no mês (%) (dados da 5ª RTP)**											
	Jan	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**
BAI0102	91	92	92	92	92	92	91	91	92	92	92	92
BAI0103	86	86	86	84	84	86	84	86	86	84	86	84
BAI0104	86	89	88	89	86	89	86	88	86	89	88	89
BAI0105	87	87	87	87	86	84	88	85	84	85	84	85
BAI0107	88	89	89	89	88	89	88	89	88	89	89	89
BAI0108	92	92	90	90	92	90	92	92	92	93	92	93
BAI0110	87	88	87	88	88	85	87	87	87	88	87	87
BAI0111	86	86	86	84	84	86	84	86	86	84	86	86

Fonte: Autor

Para realizar o cálculo de perdas com o ProgGeoPerdas através da metodologia proposta nesta dissertação, é necessário seguir os passos presentes no Apêndice I, que mostra como construir o banco de dados “GeoPerdasRTPNovaMetodologia”, e também como deve ser modificado o próprio ProgGeoPerdas, através do código fonte disponibilizado pela ANEEL em (ANEEL, 2021a). Com o novo ProgGeoPerdas e com o banco de dados “GeoPerdasRTPNovaMetodologia”, obtém-se o cálculo de perdas com a metodologia proposta nesta dissertação. O novo executável do ProgGeoPerdas deve ser executado e configurado na aba “Banco de Dados” conforme Figura 24, alterando-se apenas o nome de usuário e senha, as configurações da aba “Execução via Banco de Dados” devem ser as mesmas da Figura 23. A preparação do banco de dados e o cálculo de perdas levaram 5 dias para serem executados em um sistema i7, com 32GB de RAM e SSD.

Figura 24 - Configuração de Banco de Dados do novo ProgGeoPerdas



Fonte: Autor

Para a comparação com o cálculo de perdas através da metodologia regulatória (atual no momento de elaboração desta dissertação) é necessário executar o seguinte código SQL:

WITH RTPRegulatorio AS

```
(
        SELECT
                result.CodAlim
                ,result.[EnergiaInj_kWh]
                ,result.[PerdaEnergiaTecnica_kWh]
                ,nt.EnerNT_MWh*1000 as PerdaEnergiaNT_kWh
                ,(result.[PerdaEnergiaTecnica_kWh])/result.[EnergiaInj_kWh]*100
AS PerdaEnergiaTecnica_EnergiaInj_per
                ,(nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100               AS
PerdaEnergiaNT_EnergiaInj_per
                ,(result.[PerdaEnergiaTecnica_kWh] +
                        nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100         AS
PerdaEnergiaTotal_EnergiaInj_per
        FROM
                [GeoPerdasRTPRegulatorio].[dbo].[390AuxResultadoAno] result
        LEFT JOIN
                (
                        SELECT
                                CodAlimAtrib
                                ,sum(EnerNT) EnerNT_MWh
                        FROM
                                (SELECT
                                        CodAlimAtrib
                                        ,sum([EnerMedid01_MWh]
                                                +[EnerMedid02_MWh]
                                                +[EnerMedid03_MWh]
                                                +[EnerMedid04_MWh]
                                                +[EnerMedid05_MWh]
                                                +[EnerMedid06_MWh]
                                                +[EnerMedid07_MWh]
                                                +[EnerMedid08_MWh]
                                                +[EnerMedid09_MWh]
                                                +[EnerMedid10_MWh]
                                                +[EnerMedid11_MWh]
                                                +[EnerMedid12_MWh]
                                        ) EnerNT
                                FROM

        [GeoPerdasRTPRegulatorio].[dbo].[390AuxCargaBTNTDem]
                                WHERE
                                        CodAlimAtrib IS NOT NULL
                                GROUP BY
                                        CodAlimAtrib
                                UNION ALL
                                SELECT
                                        [CodAlimAtrib]
                                        ,sum([EnerMedid01_MWh]
                                                +[EnerMedid02_MWh]
                                                +[EnerMedid03_MWh]
                                                +[EnerMedid04_MWh]
```

```
                                        +[EnerMedid05_MWh]
                                        +[EnerMedid06_MWh]
                                        +[EnerMedid07_MWh]
                                        +[EnerMedid08_MWh]
                                        +[EnerMedid09_MWh]
                                        +[EnerMedid10_MWh]
                                        +[EnerMedid11_MWh]
                                        +[EnerMedid12_MWh]
                                ) EnerNT
                        FROM

        [GeoPerdasRTPRegulatorio].[dbo].[390AuxCargaMTNTDem]
                        WHERE
                                CodAlimAtrib IS NOT NULL
                        GROUP BY
                                CodAlimAtrib
                        ) nt
                GROUP BY
                        CodAlimAtrib
            ) nt
        ON result.CodAlim = nt.CodAlimAtrib
        WHERE
            nt.EnerNT_MWh >= 0
)
,RTPNovaMetodologia AS
(
        SELECT
            result.CodAlim
            ,result.[EnergiaInj_kWh]
            ,result.[PerdaEnergiaTecnica_kWh]
            ,nt.EnerNT_MWh*1000 as PerdaEnergiaNT_kWh
            ,(result.[PerdaEnergiaTecnica_kWh])/result.[EnergiaInj_kWh]*100
AS PerdaEnergiaTecnica_EnergiaInj_per
            ,(nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100              AS
PerdaEnergiaNT_EnergiaInj_per
            ,(result.[PerdaEnergiaTecnica_kWh] +
                nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100        AS
PerdaEnergiaTotal_EnergiaInj_per
        FROM
            [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxResultadoAno]
result
        LEFT JOIN
            (
                SELECT
                    CodAlimAtrib
                    ,sum(EnerNT) EnerNT_MWh
                FROM
                    (SELECT
                        CodAlimAtrib
                        ,sum([EnerMedid01_MWh]
```

```
                                        +[EnerMedid02_MWh]
                                        +[EnerMedid03_MWh]
                                        +[EnerMedid04_MWh]
                                        +[EnerMedid05_MWh]
                                        +[EnerMedid06_MWh]
                                        +[EnerMedid07_MWh]
                                        +[EnerMedid08_MWh]
                                        +[EnerMedid09_MWh]
                                        +[EnerMedid10_MWh]
                                        +[EnerMedid11_MWh]
                                        +[EnerMedid12_MWh]
                                ) EnerNT
                        FROM

        [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxCargaBTNTDem]
                        WHERE
                                CodAlimAtrib IS NOT NULL
                        GROUP BY
                                CodAlimAtrib
                        UNION ALL
                        SELECT
                                [CodAlimAtrib]
                                ,sum([EnerMedid01_MWh]
                                        +[EnerMedid02_MWh]
                                        +[EnerMedid03_MWh]
                                        +[EnerMedid04_MWh]
                                        +[EnerMedid05_MWh]
                                        +[EnerMedid06_MWh]
                                        +[EnerMedid07_MWh]
                                        +[EnerMedid08_MWh]
                                        +[EnerMedid09_MWh]
                                        +[EnerMedid10_MWh]
                                        +[EnerMedid11_MWh]
                                        +[EnerMedid12_MWh]
                                ) EnerNT
                        FROM

        [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxCargaMTNTDem]
                        WHERE
                                CodAlimAtrib IS NOT NULL
                        GROUP BY
                                CodAlimAtrib
                        ) nt
                GROUP BY
                        CodAlimAtrib
        ) nt
        ON result.CodAlim = nt.CodAlimAtrib
)
SELECT
        RTPRegulatorio.*
```

```
        ,RTPNovaMetodologia.EnergiaInj_kWh AS EnergiaInj_kWh_N
        ,RTPNovaMetodologia.PerdaEnergiaTecnica_kWh AS
PerdaEnergiaTecnica_kWh_N
        ,RTPNovaMetodologia.PerdaEnergiaNT_kWh AS
PerdaEnergiaNT_kWh_N
        ,RTPNovaMetodologia.PerdaEnergiaTecnica_EnergiaInj_per AS
PerdaEnergiaTecnica_EnergiaInj_per_N
        ,RTPNovaMetodologia.PerdaEnergiaNT_EnergiaInj_per AS
PerdaEnergiaNT_EnergiaInj_per_N
        ,RTPNovaMetodologia.PerdaEnergiaTotal_EnergiaInj_per AS
PerdaEnergiaTotal_EnergiaInj_per_N
FROM
        RTPRegulatorio
INNER JOIN
        RTPNovaMetodologia
ON
        RTPNovaMetodologia.CodAlim = RTPRegulatorio.CodAlim
ORDER BY
        RTPRegulatorio.CodAlim
```

O resultado da comparação é mostrado na Tabela 12 para cada um dos alimentadores que tiveram um resultado válido (“CircAtipResultante” igual a 0 e valores de PNT maiores que 0).

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
BAI0102	19.437.093	3,83	5,48	9,31	19.437.268	3,83	5,46	9,29	-0,07	-0,49	-0,32
BAI0103	15.028.024	3,83	13,77	17,61	15.028.120	3,85	13,73	17,58	0,37	-0,30	-0,16
BAI0104	18.464.246	5,53	7,88	13,41	18.453.424	5,54	7,87	13,41	0,29	-0,10	0,06
BAI0105	17.571.223	5,26	14,51	19,77	17.569.660	5,28	14,44	19,72	0,40	-0,45	-0,22
BAI0106	22.551.578	5,26	9,62	14,88	22.551.883	5,28	9,56	14,83	0,24	-0,60	-0,30
BAI0107	16.354.383	4,70	10,94	15,65	16.354.301	4,73	10,95	15,68	0,50	0,04	0,18
BAI0108	36.316.029	3,58	4,20	7,77	36.315.322	3,58	4,20	7,79	0,22	0,10	0,16
BAI0109	10.467.810	5,95	15,14	21,10	10.467.743	5,99	15,15	21,14	0,59	0,05	0,20
BAI0110	33.160.705	6,48	13,49	19,97	33.160.902	6,50	13,44	19,95	0,35	-0,33	-0,11
BAI0111	28.843.475	7,37	8,40	15,77	28.849.549	7,38	8,38	15,77	0,20	-0,22	-0,02
BAI0112	6.846.083	4,62	9,11	13,73	6.845.830	4,67	9,17	13,84	1,11	0,66	0,81
BAR0102	40.254.006	4,09	14,47	18,57	40.254.038	4,09	14,47	18,56	0,02	-0,06	-0,04
BAR0103	49.362.345	5,90	14,37	20,27	49.362.479	5,86	14,34	20,21	-0,63	-0,18	-0,31
BAR0104	21.175.574	4,18	15,43	19,62	21.175.572	4,20	15,43	19,62	0,33	-0,06	0,03
BAR0105	11.116.560	3,69	15,28	18,97	11.116.585	3,71	15,26	18,97	0,52	-0,13	-0,01

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
BAR0106	17.063.172	4,36	15,58	19,94	17.063.072	4,40	15,48	19,89	0,98	-0,61	-0,26
BAR0107	19.452.489	4,85	16,62	21,46	19.452.341	4,87	16,64	21,51	0,43	0,16	0,22
BAR0108	19.530.049	5,05	16,71	21,76	19.530.002	5,06	16,71	21,77	0,19	0,02	0,06
BAR0109	27.723.064	6,00	16,98	22,98	27.723.537	6,06	17,05	23,10	0,91	0,42	0,55
BAR0111	29.706.280	6,38	17,88	24,25	29.706.292	6,47	18,19	24,66	1,38	1,76	1,66
BAR0112	31.448.050	5,33	16,77	22,10	31.448.112	5,35	16,72	22,07	0,31	-0,25	-0,11
BAR0113	41.269.821	8,07	16,10	24,17	41.269.318	8,13	16,20	24,33	0,74	0,62	0,66
BAR0114	23.569.547	5,32	17,67	23,00	23.569.451	5,34	17,69	23,03	0,27	0,11	0,15
BAR0115	44.288.057	6,33	17,35	23,68	44.289.386	6,32	17,45	23,77	-0,11	0,58	0,39
BRA0102	10.631.667	2,19	1,71	3,90	10.631.604	2,19	1,70	3,89	-0,12	-0,50	-0,29
BRA0103	3.385.621	3,24	7,78	11,02	3.385.928	3,26	7,71	10,97	0,54	-0,83	-0,43
BRA0105	7.188.817	4,91	11,16	16,07	7.188.922	4,90	11,11	16,01	-0,30	-0,42	-0,38
BRA0106	28.699.722	2,33	1,82	4,15	28.699.740	2,34	1,82	4,16	0,36	0,45	0,40
BRA0107	970.386	3,25	8,90	12,15	969.294	3,27	8,83	12,10	0,60	-0,82	-0,44
BRA0108	9.264.583	3,58	2,87	6,45	9.264.512	3,58	2,86	6,44	0,01	-0,22	-0,09
BRA0109	10.708.469	3,09	3,06	6,15	10.708.432	3,09	3,07	6,16	0,11	0,14	0,12
BRA0110	2.305.127	4,73	7,35	12,08	2.305.313	4,72	7,32	12,05	-0,25	-0,33	-0,30
BRA0111	33.470.597	4,85	5,20	10,05	33.470.718	4,85	5,19	10,04	0,06	-0,20	-0,08
BRA0112	8.956.807	3,75	7,77	11,52	8.956.846	3,75	7,75	11,50	0,16	-0,27	-0,13
BRA0113	24.026.109	2,19	5,74	7,93	24.026.130	2,20	5,74	7,94	0,53	0,00	0,15
BRA0114	15.870.934	2,87	5,27	8,14	15.870.935	2,87	5,26	8,13	0,02	-0,19	-0,12
BRA0115	37.288.290	4,49	5,70	10,19	37.288.425	4,49	5,67	10,16	-0,09	-0,37	-0,25
CAM0003	15.051.591	5,23	24,67	29,90	15.051.677	5,25	24,76	30,02	0,39	0,38	0,38
CAM0012	3.562.321	4,07	21,08	25,15	3.562.013	4,28	21,13	25,41	5,29	0,23	1,05
CAP0103	11.910.061	4,32	9,34	13,66	11.909.814	4,31	9,38	13,69	-0,27	0,50	0,26
CAT0102	3.894.239	0,53	3,42	3,96	3.894.236	0,53	3,42	3,96	0,12	-0,01	0,00
CAT0103	26.244.448	4,37	3,14	7,51	26.244.730	4,37	3,12	7,49	-0,06	-0,46	-0,22
CAT0104	8.057.224	3,54	8,20	11,74	8.057.427	3,55	8,17	11,71	0,06	-0,39	-0,25
CAT0105	25.182.411	2,28	0,18	2,45	25.182.386	2,28	0,17	2,45	0,00	-1,69	-0,12
CAT0106	26.755.853	3,31	8,04	11,36	26.755.940	3,30	8,02	11,33	-0,23	-0,25	-0,24
CAT0107	20.882.936	5,03	11,28	16,31	20.883.265	5,04	11,21	16,25	0,15	-0,63	-0,39
CAT0108	13.875.552	3,19	7,79	10,98	13.875.255	3,21	7,81	11,02	0,51	0,32	0,37
CAT0110	6.432.319	4,00	2,90	6,90	6.432.251	4,00	2,91	6,91	0,06	0,05	0,05
CAT0111	15.768.511	3,38	7,25	10,63	15.768.506	3,39	7,23	10,62	0,03	-0,16	-0,10
CAT0113	17.986.207	3,99	3,04	7,03	17.985.891	3,97	3,00	6,97	-0,54	-1,42	-0,92
CAT0114	22.163.298	3,13	6,62	9,75	22.163.353	3,14	6,60	9,74	0,29	-0,27	-0,09
CLA0102	37.508.642	4,61	5,83	10,43	37.508.743	4,62	5,81	10,43	0,21	-0,23	-0,03

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
CLA0103	24.401.686	6,06	7,99	14,05	24.400.948	6,05	7,90	13,94	-0,18	-1,19	-0,76
CLA0106	14.082.798	3,47	4,64	8,11	14.082.870	3,45	4,63	8,08	-0,34	-0,29	-0,31
CLA0107	24.178.747	6,61	8,61	15,23	24.178.864	6,69	8,58	15,27	1,14	-0,32	0,32
CLA0108	54.622.799	7,70	8,65	16,35	54.622.738	7,73	8,65	16,37	0,32	0,00	0,15
CLA0109	53.892.234	7,24	7,78	15,03	53.893.019	7,24	7,71	14,95	-0,11	-0,98	-0,56
CLA0110	45.057.456	3,83	6,50	10,33	45.057.672	3,85	6,48	10,32	0,41	-0,33	-0,05
CLA0111	13.654.191	6,61	10,83	17,44	13.653.645	6,69	10,92	17,60	1,08	0,81	0,92
CLA0112	42.829.841	6,58	7,50	14,08	42.830.014	6,58	7,47	14,05	0,00	-0,36	-0,19
CLA0113	19.583.991	4,76	6,55	11,31	19.584.798	4,77	6,46	11,23	0,08	-1,37	-0,76
CRA0102	23.490.381	4,62	10,87	15,49	23.490.503	4,66	10,84	15,50	0,85	-0,29	0,05
CRA0103	37.345.579	5,11	12,88	17,99	37.345.660	5,15	12,85	18,00	0,69	-0,23	0,03
CRA0104	25.370.981	6,79	13,45	20,24	25.371.469	6,79	13,34	20,13	0,07	-0,82	-0,53
CRA0105	25.422.459	5,71	11,96	17,68	25.422.361	5,72	11,88	17,60	0,13	-0,68	-0,42
CRA0106	38.256.991	5,43	9,39	14,82	38.257.263	5,45	9,34	14,79	0,34	-0,44	-0,16
CRA0107	30.345.052	4,45	8,34	12,79	30.357.995	4,43	8,35	12,78	-0,31	0,10	-0,04
CRA0108	39.730.997	5,48	10,33	15,81	39.731.140	5,50	10,30	15,80	0,36	-0,31	-0,08
CRA0110	40.235.958	5,47	11,46	16,94	40.235.657	5,50	11,51	17,00	0,43	0,38	0,39
CRA0111	22.509.342	5,21	10,33	15,54	22.509.765	5,17	10,27	15,44	-0,69	-0,62	-0,65
CRA0112	19.185.653	5,93	12,77	18,71	19.185.488	5,91	12,70	18,61	-0,38	-0,58	-0,52
CRA0113	35.777.606	2,15	4,71	6,85	35.777.586	2,15	4,70	6,85	0,31	-0,11	0,02
CRA0114	49.005.677	6,42	8,39	14,81	49.005.640	6,40	8,33	14,74	-0,23	-0,70	-0,50
CRA0115	32.377.139	3,02	6,75	9,76	32.377.055	3,02	6,75	9,77	0,16	0,09	0,11
ERM0102	40.468.905	6,18	7,55	13,73	40.434.019	6,16	7,58	13,74	-0,22	0,34	0,09
ERM0103	32.240.184	6,38	17,70	24,08	32.240.514	6,35	17,73	24,08	-0,53	0,19	0,00
ERM0104	37.728.859	5,70	14,00	19,70	37.728.799	5,70	14,01	19,72	0,00	0,10	0,07
ERM0105	18.224.374	4,81	14,61	19,42	18.224.480	4,82	14,58	19,41	0,20	-0,16	-0,07
ERM0106	26.661.626	4,97	8,85	13,82	26.661.920	4,98	8,81	13,80	0,19	-0,36	-0,16
ERM0107	29.017.828	5,14	11,08	16,22	29.017.792	5,13	11,08	16,22	-0,13	0,04	-0,01
ERM0108	34.272.431	4,66	10,47	15,13	34.271.699	4,68	10,59	15,27	0,52	1,10	0,92
ERM0109	29.506.984	4,96	14,47	19,43	29.506.870	4,99	14,48	19,47	0,60	0,11	0,23
ERM0110	30.577.765	5,41	11,68	17,09	30.577.704	5,43	11,69	17,12	0,41	0,09	0,19
ERM0111	26.321.479	4,92	12,01	16,93	26.322.203	4,93	12,06	16,99	0,25	0,37	0,33
GCA0102	21.504.074	5,99	15,17	21,16	21.500.846	5,49	15,18	20,67	-8,40	0,11	-2,30
GCA0103	30.227.693	6,28	18,96	25,24	30.227.430	6,32	18,68	25,00	0,55	-1,44	-0,95
GCA0104	29.416.354	5,15	11,86	17,01	29.416.830	5,18	11,90	17,08	0,67	0,31	0,42
GCA0105	23.620.402	6,48	18,39	24,86	23.620.794	6,45	18,26	24,71	-0,38	-0,71	-0,62
GNA0102	33.212.057	5,34	12,32	17,66	33.212.274	5,39	12,33	17,71	0,84	0,07	0,30

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
GNA0103	31.633.769	4,01	11,46	15,48	31.632.322	4,06	11,65	15,71	1,13	1,61	1,49
GNA0104	10.742.532	2,41	10,84	13,25	10.742.065	2,37	11,99	14,36	-1,85	10,62	8,35
GNA0105	4.918.514	4,54	8,95	13,49	4.918.729	4,62	8,82	13,44	1,77	-1,47	-0,38
GNA0107	30.356.886	6,43	8,98	15,41	30.386.383	6,41	8,99	15,40	-0,31	0,06	-0,10
GNA0109	6.803.112	4,78	13,58	18,36	6.803.960	4,89	13,78	18,67	2,36	1,45	1,69
GNA0111	13.731.859	7,06	44,69	51,75	13.746.766	7,03	44,66	51,69	-0,40	-0,06	-0,11
GNA0112	32.645.544	2,18	9,07	11,25	32.645.213	2,23	9,22	11,45	2,63	1,65	1,84
GNA0113	31.720.355	4,68	11,09	15,76	31.720.494	4,71	11,04	15,75	0,77	-0,41	-0,06
GNA0114	10.131.250	2,12	8,59	10,71	10.131.027	2,12	8,61	10,73	0,19	0,21	0,20
GNA0115	39.167.148	3,44	13,23	16,67	39.167.645	3,47	13,43	16,90	0,73	1,52	1,36
HIP0102	3.555.575	8,06	11,02	19,08	3.554.821	8,11	10,98	19,09	0,67	-0,35	0,08
HIP0103	13.798.475	6,80	6,49	13,29	13.797.920	6,80	6,46	13,26	0,10	-0,49	-0,19
HIP0104	12.465.072	7,08	8,14	15,22	12.464.984	7,16	8,14	15,30	1,09	0,03	0,52
HIP0105	11.152.052	2,93	5,95	8,87	11.152.149	2,95	5,92	8,87	0,81	-0,47	-0,05
HIP0106	34.353.620	4,51	6,37	10,88	34.353.626	4,55	6,35	10,90	0,80	-0,26	0,18
HIP0107	8.920.966	5,41	8,74	14,16	8.920.679	5,41	8,64	14,05	-0,02	-1,20	-0,75
HIP0108	14.575.789	3,78	3,14	6,92	14.575.799	3,79	3,13	6,92	0,28	-0,19	0,07
HIP0109	10.902.999	10,48	9,77	20,25	10.902.535	10,61	9,96	20,57	1,25	1,98	1,60
HIP0110	1.831.304	2,23	2,77	5,00	1.831.313	2,24	2,77	5,01	0,38	0,00	0,16
HIP0111	2.088.289	4,45	8,04	12,50	2.088.167	4,46	8,07	12,54	0,16	0,41	0,32
HIP0112	6.007.301	3,18	2,85	6,03	6.008.034	3,21	2,90	6,10	0,84	1,83	1,31
HIP0113	24.490.245	4,46	5,37	9,83	24.490.298	4,46	5,33	9,79	-0,11	-0,64	-0,40
HIP0114	11.895.913	3,15	6,52	9,66	11.895.751	3,17	6,53	9,70	0,92	0,18	0,43
ITN0102	30.495.096	6,03	24,22	30,25	30.495.264	6,07	24,04	30,11	0,66	-0,76	-0,47
ITN0103	41.571.883	6,61	16,20	22,81	41.572.344	6,64	16,27	22,90	0,47	0,41	0,43
ITN0104	14.999.774	4,30	15,42	19,72	15.000.046	4,30	15,36	19,66	0,06	-0,39	-0,29
ITN0105	37.255.561	6,08	4,71	10,79	37.255.532	6,06	4,70	10,77	-0,28	-0,08	-0,19
ITN0106	18.092.119	4,10	15,82	19,91	18.091.927	4,09	15,75	19,84	-0,29	-0,42	-0,39
ITN0107	39.295.324	4,95	10,72	15,67	39.295.563	4,93	10,69	15,63	-0,26	-0,31	-0,29
ITN0108	47.873.554	6,92	14,90	21,82	47.873.668	6,93	14,94	21,87	0,14	0,29	0,24
ITN0109	13.673.347	3,88	13,17	17,05	13.672.988	3,89	13,15	17,04	0,16	-0,16	-0,09
ITN0110	37.373.855	5,53	15,64	21,16	37.373.982	5,55	15,68	21,23	0,45	0,29	0,33
ITN0111	35.328.593	6,30	14,21	20,51	35.328.323	6,34	14,25	20,59	0,68	0,27	0,39
ITR0102	22.728.916	5,34	11,25	16,59	22.728.469	5,45	11,31	16,76	2,14	0,51	1,04
ITR0103	31.326.129	3,42	8,28	11,69	31.325.677	3,42	8,23	11,65	0,17	-0,59	-0,37
ITR0104	29.771.072	5,65	12,55	18,20	29.770.922	5,65	12,48	18,13	0,00	-0,55	-0,38
ITR0105	29.369.163	6,17	11,87	18,05	29.369.929	6,19	11,90	18,09	0,37	0,18	0,25

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
ITR0106	51.800.438	6,40	10,70	17,10	51.800.411	6,41	10,70	17,11	0,09	0,01	0,04
ITR0107	35.764.657	4,83	11,59	16,41	35.764.922	4,70	11,30	16,00	-2,67	-2,43	-2,50
ITR0108	42.121.804	5,08	10,22	15,29	42.121.689	5,07	10,24	15,31	-0,17	0,21	0,08
ITR0109	15.118.841	5,76	10,81	16,57	15.119.324	5,71	10,55	16,26	-0,84	-2,38	-1,85
ITR0110	14.238.844	5,61	12,13	17,74	14.238.491	5,62	12,20	17,82	0,19	0,53	0,42
ITR0111	18.820.623	6,32	15,77	22,09	18.820.862	6,24	15,51	21,75	-1,24	-1,71	-1,57
ITR0112	20.056.835	3,12	8,43	11,55	20.057.137	3,13	8,45	11,58	0,19	0,25	0,23
ITR0113	45.241.494	3,02	5,12	8,14	45.241.604	3,03	5,17	8,19	0,20	0,86	0,62
ITR0114	24.647.741	5,69	15,34	21,03	24.647.410	5,69	15,40	21,10	0,14	0,38	0,31
MAT0103	43.796.325	4,91	17,53	22,44	43.796.668	4,92	17,55	22,47	0,23	0,11	0,14
MAT0104	35.425.121	3,39	9,89	13,28	35.425.126	3,39	9,88	13,27	0,18	-0,13	-0,05
MAT0105	56.095.900	6,37	15,92	22,28	56.095.984	6,42	15,95	22,37	0,84	0,21	0,39
MAT0106	27.531.425	5,09	8,18	13,28	27.531.512	5,09	8,17	13,25	-0,14	-0,22	-0,19
MAT0107	34.788.690	5,81	13,45	19,26	34.788.779	5,83	13,56	19,39	0,24	0,85	0,66
MAT0108	30.195.467	4,29	2,37	6,66	30.195.459	4,29	2,36	6,65	0,07	-0,33	-0,07
MAT0109	38.673.237	3,83	16,56	20,39	38.673.162	3,85	16,67	20,52	0,53	0,65	0,63
MAT0110	42.360.365	5,84	15,87	21,71	42.360.761	5,78	15,80	21,58	-0,87	-0,48	-0,58
MAT0111	45.314.448	4,10	9,55	13,65	45.314.515	4,09	9,54	13,63	-0,20	-0,11	-0,13
MAT0112	36.468.425	8,10	10,03	18,13	36.710.075	8,02	10,00	18,02	-0,93	-0,32	-0,59
MAT0113	31.867.610	7,15	10,89	18,04	31.849.475	7,17	10,87	18,05	0,32	-0,12	0,06
MAT0114	52.357.199	6,95	13,58	20,52	52.357.531	6,92	13,53	20,45	-0,39	-0,37	-0,38
MOO0102	11.475.802	5,05	5,72	10,76	11.475.973	5,06	5,69	10,75	0,25	-0,45	-0,12
MOO0103	17.970.038	2,21	1,72	3,93	17.970.031	2,22	1,70	3,92	0,24	-0,80	-0,22
MOO0105	2.700.386	4,94	4,49	9,43	2.700.357	4,94	4,49	9,43	0,05	0,09	0,07
MOO0106	24.960.092	3,22	1,73	4,95	24.960.095	3,22	1,73	4,96	0,20	0,14	0,18
MOO0107	35.328.282	2,79	1,72	4,51	35.328.275	2,79	1,72	4,51	0,04	-0,05	0,01
MOO0108	16.781.948	5,26	5,81	11,06	16.782.083	5,24	5,79	11,04	-0,25	-0,27	-0,26
MPA0102	11.582.562	3,67	10,48	14,15	11.582.665	3,66	10,46	14,12	-0,24	-0,21	-0,21
MPA0103	25.576.328	4,32	11,28	15,59	25.576.562	4,32	11,26	15,58	0,16	-0,16	-0,07
MPA0104	10.401.213	3,56	9,20	12,76	10.401.615	3,54	9,17	12,70	-0,56	-0,37	-0,42
MPA0105	24.247.655	4,98	6,78	11,77	24.247.686	5,00	6,75	11,75	0,30	-0,53	-0,18
MPA0106	35.923.213	6,36	14,71	21,07	35.923.511	6,39	14,72	21,11	0,45	0,13	0,23
MPA0107	24.268.025	6,05	20,74	26,79	24.267.768	6,14	20,78	26,92	1,47	0,21	0,49
MPA0109	32.910.494	5,78	18,49	24,27	32.910.346	5,81	18,52	24,34	0,57	0,15	0,25
MPA0110	32.405.218	6,13	12,10	18,23	32.405.133	6,15	12,11	18,26	0,40	0,06	0,18
MPA0111	24.709.319	6,13	15,92	22,05	24.709.443	6,14	15,88	22,02	0,20	-0,25	-0,13
MPA0112	33.868.402	7,21	20,17	27,37	33.868.620	7,17	20,30	27,47	-0,51	0,66	0,36

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
MSA0102	19.707.392	4,30	12,91	17,21	19.707.187	4,33	12,94	17,26	0,64	0,21	0,32
MSA0103	34.534.607	5,67	13,77	19,44	34.534.262	5,68	13,74	19,41	0,16	-0,28	-0,15
MSA0104	35.506.764	6,77	8,26	15,03	35.421.279	6,83	8,23	15,06	0,92	-0,32	0,24
MSA0106	14.605.547	5,26	12,43	17,70	14.605.612	5,27	12,42	17,68	0,10	-0,15	-0,08
MSA0107	31.219.475	6,50	13,82	20,32	31.219.797	6,50	13,74	20,24	-0,06	-0,55	-0,39
MSA0108	11.051.163	5,94	7,47	13,41	11.050.611	5,96	7,46	13,42	0,28	-0,19	0,01
MSA0109	18.792.234	5,56	14,55	20,11	18.792.208	5,58	14,55	20,13	0,29	0,01	0,09
MSA0110	50.029.423	5,43	9,37	14,79	50.029.271	5,41	9,38	14,79	-0,29	0,16	0,00
MSA0111	41.863.353	6,15	5,10	11,26	41.863.162	6,16	5,10	11,26	0,07	-0,02	0,03
MSA0112	33.789.898	5,46	13,10	18,56	33.789.967	5,49	13,08	18,56	0,48	-0,18	0,02
MSA0113	14.208.035	5,25	13,86	19,12	14.208.088	5,26	13,85	19,11	0,10	-0,11	-0,05
MSA0114	6.300.201	5,42	14,67	20,10	6.300.252	5,44	14,65	20,09	0,31	-0,16	-0,03
MSA0115	12.765.861	4,37	11,27	15,64	12.765.911	4,38	11,25	15,64	0,33	-0,16	-0,02
MSA0116	26.345.868	1,52	5,88	7,40	26.345.906	1,55	5,88	7,43	1,87	-0,03	0,36
NAC0102	50.236.988	6,27	15,62	21,89	50.237.389	6,23	15,52	21,75	-0,53	-0,65	-0,62
NAC0103	25.857.096	5,38	16,83	22,21	25.857.303	5,34	16,80	22,14	-0,76	-0,18	-0,32
NAC0104	32.892.231	5,13	13,24	18,37	32.891.958	5,12	13,29	18,41	-0,19	0,37	0,21
NAC0107	30.131.616	5,20	11,76	16,96	30.131.310	5,20	11,73	16,94	0,12	-0,20	-0,10
NAC0108	7.260.211	4,66	2,79	7,45	7.260.380	4,62	3,02	7,64	-0,83	8,17	2,54
NAC0109	34.515.883	4,80	14,56	19,36	34.515.815	4,82	14,57	19,38	0,29	0,07	0,13
NAC0110	36.670.254	5,16	14,04	19,20	36.670.131	5,12	14,08	19,20	-0,85	0,33	0,01
NAC0111	31.579.877	7,70	15,17	22,87	31.579.474	7,78	15,24	23,01	0,95	0,44	0,61
NAC0112	32.038.496	5,65	15,64	21,29	32.038.392	5,67	15,66	21,33	0,30	0,15	0,19
NAC0113	20.114.800	5,83	11,75	17,59	20.114.443	5,84	11,72	17,56	0,01	-0,23	-0,15
ORA0102	23.048.677	5,47	5,95	11,42	23.048.378	5,47	5,94	11,41	0,09	-0,25	-0,08
ORA0103	33.052.176	6,33	3,21	9,54	33.052.290	6,34	3,20	9,54	0,15	-0,17	0,04
ORA0104	23.229.488	4,13	2,04	6,16	23.229.484	4,13	2,02	6,15	0,14	-0,90	-0,21
ORA0105	36.078.241	4,28	1,47	5,75	36.078.246	4,29	1,47	5,76	0,18	0,04	0,15
PEN0102	13.741.741	6,48	11,46	17,94	13.742.249	6,52	11,43	17,96	0,70	-0,23	0,11
PEN0103	43.567.189	8,08	10,24	18,32	43.566.908	8,07	10,16	18,23	-0,18	-0,80	-0,53
PEN0104	26.713.131	5,49	8,40	13,89	26.713.197	5,52	8,39	13,90	0,46	-0,15	0,10
PEN0105	13.304.119	5,63	6,51	12,14	13.304.180	5,64	6,50	12,14	0,22	-0,20	0,00
PEN0106	21.485.351	9,02	9,50	18,52	21.485.812	9,00	9,42	18,42	-0,23	-0,85	-0,55
PEN0107	7.858.715	0,10	0,15	0,25	7.858.752	0,10	0,15	0,26	2,27	2,15	2,23
PEN0108	39.237.283	5,54	7,05	12,59	39.237.052	5,55	7,07	12,61	0,15	0,22	0,19
PEN0109	30.662.274	6,19	11,00	17,18	30.662.273	6,22	10,99	17,21	0,57	-0,07	0,16
PEN0110	30.271.090	5,86	9,08	14,95	30.271.266	5,87	9,07	14,94	0,10	-0,18	-0,07

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia

(continua)

Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
PEN0111	40.774.422	4,42	8,64	13,05	40.774.732	4,43	8,57	13,00	0,34	-0,76	-0,39
PEN0112	21.598.156	1,21	1,75	2,96	21.598.123	1,22	1,73	2,95	0,50	-1,05	-0,42
PEN0113	17.836.127	5,83	11,08	16,91	17.836.333	5,86	11,03	16,89	0,50	-0,45	-0,12
PEN0114	5.497.883	3,10	7,84	10,94	5.499.055	3,11	7,77	10,88	0,42	-0,89	-0,52
PEN0115	20.917.125	5,29	6,34	11,62	20.915.962	5,31	6,31	11,61	0,38	-0,47	-0,09
PSO0003	702.867	2,89	8,60	11,48	702.899	2,88	8,59	11,47	-0,10	-0,12	-0,11
PSO0005	4.173.136	3,17	7,05	10,21	4.173.344	3,17	7,03	10,19	0,08	-0,31	-0,19
PSO0017	8.408.860	2,09	0,75	2,84	8.408.853	2,09	0,75	2,84	0,01	-0,13	-0,03
PSO0021	10.479.543	7,77	5,68	13,44	10.479.722	7,79	5,70	13,49	0,35	0,41	0,37
SND0102	27.127.756	2,99	3,26	6,25	27.127.956	2,98	3,25	6,23	-0,27	-0,27	-0,27
SND0115	28.618.789	2,28	3,41	5,70	28.618.725	2,27	3,42	5,69	-0,62	0,12	-0,18
TIR0102	28.963.245	4,98	17,97	22,95	28.963.194	4,98	17,98	22,96	0,02	0,08	0,07
TIR0103	10.985.556	5,36	52,41	57,77	10.958.500	5,33	52,58	57,91	-0,62	0,33	0,25
TIR0104	40.494.702	5,23	14,39	19,63	40.494.697	5,21	14,45	19,67	-0,33	0,41	0,22
TIR0105	38.132.512	6,62	18,42	25,04	38.132.405	6,63	18,34	24,97	0,16	-0,45	-0,29
TIR0106	45.516.258	6,56	24,75	31,30	45.516.133	6,63	24,65	31,28	1,06	-0,39	-0,09
TIR0107	38.515.266	6,00	23,29	29,30	38.514.643	6,01	23,45	29,46	0,13	0,69	0,58
TIR0108	45.489.554	5,26	16,71	21,97	45.489.795	5,27	16,65	21,92	0,13	-0,32	-0,22
TIR0109	38.274.853	6,12	12,49	18,61	38.255.092	6,11	12,51	18,62	-0,09	0,12	0,05
TIR0110	44.285.822	6,54	24,10	30,64	44.285.941	6,54	23,93	30,47	0,03	-0,70	-0,54
TIR0111	34.666.272	3,91	14,60	18,51	34.666.563	3,93	14,54	18,48	0,55	-0,40	-0,20
TTI0102	8.548.873	11,93	7,90	19,82	8.520.882	12,04	7,85	19,89	0,96	-0,62	0,33
TTI0103	21.044.775	2,66	12,22	14,88	21.042.953	2,82	12,36	15,18	5,75	1,15	1,98
TTI0104	36.647.802	6,70	15,79	22,49	36.647.903	6,82	15,83	22,66	1,89	0,28	0,76
TTI0105	34.613.201	6,78	3,33	10,10	34.613.926	6,82	3,35	10,17	0,55	0,82	0,64
TTI0106	17.881.991	2,81	1,23	4,04	17.882.457	2,83	1,23	4,06	0,71	0,02	0,50
TTI0108	2.373.291	0,73	0,22	0,95	2.373.320	0,73	0,23	0,96	0,53	3,46	1,22
TTI0109	2.881.413	2,08	13,33	15,41	2.881.412	2,08	13,33	15,41	0,08	-0,01	0,00
TTI0110	24.636.938	10,74	15,56	26,31	24.637.200	10,69	15,51	26,20	-0,47	-0,34	-0,39
TTI0111	11.195.540	1,36	1,42	2,78	11.195.251	1,36	1,34	2,70	0,41	-6,11	-2,93
TTI0112	9.497.004	5,28	14,16	19,43	9.497.119	5,35	14,10	19,46	1,48	-0,37	0,13
TTI0113	4.740.097	4,80	2,63	7,43	4.740.022	4,82	2,63	7,46	0,44	0,11	0,32
TTI0114	12.545.066	4,53	14,44	18,96	12.544.926	4,64	14,40	19,04	2,46	-0,26	0,39
UTI0102	22.310.211	4,76	1,19	5,95	22.310.258	4,76	1,19	5,95	-0,12	0,37	-0,03
UTI0103	50.686.784	5,04	6,68	11,72	50.686.890	5,02	6,67	11,69	-0,34	-0,16	-0,24
UTI0104	38.375.352	6,76	9,19	15,95	38.379.032	6,72	9,23	15,95	-0,52	0,37	-0,01
UTI0105	43.443.050	5,79	9,30	15,09	43.443.394	5,78	9,39	15,18	-0,14	1,01	0,57

Tabela 12 - Comparação dos resultados de cálculos de perdas: Metodologia Regulatória X Nova Metodologia (conclusão)

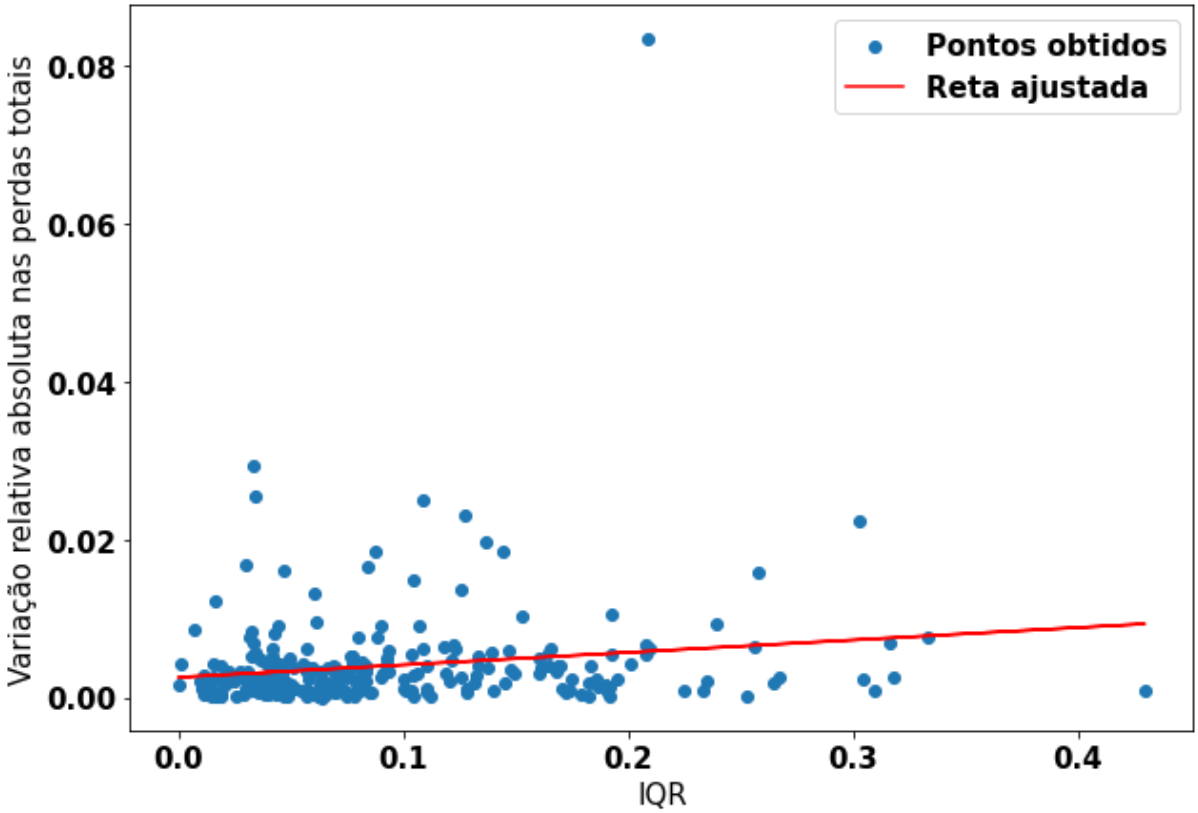
Alimentador	Metodologia Regulatória Atual				Nova Metodologia				Variação		
	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	Energia injetada (kWh)	PT (%)	PNT (%)	PD (%)	PT (%)	PNT (%)	PD (%)
UTI0107	8.121.055	2,33	3,58	5,91	8.121.067	2,30	3,56	5,86	-1,47	-0,47	-0,87
UTI0108	25.003.348	5,08	10,09	15,17	25.003.327	5,05	10,05	15,10	-0,56	-0,48	-0,51
UTI0110	14.323.992	5,13	34,26	39,39	14.338.130	5,14	34,21	39,35	0,16	-0,14	-0,10
VEM0102	34.998.737	5,61	10,40	16,01	34.998.956	5,61	10,37	15,98	0,03	-0,30	-0,18
VEM0103	27.970.880	4,74	13,95	18,69	27.970.651	4,76	14,00	18,76	0,43	0,34	0,36
VEM0104	30.372.134	5,30	9,35	14,65	30.372.265	5,34	9,32	14,66	0,73	-0,27	0,09
VEM0105	33.516.240	4,51	9,21	13,72	33.516.315	4,52	9,20	13,72	0,16	-0,14	-0,04
VEM0106	22.073.606	5,07	11,00	16,07	22.073.122	5,11	11,10	16,22	0,88	0,94	0,92
VEM0107	13.698.681	4,31	9,81	14,12	13.698.439	4,31	9,77	14,08	0,04	-0,36	-0,24
VEM0108	29.287.365	4,55	9,21	13,76	29.287.445	4,55	9,20	13,75	0,07	-0,13	-0,06
VEM0109	38.752.190	5,52	12,29	17,81	38.752.216	5,52	12,28	17,81	0,11	-0,07	-0,02
VEM0110	23.897.359	4,28	9,91	14,20	23.897.463	4,27	9,88	14,16	-0,23	-0,28	-0,26
VEM0111	18.723.809	4,02	8,96	12,99	18.723.578	4,03	8,93	12,96	0,21	-0,40	-0,21
VEM0112	24.196.135	4,40	9,60	14,00	24.195.892	4,40	9,54	13,94	0,16	-0,67	-0,41
VFO0102	28.771.203	5,10	7,31	12,41	28.771.023	5,10	7,26	12,36	0,05	-0,69	-0,39
VFO0103	11.583.516	3,79	8,63	12,42	11.583.811	3,80	8,57	12,37	0,28	-0,71	-0,41
VFO0104	6.930.210	5,08	8,56	13,64	6.930.551	5,04	8,49	13,53	-0,85	-0,83	-0,83
VFO0105	13.230.194	5,56	12,15	17,71	13.230.202	5,58	12,15	17,73	0,31	-0,01	0,09
VFO0106	13.042.620	3,74	8,37	12,11	13.042.708	3,75	8,33	12,08	0,23	-0,48	-0,26
VFO0107	27.498.440	3,39	7,59	10,98	27.498.450	3,39	7,59	10,98	0,02	-0,04	-0,02
VFO0108	37.595.176	6,64	10,24	16,88	37.594.948	6,67	10,21	16,88	0,38	-0,25	0,00
VFO0109	47.090.783	6,34	2,34	8,68	47.090.791	6,35	2,33	8,68	0,09	-0,14	0,03
VFO0110	42.304.045	6,22	4,44	10,66	42.303.512	6,22	4,41	10,63	0,06	-0,64	-0,23
VFO0111	31.769.959	6,48	4,93	11,42	31.770.051	6,49	4,87	11,36	0,11	-1,23	-0,47
VFO0112	16.716.540	4,17	5,88	10,05	16.716.544	4,18	5,87	10,05	0,15	-0,05	0,04
VFO0113	9.839.653	5,96	0,08	6,04	9.840.085	5,95	0,04	5,99	-0,10	-44,71	-0,69
VFO0114	42.837.094	5,53	8,31	13,84	42.837.320	5,53	8,28	13,81	-0,10	-0,37	-0,26
VFO0115	6.303.959	3,27	8,08	11,35	6.303.842	3,28	8,08	11,36	0,27	0,10	0,15
VPR0102	30.375.344	5,29	7,39	12,68	30.375.357	5,33	7,44	12,76	0,67	0,69	0,68
VPR0109	34.953.655	4,41	3,49	7,90	34.953.581	4,42	3,47	7,89	0,32	-0,57	-0,08
VPR0114	12.333.676	3,24	4,81	8,04	12.333.671	3,25	4,80	8,05	0,25	-0,02	0,08
VPR0115	36.370.349	4,92	2,24	7,16	36.371.071	4,94	2,25	7,18	0,27	0,38	0,31
VTA0102	40.849.108	5,90	12,91	18,81	40.849.406	5,91	12,86	18,77	0,21	-0,44	-0,23
VTA0103	32.136.217	4,30	7,55	11,85	32.136.633	4,30	7,48	11,77	0,03	-1,00	-0,63
VTA0104	36.740.105	5,11	4,76	9,87	36.740.558	5,11	4,72	9,84	0,12	-0,77	-0,31
VTA0106	43.062.876	5,61	4,59	10,20	43.063.033	5,62	4,57	10,19	0,13	-0,39	-0,10
Total	**6.942.655.490**	**5,25**	**10,77**	**16,02**	**6.942.754.987**	**5,26**	**10,76**	**16,02**	**0,15**	**-0,05**	**0,02**

Fonte: Autor

Nota-se que para a maioria dos alimentadores há pouca diferença entre os valores de PT, PNT e PD com relação às diferentes metodologias de cálculo de perdas, porém alguns tiveram diferença significativa como o VFO0113, TTI0111, TTI0103, GNA0104 e GCA0102. A pequena diferença na maioria dos alimentadores leva o resultado geral, correspondente a última linha da tabela, ser de apenas 0,15% para as PT, -0.05% para as PNT e de 0,02% para as PD. Isso deve ocorrer por a maioria dos alimentadores atender, cada um, uma região com um Índice Potencial de Fraude aproximadamente uniforme, o que leva ao novo método de cálculo proposto não trazer diferença significativa nos resultados, já que toda a região atendida por certo alimentador teria, aproximadamente, o mesmo IPF.

Para se verificar essa hipótese, foi calculado o *Inter Quartile Range* (IQR) do Índice Potencial de Fraude de cada alimentador (IQR é a diferença entre o 3º Quartil e o 1º Quartil de uma distribuição de valores) e o IQR foi plotado contra o absoluto da diferença relativa de perdas totais entre a nova metodologia e a metodologia regulatória. Além disso foi ajustada uma reta a esses pontos, conforme mostra a Figura 25.

Figura 25 - Relação entre IQR do IPF do alimentador e variação relativa absoluta de perdas totais entre as metodologias



Fonte: Autor

Nota-se uma leve inclinação positiva, a saber +0,016, ou seja, uma relação positiva entre quanto maior o IQR do Índice Potencial de Fraude do alimentador, maior a diferença absoluta para as perdas totais entre as duas metodologias. Não é uma relação forte, mas um indício de que as diferenças entre as duas metodologias foram pequenas, porque a maioria dos alimentadores possui um baixo IQR do Índice Potencial de Fraude, como pode ser visto na Figura 25, já que a maioria dos pontos se acumula abaixo de 0,2. O notebook utilizado para obter tal análise se encontra no Apêndice J.

6. CONCLUSÕES

Este trabalho apresentou modelos de *machine learning* para detecção de fraude que utilizam treinamento supervisionado com dados de inspeções e a proposição de uma nova abordagem para o cálculo regulatório de perdas na distribuição.

Com relação aos modelos de detecção de fraude, o tipo de modelo que obteve o melhor desempenho foi o XGBoost, obtendo uma métrica de F_1 de 0,121. Apesar de ser um valor baixo de F_1, foi mostrado que a priorização de inspeções através da probabilidade de fraude gerada pelo modelo pode gerar um aumento na precisão das equipes de inspeção de 51%. Importante salientar que se espera que o modelo teria melhor performance se houvesse uma estrutura de AMI, com dados de *smart meters* com grande resolução, ou seja, uma resolução de medida por hora ou maior, já que isso poderia melhor caracterizar o perfil de carga de um fraudador contra um perfil de carga de um não fraudador. Mas, para os modelos criados, só havia a informação do consumo mensal faturado e se tentou criar várias variáveis para caracterizar esse consumo mensal faturado, para que os modelos pudessem tentar achar uma distinção entre um consumo mensal característico de um fraudador contra um consumo mensal de um não fraudador.

Já com relação a metodologia proposta para o cálculo regulatório de perdas, que utiliza os dados de inspeções em conjunto com a técnica de *Kriging* para interpolação em toda a região piloto para a obtenção do chamado de Índice Potencial de Fraude (IPF), notou-se que não houve diferença significativa entre as perdas calculadas com a metodologia proposta e a metodologia regulatória atual, exceto para alguns alimentadores. A investigação do motivo disso mostrou que a maior parte dos alimentadores atende regiões que possuem uma distribuição do IPF com baixa variabilidade, portanto, para os alimentadores que possuem baixa variabilidade do IPF as duas metodologias de cálculo apresentam o mesmo resultado, como deveria ser esperado. Notou-se que há uma tendência de quanto maior a variabilidade do IPF, maiores seriam as diferenças absolutas entre as perdas calculadas pela nova metodologia e pela metodologia regulatória.

7. PUBLICAÇÕES REALIZADAS

Durante o período de pesquisa deste trabalho, duas publicações foram realizadas.

A primeira publicação (PULZ, et al., 2017) se refere a modelos de *machine learning* para detecção de fraude na baixa tensão. Foi realizada em parceria com a distribuidora brasileira Sulgipe. O estudo mostrou que com o uso de SVM, que era o estado da arte em termos de *machine learning* na época da escrita do estudo, obteve bons resultados para a detecção de potenciais unidades consumidoras fraudadoras com o uso de variáveis relacionadas ao consumo mensal, ao cadastro e a indicadores socioeconômicos.

A segunda publicação (PULZ & ALMEIDA, 2021) se refere a mesma metodologia proposta no capítulo 5 deste trabalho, porém naquele caso foram utilizados os dados da BDGD de 2018, mas os mesmos dados de inspeções e a mesma área piloto. Como é mostrado no capítulo 5, os dados da BDGD de 2019 não contêm informações precisas sobre as conexões das cargas aos alimentadores e, provavelmente, o mesmo ocorre com os dados da BDGD de 2018. O estudo mostrou que a diferença entre o cálculo de perdas com a metodologia proposta e a metodologia regulatória não apresenta diferenças significativas, porém nenhuma análise mais aprofundada daquilo que motivou esse comportamento foi realizada, ao contrário do que foi feito no capítulo 5 deste trabalho.

8. FUTUROS TRABALHOS

Como desdobramento deste trabalho, os seguintes itens poderiam ser objeto de estudo futuro:

- Testar a metodologia proposta de cálculo regulatório de PNT para uma distribuidora como um todo para verificação de haver diferença entre a metodologia regulatória atual e a proposta, considerando que a não diferença significativa obtida neste trabalho entre as metodologias pode ter sido obtida por sorte;
- Melhorar o modelo de detecção de fraude utilizando uma superamostragem ou uma mistura de sub e superamostragem.

REFERÊNCIAS BIBLIOGRÁFICAS

7-Zip. (2022). *7-Zip*. Fonte: https://www.7-zip.org/. Acesso em: 8 de mar. 2022

ANEEL. (2008). *Nota Técnica nº 342/2008 - Metodologia de tratamento regulatório para perdas não técnicas de energia elétrica.* Brasília.

ANEEL. (2013). *Nota Técnica nº 453/2013 - Discussão conceitual sobre a metodologia de definição do nível regulatório de perdas na distribuição, a qualidade do serviço e os encentivos regulatórios.* Brasília.

ANEEL. (2014a). *Nota Técnica nº 188/2014 - Atualização do índice de complexidade soceioeconômica adotado para definição do nível regulatório de perdas não técnicas na distribuição.* Brasília.

ANEEL. (2014b). *Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional - Módulo 2 - Planejamento da Expansão do Sistema de Distribuição.* Brasília.

ANEEL. (2015). *Procedimentos de Regulação Tarifária – PRORET, Módulo 2: Revisão Tarifária Periódica das Concessionárias de Distribuição – Submódulo 2.6 – Perdas de Energia.*

ANEEL. (2016). *Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional - Módulo 10 - Sistema de Informação Geográfica Regulatório.* Brasília.

ANEEL. (2018). *Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional - Módulo 7 - Cálculo de Perdas na Distribuição.* Brasília.

ANEEL. (2019). *Audiência 011/2019 - Relatórios CTR.* Fonte: https://www.aneel.gov.br/audiencias-publicas-antigas?p_p_id=participacaopublica_WAR_participacaopublicaportlet&p_p_lifecycle=2&_participacaopublica_WAR_participacaopublicaportlet_ideDocumento=32299&_participacaopublica_WAR_participacaopublicaportlet_tipoF

ANEEL. (2021a). *Manual BDGD e Anexos*. Fonte: https://sistemas.aneel.gov.br/concessionarios/administracao/Manual_BDGD_e_ANEXOS.zip. Acesso em: 28 fev. 2021

ANEEL. (2021b). *Perdas de Energia*. Fonte: Cálculo Tarifário e Metodologia: https://www.aneel.gov.br/documents/654800/18766993/Base_Perdas_Internet+-06-+2020+%28dados+2019%29.xlsx/27884ca0-97bb-4d46-00bb-465dbe8dcec2. Acesso em: 17 mai. 2021

BERGSTRA, J., YAMINS, D., & COX, D. D. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *30th International Conference on Machine Learning (ICML 2013).*

BREIMAN, L. (2001). Random Forests. *Machine Learning*, pp. 5-32.

CHEN, T., & C., G. (2016). XGBoost: A Scalable Tree Boosting System. *22nd SIGKDD Conference on Knowledge Discovery and Data Mining.*

CHILÈS, J., & DESASSIS, N. (2018). *Handbook of Mathematical Geosciences - Fifty Years of Kriging.* Springer.

DONADEL, C., ANICIO, J., FREDES, M., VAREJÃO, F., COMARELA, G., & PERIM, G. (2009). A methodology to refine the technical losses calculation from estimates of non-technical losses. *20th International Conference on Electricity Distribution.*

EPRI. (2022). *OpenDSS*. Fonte: https://www.epri.com/pages/sa/opendss. Acesso em: 1 mar. 2022

Fala.BR. (2020). Fonte: Plataforma Integrada de Ouvidoria e Acesso à Informação: https://falabr.cgu.gov.br. Acesso em: 16 nov. 2020

GDAL. (2022). *GAISInternals*. Fonte: https://download.gisinternals.com/sdk/downloads/release-1930-x64-gdal-3-4-1-mapserver-7-6-4/gdal-304-1930-x64-core.msi. Acesso em: 8 mar. 2022

GLAUNER, P., MEIRA, J. A., VALTCHEV, P., STATE, R., & BETTINGER, F. (2017). The Challenge of Non-Technical Loss Detection using Artificial Intelligence: A Survey. *International Journal of Computational Intelligence Systems (IJCIS)*, pp. 760-775.

HENRIQUES, H. O., CORRÊA, R. L., FORTES, M. Z., BORBA, B. S., & FERREIRA, V. H. (2020). Monitoring technical losses to improve non-technical losses estimation and detection in LV distribution systems. *Measurement*.

HU, T., GUO, Q., SHEN, X., SUN, H., WU, R., & X. H. (2019). Utilizing Unlabeled Data to Detect Electricity Fraud in AMI: A Semisupervised Deep Learning Approach. *IEEE Transactions on Neural Networks and Learning Systems*.

HUBACK, V. B. (2018). Medidas ao Combate a Perdas Elétricas Não Técnicas em Áreas com Severas Restrições à Operação de Sistemas de Distribuição de Energia Elétrica. *Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Planejamento Energético*.

IBGE. (2010). *Características urbanísticas do entorno dos domicílios*. Fonte: https://biblioteca.ibge.gov.br/visualizacao/periodicos/96/cd_2010_entorno_domicilios.pdf

IBGE. (2022a). *Resultados Agregados por Setores Censitários*. Fonte: https://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Resultados_do_Universo/Agregados_por_Setores_Censitarios/. Acesso em: 8 abr. 2022

IBGE. (2022b). *Setores Censitários*. Fonte: https://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_de_setores_censitarios__divisoes_intramunicipais/censo_2010/setores_censitarios_shp/sp/. Acesso em: 8 abr. 2022

LEMAÎTRE, G., NOGUEIRA, F., & ARIDAS, C. K. (2016). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*.

MASSAFERRO, P., MARTINO, J. M., & FERNÁNDEZ, A. (2020). Fraud Detection in Electric Power Distribution: An Approach That Maximizes the Economic Return. *Power Systems IEEE Transactions*, pp. 703-710.

MICROSOFT. (2022a). *SQL Server*. Fonte: https://www.microsoft.com/en-us/sql-server/sql-server-downloads. Acesso em: 2 mar. 2022

MICROSOFT. (2022b). *SQL Server Management Studio*. Fonte: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. Acesso em: 2 mar. 2022

P., C. J., & P., D. (1999). *Geostatistics: Modeling Spatial Uncertainty.* Wiley.

PANDOC. (2022). *Pandoc - a universal document converter*. Fonte: https://pandoc.org/. Acesso em: 2 mar. 2022

PULZ, J. (2022). *Códigos e arquivos para a dissertação*. Fonte: https://1drv.ms/u/s!AjrHqWdOGS_Cs9ho3F7Brj6jBdbpeg?e=KpLeb4

PULZ, J., & ALMEIDA, C. F. (2021). An alternative approach for regulatory evaluation of nontechnical losses in Brazil. *The 26th International Conference and Exhibition on Electricity Distribution.*

PULZ, J., MULLER, R. B., ROMERO, F., MEFFE, A., NETO, Á. F., & JESUS, A. S. (2017). Fraud detection in low-voltage electricity consumers using socio-economic indicators and billing profile in smart grids. *24th International Conference & Exhibition on Electricity Distribution (CIRED).*

QGIS. (2022). *QGIS*. Fonte: https://www.qgis.org/en/site/. Acesso em: 2 mar. 2022

ROSSONI, A., TREVIZAN, R., BRETAS, A., GAZZANA, D., BETTIOL, A., CARNIATO, A., . . . MARTIN, R. (2015). Hybrid formulation for technical and non-technical losses estimation and identification in distribution networks: application in a brazilian power system. *23rd International Conference on Electricity Distribution.*

SAINI, S. (2017). Social and behavioral aspects of electricity theft: An explorative review. *International Journal of Research in Economics and Social Sciences (IJRESS)*, pp. 26-37.

VIEGAS, J. L., ESTEVES, P. R., MELÍCIO, R., MENDES, V. M., & VIEIRA, S. M. (2017). Solutions for detection of non-technical losses in the electricity grid: A review. *Renewable and Sustainable Energy Reviews*, pp. 1256–1268.

WIKIPEDIA. (2022). *GeoJSON*. Fonte: https://en.wikipedia.org/wiki/GeoJSON. Acesso em: 9 mar. 2022

YAN, Z., & WEN, H. (2021). Electricity Theft Detection Base on Extreme Gradient Boosting in AMI. *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*.

YURTSEVEN, Ç. (2015). The causes of electricity thref: An econometric analysis of the case of Turkey. *Utilities Policy*, pp. 70-78.

Apêndice A - *Notebook* Python para treinamento dos modelos de *machine learning*

Esse *notebook* Python utiliza um módulo Python adicional que se encontra no Apêndice B. A versão Python utilizada é 3.10.4. Esse notebook também se encontra no link em (PULZ, 2022) – arquivo “4.0-jp-fraud_modeling.ipynb”. Os seguintes pacotes, com suas respectivas versões, foram utilizados para execução do notebook Python:

argon2-cffi==21.3.0
argon2-cffi-bindings==21.2.0
asttokens==2.0.5
attrs==21.4.0
backcall==0.2.0
beautifulsoup4==4.10.0
bleach==4.1.0
catboost==1.0.5
certifi==2021.10.8
cffi==1.15.0
click==8.1.2
click-plugins==1.1.1
cligj==0.7.2
cloudpickle==2.0.0
colorama==0.4.4
cycler==0.11.0
debugpy==1.6.0
decorator==5.1.1
defusedxml==0.7.1
entrypoints==0.4
executing==0.8.3
fastjsonschema==2.15.3
Fiona==1.8.20
fonttools==4.31.2
future==0.18.2
GDAL==3.4.1
geopandas==0.10.2
graphviz==0.19.1
hyperopt==0.2.7
imbalanced-learn==0.9.0
imblearn==0.0
ipykernel==6.12.1
ipython==8.2.0
ipython-genutils==0.2.0
ipywidgets==7.7.0
jedi==0.18.1
Jinja2==3.1.1

joblib==1.1.0
jsonschema==4.4.0
jupyter==1.0.0
jupyter-client==7.2.1
jupyter-console==6.4.3
jupyter-core==4.9.2
jupyterlab-pygments==0.1.2
jupyterlab-widgets==1.1.0
kiwisolver==1.4.2
lightgbm==3.3.2
MarkupSafe==2.1.1
matplotlib==3.5.1
matplotlib-inline==0.1.3
mistune==0.8.4
munch==2.5.0
nbclient==0.5.13
nbconvert==6.4.5
nbformat==5.3.0
nest-asyncio==1.5.5
networkx==2.7.1
notebook==6.4.10
numpy==1.22.3
packaging==21.3
pandas==1.4.2
pandocfilters==1.5.0
parso==0.8.3
pickleshare==0.7.5
Pillow==9.1.0
plotly==5.7.0
prometheus-client==0.13.1
prompt-toolkit==3.0.29
psutil==5.9.0
pure-eval==0.2.2
py4j==0.10.9.5
pycparser==2.21
Pygments==2.11.2
PyKrige==1.6.1
pyodbc==4.0.32
pyparsing==3.0.7
pyproj==3.3.0
pyrsistent==0.18.1
python-dateutil==2.8.2
pytz==2022.1
pywin32==303
pywinpty==2.0.5
pyzmq==22.3.0
qtconsole==5.3.0
QtPy==2.0.1
Rtree==0.9.7
scikit-learn==1.0.2

scipy==1.8.0
seaborn==0.11.2
Send2Trash==1.8.0
Shapely==1.8.0
six==1.16.0
soupsieve==2.3.1
stack-data==0.2.0
tenacity==8.0.1
terminado==0.13.3
testpath==0.6.0
threadpoolctl==3.1.0
tornado==6.1
tqdm==4.64.0
traitlets==5.1.1
Unidecode==1.3.4
wcwidth==0.2.5
webencodings==0.5.1
widgetsnbextension==3.6.0
xgboost==1.5.2

Modelos para detecção de fraude

2022-04-20 Jonatas Pulz

```
import os
import datetime
import math

import pandas as pd
import numpy as np

import unidecode
import pyodbc

import fiona
from shapely.geometry import box, Point
import geopandas

from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support

import matplotlib
from matplotlib import pyplot as plt
from IPython import display

from models import ClassifierModelTournament, HPClassifierOptimizer, Model,
Transformer
```

Configurações

```
pd.set_option('display.max_rows', 200)
pd.set_option('display.max_columns', 200)
```

```
font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 15}

matplotlib.rc('font', **font)
```

Constantes

```
train_test_split_date = 201705
max_date = 201707
```

Conexão com o banco de dados

```
conn = pyodbc.connect('DRIVER={SQL Server Native Client
11.0};SERVER=localhost;DATABASE=master;Trusted_Connection=yes;')
```

Queries

```
next_date = lambda x: int((pd.to_datetime(x, format='%Y%m') + pd.Timedelta(31,
'd')).strftime(format='%Y%m'))
list_test_dates = [
    next_date(train_test_split_date)
]
while list_test_dates[-1] < max_date:
    list_test_dates.append(next_date(list_test_dates[-1]))

def get_sql_chunk_test_comsumption(ref_date, current_date):
    return f'SELECT \
             \'{current_date}\' AS [ANO_MES] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_1] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_2] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_3] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_4] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_5] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_6] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_7] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_8] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_9] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_10] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_11] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_12] \
             ,(CASE \
                   WHEN TRY_CAST(INSTALACAO AS BIGINT) IS NULL THEN
CAST(INSTALACAO AS NVARCHAR(100)) \
                   ELSE CAST(CAST(INSTALACAO AS BIGINT) AS NVARCHAR(100)) \
             END) AS INSTALACAO_PREPARED \
       FROM \
             [InspectionEnel].[dbo].[Consumption] \
       WHERE \
             [InspectionEnel].[dbo].[Consumption].[ANO_MES] = \'{ref_date}\' \
    '


sql_inspection = f'WITH W_Inspection AS ( \
       SELECT \
             [InspectionEnel].[dbo].[Inspection].* \
             ,(CASE \
```

```
                    WHEN TRY_CAST(INSTALACAO AS BIGINT) IS NULL THEN
CAST(INSTALACAO AS NVARCHAR(100)) \
                    ELSE CAST(CAST(INSTALACAO AS BIGINT) AS NVARCHAR(100)) \
             END) AS INSTALACAO_PREPARED \
      FROM \
             [InspectionEnel].[dbo].[Inspection] \
      WHERE \
             ANOMES >= 201608 \
             AND \
             ANOMES <= 201707 \
             AND \
             HIT_A = 0 \
             AND \
             HIT_N = 0 \
) \
,W_Consumer AS \
( \
      SELECT \
             [InspectionEnel].[dbo].[Consumer_201707].* \
             ,(CASE \
                    WHEN TRY_CAST(INSTALACAO AS BIGINT) IS NULL THEN
CAST(INSTALACAO AS NVARCHAR(100)) \
                    ELSE CAST(CAST(INSTALACAO AS BIGINT) AS NVARCHAR(100)) \
             END) AS INSTALACAO_PREPARED \
      FROM \
             [InspectionEnel].[dbo].[Consumer_201707] \
) \
,W_Consumption AS \
( \
      SELECT \
             [InspectionEnel].[dbo].[Consumption].[ANO_MES] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_1] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_2] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_3] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_4] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_5] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_6] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_7] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_8] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_9] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_10] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_11] \
             ,[InspectionEnel].[dbo].[Consumption].[CONSUMO_MES_12] \
             ,(CASE \
                    WHEN TRY_CAST(INSTALACAO AS BIGINT) IS NULL THEN
CAST(INSTALACAO AS NVARCHAR(100)) \
                    ELSE CAST(CAST(INSTALACAO AS BIGINT) AS NVARCHAR(100)) \
             END) AS INSTALACAO_PREPARED \
      FROM \
             [InspectionEnel].[dbo].[Consumption] \
      WHERE \
             [InspectionEnel].[dbo].[Consumption].[ANO_MES] <=
\'{train_test_split_date}\' \
      UNION ALL \
      ' + ' UNION ALL '.join([get_sql_chunk_test_comsumption(list_test_dates[0],
current_date) for current_date in list_test_dates]) + ' \
) \
,W_Default AS \
( \
```

```
	SELECT \
		[InspectionEnel].[dbo].[Default].* \
		,(CASE \
			WHEN TRY_CAST(INSTALACAO AS BIGINT) IS NULL THEN
CAST(INSTALACAO AS NVARCHAR(100)) \
			ELSE CAST(CAST(INSTALACAO AS BIGINT) AS NVARCHAR(100)) \
		END) AS INSTALACAO_PREPARED \
	FROM \
		[InspectionEnel].[dbo].[Default] \
) \
SELECT \
	W_Inspection.[BASE] \
	,W_Inspection.[Cod Turma] \
	,W_Inspection.[Ordem] \
	,W_Inspection.[instalacao] \
	,W_Inspection.[Estratégia] \
	,W_Inspection.[criador] \
	,W_Inspection.[data_entrada] \
	,W_Inspection.[LS] \
	,W_Inspection.[Dia] \
	,W_Inspection.[hora_final] \
	,W_Inspection.[data_leitura] \
	,W_Inspection.[Dias_Leit_Insp] \
	,W_Inspection.[Leit_03_Cad] \
	,W_Inspection.[Leit_03_Enc] \
	,W_Inspection.[Leit_04_Cad] \
	,W_Inspection.[Leit_06_Cad] \
	,W_Inspection.[Leit_08_Cad] \
	,W_Inspection.[Leit_04_Enc] \
	,W_Inspection.[Leit_06_Enc] \
	,W_Inspection.[Leit_08_Enc] \
	,W_Inspection.[Consumo_Medio] \
	,W_Inspection.[Cod Retorno] \
	,W_Inspection.[Iniciativa] \
	,W_Inspection.[Hit] \
	,W_Inspection.[HIT_A] \
	,W_Inspection.[HIT_F] \
	,W_Inspection.[HIT_N] \
	,W_Inspection.[HIT_O] \
	,W_Inspection.[Hit_205] \
	,W_Inspection.[HIT_203] \
	,W_Inspection.[Qtde_insp_Pos] \
	,W_Inspection.[Hit_Pos] \
	,W_Inspection.[HIT_N_Pos] \
	,W_Inspection.[Hit_Pre] \
	,W_Inspection.[HIT_N_Pre] \
	,W_Inspection.[Hit_205_Pre] \
	,W_Inspection.[Direto] \
	,W_Inspection.[DiaÚtil] \
	,W_Inspection.[ANOMES] \
	,W_Inspection.[Sit Deixada] \
	,W_Inspection.[tipo_estrategia] \
	,W_Inspection.[tipo_TOI] \
	,W_Inspection.[BASE_INSP] \
	,W_Inspection.[Num_Semana] \
	,W_Inspection.[Localidade] \
	,W_Inspection.[Obs] \
	,W_Inspection.[loc_tableau] \
	,W_Inspection.[acao] \
```

```
,W_Inspection.[classe] \
,W_Inspection.[ETD] \
,W_Inspection.[circuito] \
,W_Consumer.[INSTALACAO] AS INSTALACAO_CONSUMER \
,W_Consumer.[DATETO] \
,W_Consumer.[DATEFROM] \
,W_Consumer.[EPTMPAREA] \
,W_Consumer.[EPTMPAREA_TEXT] \
,W_Consumer.[IND_CODE] \
,W_Consumer.[IND_CODE_TEXT] \
,W_Consumer.[EPLOCCONS] \
,W_Consumer.[UCVOLTLEVL] \
,W_Consumer.[UCVOLTLEVL_TEXT] \
,W_Consumer.[EPSTINST] \
,W_Consumer.[EPSTINST_TEXT] \
,W_Consumer.[EPPORTION] \
,W_Consumer.[EPUNLEITU] \
,W_Consumer.[EPNUMFAS] \
,W_Consumer.[EPNUMFAS_TEXT] \
,W_Consumer.[EPOBJLIGA] \
,W_Consumer.[EPNMRUA] \
,W_Consumer.[HOUSE_NUM2] \
,W_Consumer.[POSTCD_GIS] \
,W_Consumer.[EPBAIRRO] \
,W_Consumer.[EPMUNIC] \
,W_Consumer.[LATITUDE] \
,W_Consumer.[LONGITUDE] \
,W_Consumer.[EPCONTRAT] \
,W_Consumer.[EPCTACONT] \
,W_Consumer.[UCMOVEIN_D] \
,W_Consumer.[UCMOVEOUTD] \
,W_Consumer.[BPARTNER] \
,W_Consumer.[EPCDEBAUT] \
,W_Consumer.[ULT_TROCA_TIT] \
,W_Consumer.[FL_BXRENDA] \
,W_Consumer.[TIPO_MEDIDOR] \
,W_Consumer.[FLAG_MEDICAO_EXTERNA] \
,W_Consumer.[FLAG_MEDICAO_ENCAPSULADA] \
,W_Consumer.[ET] \
,W_Consumer.[RELIGADOR] \
,W_Consumer.[ALIMENTADOR] \
,W_Consumer.[SE] \
,W_Consumer.[FL_AREARISCO] \
,W_Consumer.[FLAG_VERTICALIZACAO] \
,W_Consumer.[FIC] \
,W_Consumer.[DIC] \
,W_Consumer.[CIRC_PRIMARIO] \
,W_Consumer.[EPBPCNPJ] \
,W_Consumer.[EPBPCPF] \
,W_Consumer.[FCACTDETID] \
,W_Consumer.[FCACTDETID_TEXT] \
,W_Consumer.[EPDEVICE] \
,W_Consumer.[UCINSTDATE] \
,W_Consumer.[UCREMOVDAT] \
,W_Consumer.[EPEQFABR] \
,W_Consumer.[UCSERIALNR] \
,W_Consumer.[UCDEV_CAT] \
,W_Consumer.[EPPREFIXO] \
,W_Consumer.[EPNELEM] \
```

```
      ,W_Consumer.[EPNFIOS] \
      ,W_Consumer.[EPANOFAB] \
      ,W_Consumer.[EPREFNUM] \
      ,W_Consumption.[CONSUMO_MES_1] \
      ,W_Consumption.[CONSUMO_MES_2] \
      ,W_Consumption.[CONSUMO_MES_3] \
      ,W_Consumption.[CONSUMO_MES_4] \
      ,W_Consumption.[CONSUMO_MES_5] \
      ,W_Consumption.[CONSUMO_MES_6] \
      ,W_Consumption.[CONSUMO_MES_7] \
      ,W_Consumption.[CONSUMO_MES_8] \
      ,W_Consumption.[CONSUMO_MES_9] \
      ,W_Consumption.[CONSUMO_MES_10] \
      ,W_Consumption.[CONSUMO_MES_11] \
      ,W_Consumption.[CONSUMO_MES_12] \
      ,ISNULL(W_Default.[INAD_MES_1], 0) AS INAD_MES_1 \
      ,ISNULL(W_Default.[INAD_MES_2], 0) AS INAD_MES_2 \
      ,ISNULL(W_Default.[INAD_MES_3], 0) AS INAD_MES_3 \
      ,ISNULL(W_Default.[INAD_MES_4], 0) AS INAD_MES_4 \
      ,ISNULL(W_Default.[INAD_MES_5], 0) AS INAD_MES_5 \
      ,ISNULL(W_Default.[INAD_MES_6], 0) AS INAD_MES_6 \
      ,ISNULL(W_Default.[INAD_MES_7], 0) AS INAD_MES_7 \
      ,ISNULL(W_Default.[INAD_MES_8], 0) AS INAD_MES_8 \
      ,ISNULL(W_Default.[INAD_MES_9], 0) AS INAD_MES_9 \
      ,ISNULL(W_Default.[INAD_MES_10], 0) AS INAD_MES_10 \
      ,ISNULL(W_Default.[INAD_MES_11], 0) AS INAD_MES_11 \
      ,ISNULL(W_Default.[INAD_MES_12], 0) AS INAD_MES_12 \
      ,shape.STX AS X \
      ,shape.STY AS Y \
FROM \
      W_Inspection \
INNER JOIN \
      W_Consumer \
ON W_Inspection.INSTALACAO_PREPARED = W_Consumer.INSTALACAO_PREPARED \
INNER JOIN \
      W_Consumption \
ON ( \
      W_Inspection.INSTALACAO_PREPARED = W_Consumption.INSTALACAO_PREPARED \
      AND W_Inspection.ANOMES = W_Consumption.ANO_MES \
      ) \
LEFT JOIN \
      W_Default \
ON \
      ( \
      W_Inspection.INSTALACAO_PREPARED = W_Default.INSTALACAO_PREPARED \
      AND W_Inspection.ANOMES = W_Default.ANO_MES \
      ) \
LEFT JOIN \
      [BDGDEnel2019].[dbo].[ucbt] \
ON W_Consumer.INSTALACAO = [BDGDEnel2019].[dbo].[ucbt].cod_id \
ORDER BY \
    W_Inspection.ANOMES \
'
```

Carregamento dos dados

Dados das inspeções com dados de cadastro, consumo e inadimplência

```
df_inspection_raw = pd.read_sql(sql_inspection, conn)

df_inspection = df_inspection_raw.copy()
df_inspection
```

Dados dos setores censitários

```
gdf_sectors = geopandas.read_file(
    os.path.join('.', '13. IBGE', 'sp_setores_censitarios', '35SEE250GC_SIR.shp')
)

for c in [
    'TIPO'
    ,'NM_SUBDIST'
    ,'NM_DISTRIT'
    ,'NM_MUNICIP'
    ,'NM_MICRO'
    ,'NM_MESO'
    ,'NM_BAIRRO'
]:
    gdf_sectors[c] = gdf_sectors[c].apply(lambda x:
unidecode.unidecode(str(x).lower()))

gdf_sectors.loc[
    gdf_sectors['NM_BAIRRO'].isna()
    ,'NM_BAIRRO'
] = gdf_sectors.loc[
    gdf_sectors['NM_BAIRRO'].isna()
    ,'NM_DISTRIT'
]
```

Dados socioeconomicos dos setores

```
DICT_TYPE_2_FILES = {
    'Basico': {
        'files': [
                os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Capital', 'CSV', 'Basico_SP1.csv')
                ,os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Exceto_Capital', 'CSV', 'Basico_SP2.csv')
            ]
        ,'encoding': [
                'latin-1'
                ,'utf-8'
            ]
    }
    ,'Domicilio01': {
        'files': [
                os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Capital', 'CSV', 'Domicilio01_SP1.csv')
                ,os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Exceto_Capital', 'CSV', 'Domicilio01_SP2.csv')
            ]
        ,'encoding': [
                'utf-8'
                ,'utf-8'
            ]
```

```
    }
    ,'Pessoa03': {
        'files': [
                os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Capital', 'CSV', 'Pessoa03_SP1.csv')
                ,os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Exceto_Capital', 'CSV', 'Pessoa03_SP.csv')
            ]
        ,'encoding': [
                'utf-8'
                ,'utf-8'
            ]
    }
    ,'Pessoa01': {
        'files': [
                os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Capital', 'CSV', 'Pessoa01_SP1.csv')
                ,os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Exceto_Capital', 'CSV', 'Pessoa01_SP.csv')
            ]
        ,'encoding': [
                'utf-8'
                ,'utf-8'
            ]
    }
    ,'Pessoa13': {
        'files': [
                os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Capital', 'CSV', 'Pessoa13_SP1.csv')
                ,os.path.join('.', '13. IBGE', 'Base informaçoes setores2010
universo SP_Exceto_Capital', 'CSV', 'Pessoa13_SP.csv')
            ]
        ,'encoding': [
                'utf-8'
                ,'utf-8'
            ]
    }
}

dict_ibge = {}
for type in DICT_TYPE_2_FILES:
    df = None
    for file, encoding in zip(DICT_TYPE_2_FILES[type]['files'],
DICT_TYPE_2_FILES[type]['encoding']):
        print(file)
        print(encoding)
        print('\n')
        if df is None:
            df = pd.read_csv(file, encoding=encoding, sep=';', low_memory=False)
        else:
            df = pd.concat((df, pd.read_csv(file, encoding=encoding, sep=';',
low_memory=False)))
    dict_ibge[type] = df
```

Prepara e junta dados

Prepara dados de inspeções

```
for c in ['LONGITUDE', 'LATITUDE']:
    df_inspection.loc[
        (df_inspection[c].apply(lambda x: str(x).strip()) == '')
        |(df_inspection[c].apply(lambda x: -1 if x == '' else float(x)) == 0.0)
        ,c
    ] = 0

for c in [
    'EPBAIRRO'
    ,'EPMUNIC'
]:
    df_inspection[c] = df_inspection[c].apply(lambda x:
unidecode.unidecode(str(x).lower()))

df_inspection.loc[
    (df_inspection['LONGITUDE'] == 0)
    &(df_inspection['X'].notna())
    ,'LONGITUDE'
] = df_inspection.loc[
    (df_inspection['LONGITUDE'] == 0)
    &(df_inspection['X'].notna())
    ,'X'
]

df_inspection.loc[
    (df_inspection['LATITUDE'] == 0)
    &(df_inspection['Y'].notna())
    ,'LATITUDE'
] = df_inspection.loc[
    (df_inspection['LATITUDE'] == 0)
    &(df_inspection['Y'].notna())
    ,'Y'
]

gdf_inspection = geopandas.GeoDataFrame(
    df_inspection
    ,geometry=[
        Point(float(p[0]), float(p[1]))
        for p in np.column_stack((
            df_inspection['LONGITUDE']
            ,df_inspection['LATITUDE']
        ))
    ]
    ,crs='EPSG:4326'
)
```

Junta os dados pelas coordenadas

```
gdf_inspection = geopandas.sjoin(
    gdf_inspection
    ,gdf_sectors
    ,how='left'
    ,predicate='intersects'
)
gdf_inspection['rank_duplicates'] = gdf_inspection.groupby(
    ['Ordem', 'instalacao']
)['BASE'].rank(method='first')
```

```
gdf_inspection = gdf_inspection.loc[
    gdf_inspection['rank_duplicates'] == 1
].copy()
gdf_inspection
```

Junta aqueles que não foi possível pela coordena, com o primeiro setor que seja do mesmo bairro e cidade

```
gdf_inspection_not_found = gdf_inspection.loc[
    gdf_inspection['ID'].isna()
    ,[c for c in gdf_inspection if c not in gdf_sectors.columns or c ==
'geometry']
].merge(
    gdf_sectors.rename({'geometry': 'sector_geometry'}, axis=1)
    ,how='inner'
    ,left_on=['EPMUNIC', 'EPBAIRRO']
    ,right_on=['NM_MUNICIP','NM_BAIRRO']
)

gdf_inspection_not_found['LONGITUDE'] =
gdf_inspection_not_found['sector_geometry'].apply(lambda pol: pol.centroid.x)
gdf_inspection_not_found['LATITUDE'] =
gdf_inspection_not_found['sector_geometry'].apply(lambda pol: pol.centroid.y)

gdf_inspection_not_found['rank_duplicates'] = gdf_inspection_not_found.groupby(
    ['Ordem', 'instalacao']
)['BASE'].rank(method='first')

gdf_inspection_not_found = gdf_inspection_not_found.loc[
    gdf_inspection_not_found['rank_duplicates'] == 1
].copy()

gdf_inspection = pd.concat((
    gdf_inspection.loc[
        gdf_inspection['ID'].notna()
    ]
    ,gdf_inspection_not_found
))

gdf_inspection['CD_GEOCODI'] = gdf_inspection['CD_GEOCODI'].apply(lambda x:
int(x))

gdf_inspection.head()
```

Prepara dados do IBGE

```
VARS_IBGE = []

df_ibge = dict_ibge['Basico']

df_ibge['Média moradores por domicílio'] = df_ibge['V003'].apply(lambda x:
float(str(x).replace('.', '').replace(',','.')))
VARS_IBGE.append('Média moradores por domicílio')
df_ibge['Rendimento mensal médio dos responsáveis'] = df_ibge['V005'].apply(lambda
x: float(str(x).replace('.', '').replace(',','.')))
VARS_IBGE.append('Rendimento mensal médio dos responsáveis')
```

```
df_ibge = df_ibge[[c for c in df_ibge.columns if c in VARS_IBGE or c ==
'Cod_setor']].copy()

df_ibge = df_ibge.merge(
    dict_ibge['Domicilio01']
    ,how='left'
    ,left_on='Cod_setor'
    ,right_on='Cod_setor'
)

df_ibge['Razão de domicilios alugados'] = df_ibge['V008'].apply(
    lambda x: 0 if x == 'X' else int(x)
    )/df_ibge['V002'].apply(lambda x: 1 if x == 'X' else int(x))
VARS_IBGE.append('Razão de domicilios alugados')
df_ibge['Razão de domicilios com abastecimento de água'] = df_ibge['V012'].apply(
    lambda x: 0 if x == 'X' else int(x)
    )/df_ibge['V002'].apply(lambda x: 1 if x == 'X' else int(x))
VARS_IBGE.append('Razão de domicilios com abastecimento de água')
df_ibge['Razão de domicilios com coleta de esgoto'] = df_ibge['V017'].apply(
    lambda x: 0 if x == 'X' else int(x)
    )/df_ibge['V002'].apply(lambda x: 1 if x == 'X' else int(x))
VARS_IBGE.append('Razão de domicilios com coleta de esgoto')
df_ibge['Razão de domicilios com coleta de lixo'] = df_ibge['V035'].apply(
    lambda x: 0 if x == 'X' else int(x)
    )/df_ibge['V002'].apply(lambda x: 1 if x == 'X' else int(x))
VARS_IBGE.append('Razão de domicilios com coleta de lixo')

df_ibge = df_ibge[[c for c in df_ibge.columns if c in VARS_IBGE or c ==
'Cod_setor']].copy()

df_pessoa13 = dict_ibge['Pessoa13']
for c in df_pessoa13.columns:
    df_pessoa13[c] = df_pessoa13[c].apply(lambda x: 0 if x == 'X' else x)
df_ibge = df_ibge.merge(
    df_pessoa13
    ,how='left'
    ,left_on='Cod_setor'
    ,right_on='Cod_setor'
)
df_ibge['Pessoas com 10 ou mais anos'] = df_ibge.apply(
    lambda row:     int(row['V044']) + int(row['V045']) + int(row['V046']) +
int(row['V047']) + int(row['V048']) + int(row['V049']) + int(row['V050']) +
int(row['V051']) + int(row['V052']) + int(row['V053'])
    + int(row['V054']) + int(row['V055']) + int(row['V056']) + int(row['V057']) +
int(row['V058']) + int(row['V059']) + int(row['V060']) + int(row['V061']) +
int(row['V062']) + int(row['V063'])
    + int(row['V064']) + int(row['V065']) + int(row['V066']) + int(row['V067']) +
int(row['V068']) + int(row['V069']) + int(row['V070']) + int(row['V071']) +
int(row['V072']) + int(row['V073'])
    + int(row['V074']) + int(row['V075']) + int(row['V076']) + int(row['V077']) +
int(row['V078']) + int(row['V079']) + int(row['V080']) + int(row['V081']) +
int(row['V082']) + int(row['V083'])
    + int(row['V084']) + int(row['V085']) + int(row['V086']) + int(row['V087']) +
int(row['V088']) + int(row['V089']) + int(row['V090']) + int(row['V091']) +
int(row['V092']) + int(row['V093'])
    + int(row['V094']) + int(row['V095']) + int(row['V096']) + int(row['V097']) +
int(row['V098']) + int(row['V099']) + int(row['V100']) + int(row['V101']) +
int(row['V102']) + int(row['V103'])
```

```
    + int(row['V104']) + int(row['V105']) + int(row['V106']) + int(row['V107']) +
int(row['V108']) + int(row['V109']) + int(row['V110']) + int(row['V111']) +
int(row['V112']) + int(row['V113'])
    + int(row['V114']) + int(row['V115']) + int(row['V116']) + int(row['V117']) +
int(row['V118']) + int(row['V119']) + int(row['V120']) + int(row['V121']) +
int(row['V122']) + int(row['V123'])
    + int(row['V124']) + int(row['V125']) + int(row['V126']) + int(row['V127']) +
int(row['V128']) + int(row['V129']) + int(row['V130']) + int(row['V131']) +
int(row['V132']) + int(row['V133'])
    + int(row['V134'])
    ,axis=1
)
df_ibge = df_ibge[[c for c in df_ibge.columns if c in VARS_IBGE or c ==
'Cod_setor' or c == 'Pessoas com 10 ou mais anos']].copy()


df_pessoa01 = dict_ibge['Pessoa01']
for c in df_pessoa01.columns:
    df_pessoa01[c] = df_pessoa01[c].apply(lambda x: 0 if x == 'X' else x)
df_ibge = df_ibge.merge(
    df_pessoa01
    ,how='left'
    ,left_on='Cod_setor'
    ,right_on='Cod_setor'
)

df_ibge['Pessoas alfabetizadas com 10 ou mais anos'] = df_ibge.apply(
    lambda row:          int(row['V007']) + int(row['V008']) + int(row['V009'])
+int(row['V010']) + int(row['V011']) + int(row['V012']) + int(row['V013']) +
int(row['V014'])
         + int(row['V015']) + int(row['V016']) + int(row['V017']) +
int(row['V018']) +int(row['V019']) + int(row['V020']) + int(row['V021']) +
int(row['V022']) + int(row['V023'])
         + int(row['V024']) + int(row['V025']) + int(row['V026']) +
int(row['V027']) +int(row['V028']) + int(row['V029']) + int(row['V030']) +
int(row['V031']) + int(row['V032'])
         + int(row['V033']) + int(row['V034']) + int(row['V035']) +
int(row['V036']) +int(row['V037']) + int(row['V038']) + int(row['V039']) +
int(row['V040']) + int(row['V041'])
         + int(row['V042']) + int(row['V043']) + int(row['V044']) +
int(row['V045']) +int(row['V046']) + int(row['V047']) + int(row['V048']) +
int(row['V049']) + int(row['V050'])
         + int(row['V051']) + int(row['V052']) + int(row['V053']) +
int(row['V054']) +int(row['V055']) + int(row['V056']) + int(row['V057']) +
int(row['V058']) + int(row['V059'])
         + int(row['V060']) + int(row['V061']) + int(row['V062']) +
int(row['V063']) +int(row['V064']) + int(row['V065']) + int(row['V066']) +
int(row['V067']) + int(row['V068'])
         + int(row['V069']) + int(row['V070']) + int(row['V071']) +
int(row['V072']) +int(row['V073']) + int(row['V074']) + int(row['V075']) +
int(row['V076']) + int(row['V077'])
    ,axis=1
)

df_ibge['Razão de pessoas alfabetizadas com 10 ou mais anos'] = df_ibge['Pessoas
alfabetizadas com 10 ou mais anos']/df_ibge['Pessoas com 10 ou mais anos']
VARS_IBGE.append('Razão de pessoas alfabetizadas com 10 ou mais anos')
```

```
df_ibge = df_ibge[[c for c in df_ibge.columns if c in VARS_IBGE or c ==
'Cod_setor']].copy()
```

Junta com inspeções

```
gdf_inspection = gdf_inspection.merge(
    df_ibge
    ,how='left'
    ,left_on='CD_GEOCODI'
    ,right_on='Cod_setor'
)
```

Cria novas variáveis

Variáveis numéricas

```
VARS_NUM = []

gdf_inspection['Média consumo'] = gdf_inspection.apply(
    lambda row: (row['CONSUMO_MES_1'] + row['CONSUMO_MES_2'] +
row['CONSUMO_MES_3'] + row['CONSUMO_MES_4'] + row['CONSUMO_MES_5'] +
row['CONSUMO_MES_6']
    + row['CONSUMO_MES_7'] + row['CONSUMO_MES_8'] + row['CONSUMO_MES_9'] +
row['CONSUMO_MES_10'] + row['CONSUMO_MES_11'] + row['CONSUMO_MES_12'])/12
    ,axis=1
)
VARS_NUM.append('Média consumo')

gdf_inspection['Quantil 0.25'] = gdf_inspection.apply(
    lambda row: np.quantile([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']], 0.25)
    ,axis=1
)
VARS_NUM.append('Quantil 0.25')

gdf_inspection['Quantil 0.75'] = gdf_inspection.apply(
    lambda row: np.quantile([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']], 0.75)
    ,axis=1
)
VARS_NUM.append('Quantil 0.75')

gdf_inspection['Quantil 0.10'] = gdf_inspection.apply(
    lambda row: np.quantile([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']], 0.10)
    ,axis=1
)
```

```
VARS_NUM.append('Quantil 0.10')

gdf_inspection['Quantil 0.90'] = gdf_inspection.apply(
    lambda row: np.quantile([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']], 0.90)
    ,axis=1
)
VARS_NUM.append('Quantil 0.90')

gdf_inspection['Inadimplência nos últimos 3 meses'] = gdf_inspection.apply(
    lambda row: int(row['INAD_MES_1']) + int(row['INAD_MES_2']) +
int(row['INAD_MES_3'])
    ,axis=1
)
gdf_inspection['Inadimplência nos últimos 3 meses'] =
gdf_inspection['Inadimplência nos últimos 3 meses'].apply(lambda x: 0 if x == 0
else x/x)
VARS_NUM.append('Inadimplência nos últimos 3 meses')

gdf_inspection['Inadimplência nos últimos 6 meses'] = gdf_inspection.apply(
    lambda row: int(row['INAD_MES_1']) + int(row['INAD_MES_2']) +
int(row['INAD_MES_3']) + int(row['INAD_MES_4']) + int(row['INAD_MES_5']) +
int(row['INAD_MES_6'])
    ,axis=1
)
gdf_inspection['Inadimplência nos últimos 6 meses'] =
gdf_inspection['Inadimplência nos últimos 6 meses'].apply(lambda x: 0 if x == 0
else x/x)
VARS_NUM.append('Inadimplência nos últimos 6 meses')

gdf_inspection['Inadimplência nos últimos 12 meses'] = gdf_inspection.apply(
    lambda row: int(row['INAD_MES_1']) + int(row['INAD_MES_2']) +
int(row['INAD_MES_3']) + int(row['INAD_MES_4']) + int(row['INAD_MES_5']) +
int(row['INAD_MES_6'])
        + int(row['INAD_MES_7']) + int(row['INAD_MES_8']) + int(row['INAD_MES_9'])
+ int(row['INAD_MES_10']) + int(row['INAD_MES_11']) + int(row['INAD_MES_12'])
    ,axis=1
)
gdf_inspection['Inadimplência nos últimos 12 meses'] =
gdf_inspection['Inadimplência nos últimos 12 meses'].apply(lambda x: 0 if x == 0
else x/x)
VARS_NUM.append('Inadimplência nos últimos 12 meses')

VARS_NUM_OTHERS = []
gdf_inspection['Máximo degrau positivo'] = gdf_inspection.apply(
    lambda row: np.max(np.diff([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']]))
    ,axis=1
)
VARS_NUM_OTHERS.append('Máximo degrau positivo')

gdf_inspection['Máximo degrau negativo'] = gdf_inspection.apply(
    lambda row: np.abs(np.min(np.diff([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
```

```
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']]))
    ,axis=1
)
VARS_NUM_OTHERS.append('Máximo degrau negativo')

gdf_inspection['Amplitude de consumo'] = gdf_inspection.apply(
    lambda row: np.max([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'],
row['CONSUMO_MES_3'], row['CONSUMO_MES_4'], row['CONSUMO_MES_5'],
row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']]) -
    np.min([row['CONSUMO_MES_1'], row['CONSUMO_MES_2'], row['CONSUMO_MES_3'],
row['CONSUMO_MES_4'], row['CONSUMO_MES_5'], row['CONSUMO_MES_6'],
        row['CONSUMO_MES_7'], row['CONSUMO_MES_8'], row['CONSUMO_MES_9'],
row['CONSUMO_MES_10'], row['CONSUMO_MES_11'], row['CONSUMO_MES_12']])
    ,axis=1
)
VARS_NUM_OTHERS.append('Amplitude de consumo')


gdf_inspection['Longitude Arredondada'] = gdf_inspection['LONGITUDE'].apply(lambda
x: round(float(x)*100)/100)
gdf_inspection['Latitude Arredondada'] = gdf_inspection['LATITUDE'].apply(lambda
x: round(float(x)*100)/100)
VARS_NUM_OTHERS.append('Longitude Arredondada')
VARS_NUM_OTHERS.append('Latitude Arredondada')

gdf_inspection['Média de consumo da região'] = gdf_inspection[
    ['Média consumo', 'Longitude Arredondada', 'Latitude Arredondada']
].groupby(
    ['Longitude Arredondada', 'Latitude Arredondada']
).transform(np.mean)
gdf_inspection['Razão de média de consumo com relação a região'] =
gdf_inspection['Média consumo']/gdf_inspection['Média de consumo da região']
VARS_NUM_OTHERS.append('Razão de média de consumo com relação a região')
```

Variáveis categóricas

```
VARS_CAT = []

dict_normalize_category = {
    'Comercial': 'Comercial'
    ,'Comercial-AdministraÃ§Ã£o condominial: IP/': 'Comercial'
    ,'Comercial-AssociaÃ§Ã£o e entidades filantr': 'Comercial'
    ,'Comercial-B Optante': 'Comercial'
    ,'Comercial-IluminaÃ§Ã£o em rodovia;': 'Comercial'
    ,'Comercial-Outros ServiÃ§os e Atividades': 'Comercial'
    ,'Comercial-SemÃ¡foros, radares e cÃ¢meras d': 'Comercial'
    ,'Comercial-Serv ComunicaÃ§Ãµes e Telecom': 'Comercial'
    ,'Comercial-ServiÃ§os de Transporte': 'Comercial'
    ,'Comercial-Templos religiosos;': 'Comercial'
    ,'Consumo PrÃ³prio-EscritÃ³rios': 'Consumo próprio'
    ,'Consumo PrÃ³prio-Interno (EstaÃ§Ã£o,Usinas)': 'Consumo próprio'
    ,'IluminaÃ§Ã£o PÃºblica': 'Iluminação pública'
    ,'Industrial': 'Industrial'
```

```
    ,'Industrial-B Optante': 'Industrial'
    ,'Poder PÃºblico-Estadual': 'Poder público'
    ,'Poder PÃºblico-Federal': 'Poder público'
    ,'Poder PÃºblico-Municipal': 'Poder público'
    ,'Residencial-B Optante': 'Residencial-B Optante'
    ,'Residencial-Baixa Renda': 'Residencial-Baixa Renda'
    ,'Residencial-Baixa Renda BPC': 'Residencial-Baixa Renda BPC'
    ,'Residencial-Plena': 'Residencial-Plena'
    ,'Rural-AgropecuÃ¡ria': 'Rural'
    ,'Rural-IndÃºstria Rural': 'Rural'
    ,'Rural-Residencial Rural': 'Rural'
    ,'ServiÃ§o PÃºblico-TraÃ§Ã£o ElÃ©trica': 'Serviço Público'
    ,'ServiÃ§o PÃºblico-Ãgua Esgoto e Saneamento': 'Serviço Público'
}
gdf_inspection['EPTMPAREA_TEXT'] =
gdf_inspection['EPTMPAREA_TEXT'].map(dict_normalize_category)

dummies = pd.get_dummies(
    gdf_inspection['EPTMPAREA_TEXT']
    ,prefix='Categoria da Instalacao'
)
VARS_CAT.extend(
    dummies.columns.to_list()
)

gdf_inspection = gdf_inspection.merge(
    dummies
    ,how='left'
    ,left_index=True
    ,right_index=True
)

dict_normalize_phases = {
    '': 'Bifásico'
    ,'BIFÃSICO': 'Bifásico'
    ,'MONOFÃSICO': 'Monofásico'
    ,'TRIFÃSICO': 'Trifásico'
}
gdf_inspection['EPNUMFAS_TEXT'] =
gdf_inspection['EPNUMFAS_TEXT'].map(dict_normalize_phases)

dummies = pd.get_dummies(
    gdf_inspection['EPNUMFAS_TEXT']
    ,prefix='Número de fases'
)
VARS_CAT.extend(
    dummies.columns.to_list()
)

gdf_inspection = gdf_inspection.merge(
    dummies
    ,how='left'
    ,left_index=True
    ,right_index=True
)
```

Modelagem

Divide entre treino e teste

```
df_train = gdf_inspection.loc[gdf_inspection['ANOMES'].astype(int) <=
train_test_split_date]
df_test = gdf_inspection.loc[gdf_inspection['ANOMES'].astype(int) >
train_test_split_date]

df_train.shape

df_train['HIT_F'].value_counts()/df_train.shape[0]

df_test.shape
```

Treina modelos sem amostragem

```
dict_data_group = {
    'DetecçãoDeFraude': {
        ClassifierModelTournament.KEY_DICT_DATA_DF: df_train,
        ClassifierModelTournament.KEY_DICT_DATA_PREDICTORS: [
            VARS_NUM
            ,VARS_NUM + VARS_NUM_OTHERS
            ,VARS_NUM + VARS_NUM_OTHERS + VARS_CAT
            ,VARS_NUM + VARS_NUM_OTHERS + VARS_CAT + VARS_IBGE
        ]},
}

# Run the tournament
tournament = ClassifierModelTournament(
    dict_data_group,
    ['HIT_F'],
    HPClassifierOptimizer.get_available_models(),
    transformers_list=[Transformer.TransformerEnum.NoTransform],
    experiment_prefix='DetecçãoDeFraude',
    parallel=False,
    parallel_jobs=2,
    hp_metric_opt='f1_1.0',
    hp_max_evals=60
)
tournament.init_tournament()
tournament.save_results(suffix='-' + str(datetime.datetime.now()).replace(':', '-
'))
```

Treina modelos com subamostragem

```
df_train_undersamp, _ = RandomUnderSampler(random_state=42).fit_resample(df_train,
df_train['HIT_F'])
df_train_undersamp = pd.DataFrame(df_train_undersamp, columns=df_train.columns)
df_train_undersamp['HIT_F'] = df_train_undersamp['HIT_F'].astype(float)
display.display(df_train_undersamp['HIT_F'].mean())
display.display(df_train_undersamp.shape[0])

dict_data_group_undersamp = {
    'DetecçãoDeFraudeSubAmostragem': {
        ClassifierModelTournament.KEY_DICT_DATA_DF: df_train_undersamp,
        ClassifierModelTournament.KEY_DICT_DATA_PREDICTORS: [
            VARS_NUM
```

```
            ,VARS_NUM + VARS_NUM_OTHERS
            ,VARS_NUM + VARS_NUM_OTHERS + VARS_CAT
            ,VARS_NUM + VARS_NUM_OTHERS + VARS_CAT + VARS_IBGE
        ]},
}

# Run the tournament
tournament_undersamp = ClassifierModelTournament(
    dict_data_group_undersamp,
    ['HIT_F'],
    HPClassifierOptimizer.get_available_models(),
    transformers_list=[Transformer.TransformerEnum.NoTransform],
    experiment_prefix='DetecçãoDeFraudeSubAmostragem',
    parallel=False,
    parallel_jobs=2,
    hp_metric_opt='f1_1.0',
    hp_max_evals=60
)
tournament_undersamp.init_tournament()
tournament_undersamp.save_results(suffix='-' +
str(datetime.datetime.now()).replace(':', '-'))
```

Testa o melhor modelo

Melhor modelo foi o XGBoost com undersampling usando todas as variáveis

```
df_test_prob = df_test.copy()

df_result_undersamp = pd.read_pickle('DetecçãoDeFraudeSubAmostragem-ModelScores-
2022-04-25 23-28-40.599128.df')

model_enum = Model.ClassifierModelEnum.XGBClassifier
predictors = VARS_NUM + VARS_NUM_OTHERS + VARS_CAT + VARS_IBGE

row_best_model = df_result_undersamp.loc[
    (df_result_undersamp['ModelName'] == model_enum)
    &(df_result_undersamp['Predictors'].apply(lambda x: tuple(x)) ==
tuple(predictors))
]
best_model_trained = row_best_model['TrainedModel'].iloc[0]
best_model_params = row_best_model['BestParams'].iloc[0]
best_model_predictors = row_best_model['Predictors'].iloc[0]

best_model_retrained = HPClassifierOptimizer.MODELS[model_enum]()
best_model_retrained.set_params(**best_model_params)
best_model_retrained.fit(df_train_undersamp[best_model_predictors].values,
df_train_undersamp['HIT_F'].values)
display.display(best_model_retrained.score(df_train_undersamp[best_model_predictor
s].values, df_train_undersamp['HIT_F'].values))
display.display(best_model_trained.score(df_train_undersamp[best_model_predictors]
.values, df_train_undersamp['HIT_F'].values))

precision_recall_fscore_support(df_train_undersamp['HIT_F'].values,
best_model_retrained.predict(df_train_undersamp[best_model_predictors].values))

df_test_prob['HIT_F'].value_counts()

precision_recall_fscore_support(df_test_prob['HIT_F'].values,
best_model_retrained.predict(df_test_prob[best_model_predictors].values))
```

```
confusion_matrix(df_test_prob['HIT_F'].values,
best_model_retrained.predict(df_test_prob[best_model_predictors].values))

df_test_prob['HIT_F'].value_counts()[1.0]/df_test_prob['HIT_F'].shape[0]

np.sum(best_model_retrained.predict_proba(df_test_prob[best_model_predictors].valu
es)[:,1] >= 0.5)

if not
np.sum(best_model_retrained.predict(df_test_prob[best_model_predictors].values) ==
1) == \

np.sum(best_model_retrained.predict_proba(df_test_prob[best_model_predictors].valu
es)[:,1] >= 0.5):
    raise Exception

df_test_prob['predict_proba'] =
best_model_retrained.predict_proba(df_test_prob[best_model_predictors].values)[:,1
]
df_test_prob.sort_values(by='predict_proba', ascending=False, inplace=True)
df_test_prob.reset_index(inplace=True, drop=True)
df_test_prob
```

Precisão nas inspeções

```
df_test_precision = df_test_prob.loc[df_test_prob.index <
df_test_prob.shape[0]/2].copy()
display.display(df_test_precision['HIT_F'].value_counts())
df_test_precision['HIT_F'].value_counts().loc[1.0]/df_test_precision['HIT_F'].shap
e[0]

df_test_precision.shape

df_test_precision['precision_group'] = np.nan
df_test_precision.reset_index(drop=True)
n_groups = 10
group_len = math.floor(df_test_precision.shape[0]/n_groups)
display.display(group_len)

for i in range(0, n_groups):
    idx_init = i*group_len
    idx_final = (i+1)*group_len - 1
    if i == n_groups - 1:
        idx_final = df_test_precision.shape[0] - 1
    df_test_precision.loc[
        idx_init:idx_final
        ,'precision_group'
    ] = i + 1

def calc_precision_in_group(_df):
    return _df.value_counts()[1.0]/_df.shape[0]

df_test_precision.reset_index()[
    ['HIT_F', 'precision_group']
].groupby(
    'precision_group'
).agg(
    calc_precision_in_group
)
```

Apêndice B - Módulo Python adicional para treinamento dos modelos de *machine learning*

Esse módulo se chama models.py e é utilizado pelo *notebook* Python que realiza o treinamento dos modelos de *machine learning*. Esse módulo também se encontra no link em (PULZ, 2022) – arquivo “models.py”.

```
# Standard packages
from typing import Dict, List, Type, Union
from enum import Enum
import datetime
import logging
from concurrent.futures import ThreadPoolExecutor, wait as executor_wait

import pandas as pd
import numpy as np

# Optimization package
from hyperopt import tpe, hp, fmin, space_eval, Trials

# Sklearn general modules
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer, StandardScaler,
MinMaxScaler
from sklearn.metrics import make_scorer, mean_squared_error, mean_absolute_error,
f1_score, precision_score, \
    recall_score, accuracy_score, brier_score_loss, log_loss, jaccard_score,
roc_auc_score
from sklearn.model_selection import KFold, StratifiedKFold, cross_validate

# Regressors
from sklearn.linear_model import LogisticRegression

# Classifiers
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier,
AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.svm import SVC
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from xgboost import XGBClassifier


class Model:

    class ClassifierModelEnum(Enum):
        LogisticRegression = LogisticRegression
        RandomForestClassifier = RandomForestClassifier
        ExtraTreesClassifier = ExtraTreesClassifier
        AdaBoostClassifier = AdaBoostClassifier
        KNeighborsClassifier = KNeighborsClassifier
        RidgeClassifier = RidgeClassifier
        SVC = SVC
```

```
        LGBMClassifier = LGBMClassifier
        CatBoostClassifier = CatBoostClassifier
        XGBClassifier = XGBClassifier


class Metric:

    @staticmethod
    def mean_absolute_percentage_error(y_true, y_pred):
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

    @staticmethod
    def rmse(y_true, y_pred):
        mse = mean_squared_error(y_true, y_pred)
        return np.sqrt(mse)

    @staticmethod
    def _bottom_up_merge_sort(v, n):
        v_aux = v.copy()
        lost_points = 0
        # Each 1-element run in v is already 'sorted'.
        # Make successively longer sorted runs of length 2, 4, 8, 16... until
whole array is sorted.
        width = 1
        while width < n:
            # Array v is full of runs of length width.
            i = 0
            while i < n:
                # Merge two runs: v[i:i+width-1] and v[i+width:i+2*width-1] to
v_aux[]
                # or copy v[i:n-1] to v_aux[] ( if(i+width >= n) )
                lost_points += Metric._bottom_up_merge(v, i, min(i + width, n),
min(i + 2 * width, n), v_aux)
                i += 2 * width
            # Now work array v_aux is full of runs of length 2*width.
            # Copy array v_aux to array v for next iteration.
            # A more efficient implementation would swap the roles of v and v_aux.
            v, v_aux = v_aux, v
            # Now array v is full of runs of length 2*width.
            width = 2 * width
        return v, n * (n - 1) / 2 - lost_points

    @staticmethod
    def _bottom_up_merge(v, i_left, i_right, i_end, v_aux):
        # Left run is v[iLeft :iRight-1].
        # Right run is v[iRight:iEnd-1].
        lost_points = 0
        i = i_left
        j = i_right
        # While there are elements in the left or right runs...
        for k in range(i_left, i_end):
            # If left run head exists and is <= existing right run head.
            if i < i_right and (j >= i_end or v[i] <= v[j]):
                v_aux[k] = v[i]
                i += 1
            else:
                if i < i_right and j < i_end:
                    lost_points += j - k
```

```
                v_aux[k] = v[j]
                j += 1
        return lost_points

    @staticmethod
    def sorting_score(y_true, y_pred):
        y_true = list(y_true)
        y_pred = list(y_pred)

        n = len(y_true)
        max_points = n * (n - 1) / 2.0

        y_true_sorted_idxs = sorted(range(n), key=y_true.__getitem__)
        y_pred_sorted_by_true = [y_pred[i] for i in y_true_sorted_idxs]

        y_sorted, points = Metric._bottom_up_merge_sort(y_pred_sorted_by_true, n)

        return points / max_points

    @staticmethod
    def pow_sort_score(sort_score):
        if sort_score <= 0.1:
            sort_score = 0.1
        return 10 ** (1.0 / sort_score - 1)

    @staticmethod
    def mse_and_sorting_score(y_true, y_pred):
        mse = mean_squared_error(y_true, y_pred)

        return Metric.pow_sort_score(Metric.sorting_score(y_true, y_pred)) * mse


class Transformer:

    class TransformerEnum(Enum):
        NoTransform = 'NoTransform'
        StandardTransform = 'StandardTransform'
        MinMaxTransform = 'MinMaxTransform'
        Log10Transform = 'Log10Transform'
        Log10XPlus1Transform = 'Log10XPlus1Transform'

    @staticmethod
    def get_transformer_func(transformer_enum: 'Transformer.TransformerEnum'):
        if transformer_enum == Transformer.TransformerEnum.NoTransform:
            transformer_func = Transformer._no_transformer
        elif transformer_enum == Transformer.TransformerEnum.StandardTransform:
            transformer_func = StandardScaler
        elif transformer_enum == Transformer.TransformerEnum.MinMaxTransform:
            transformer_func = MinMaxScaler
        elif transformer_enum == Transformer.TransformerEnum.Log10Transform:
            transformer_func = Transformer._log10_transformer
        elif transformer_enum == Transformer.TransformerEnum.Log10XPlus1Transform:
            transformer_func = Transformer._log10_xplus1_transformer
        else:
            raise Exception(f'Transformer {transformer_enum} is not implemented')
        return transformer_func
```

```
    @staticmethod
    def _no_transformer():
        return FunctionTransformer(Transformer._x)

    @staticmethod
    def _log10_transformer():
        return FunctionTransformer(np.log10)

    @staticmethod
    def _log10_xplus1_transformer():
        return FunctionTransformer(Transformer._log10_xplus1)

    @staticmethod
    def _x(x):
        return x

    @staticmethod
    def _log10_xplus1(x):
        return np.log10(x + 1)


class _HPOptimizer:

    MODELS = None

    def __init__(self, n_features, x, y, hp_tuning: bool = True, n_folds=10,
experiment_name: str = 'default',
                 cv_jobs=4, hp_max_evals=60, parallel=True, parallel_jobs=4,
hp_metric_opt=None, params=None):
        self._n_features = n_features
        self._x = x
        self._y = y
        self._hp_tuning = hp_tuning
        self._kfold = self._get_cross_validator(n_folds)
        self._experiment_name = experiment_name
        self._cv_jobs = cv_jobs
        self._hp_max_evals = hp_max_evals
        self._parallel = parallel
        self._parallel_jobs = parallel_jobs
        if params is None:
            self._models_hp = self._create_opt_params()
        else:
            self._models_hp = params

        # Test options and evaluation metric
        self._scoring = self._make_scoring()
        # Metric for hyper parameters optimization
        self._hp_opt_objective_metric =
self._get_hp_opt_objective_metric(hp_metric_opt)
        # List of pipeline of each model
        self._pipelines = None

    def _get_hp(self, model):
        return self._models_hp.get(type(model), None)

    @staticmethod
    def _get_cross_validator(n_folds):
```

```
        raise NotImplementedError

    def _create_opt_params(self):
        raise NotImplementedError

    def run_models(self):
        # Check if the pipelines was built
        if self._pipelines is None:
            raise Exception('The pipelines must be built before running the
models.')

        results_list = []
        names_list = []
        best_params_list = []
        trained_models_list = []
        trained_models_score_list = []

        if self._parallel:
            with ThreadPoolExecutor(self._parallel_jobs,
thread_name_prefix='ModelsLoop') as executor:
                futures = [executor.submit(self._run_model, name, model) for name,
model in self._pipelines]
                executor_wait(futures)
                for future in futures:
                    future_result = future.result()
                    results = future_result[0]
                    name = future_result[1]
                    best_params = future_result[2]
                    trained_model = future_result[3]
                    trained_model_score = future_result[4]
                    results_list.append(results)
                    names_list.append(name)
                    best_params_list.append(best_params)
                    trained_models_list.append(trained_model)
                    trained_models_score_list.append(trained_model_score)
        else:
            for name, model in self._pipelines:
                results, name, best_params, trained_model, trained_model_score =
self._run_model(name, model)
                results_list.append(results)
                names_list.append(name)
                best_params_list.append(best_params)
                trained_models_list.append(trained_model)
                trained_models_score_list.append(trained_model_score)

        self._pipelines = None
        results_df = self._results_parser(names_list, results_list)
        return results_df, best_params_list, trained_models_list,
trained_models_score_list

    def _run_model(self, name, model):
        print(f'RUNNING {name}...')
        best_params = None
        if self._hp_tuning:
            space = self._get_hp(model[1])
            logging.info(space)
            args = {
                'name': name,
```

```
                'param': space,
                'data': [self._x, self._y],
                'kfold': self._kfold,
                'scoring': {'objective': self._hp_opt_objective_metric},
                'model': model,
                'experiment_name': self._experiment_name,
                'n_jobs': self._cv_jobs
            }
            if space is not None:
                best = fmin(self._get_hp_opt_objective_func(), args,
algo=tpe.suggest,
                            rstate=np.random.default_rng(3500),
max_evals=self._hp_max_evals, trials=Trials())
                s_eval = space_eval(space, best)
                logging.info(s_eval)
                args = s_eval
                model[1].set_params(**args)
                best_params = args
            else:
                print(f'**Hyperparameters optimization does not exist or is not
implemented for {name} **')
                best_params = None
        cv_result = cross_validate(
            model, self._x.values, self._y.values, cv=self._kfold,
scoring=self._scoring, n_jobs=self._cv_jobs,
            return_train_score=True)

        cv_result['name'] = name
        trained_model = model.fit(self._x.values, self._y.values)
        trained_model_score = trained_model.score(self._x.values, self._y.values)
        print(name, 'ENDED')
        return cv_result, name, best_params, trained_model, trained_model_score

    def _get_hp_opt_objective_metric(self, hp_metric_opt):
        raise NotImplementedError

    def _get_hp_opt_objective_func(self):
        loss_func = self._loss_func

        def _hp_opt_objective_func(args):
            x, y = args['data'][0], args['data'][1]
            # name = args['name']
            param = args['param']
            kfold = args['kfold']
            scoring = args['scoring']
            model = args['model']
            n_jobs = args['n_jobs']
            model[1].set_params(**param)

            cv_results = cross_validate(
                model, x.values, y.values, cv=kfold, scoring=scoring,
n_jobs=n_jobs)

            kfold_metric = cv_results.get('test_objective', '')
            loss = loss_func(kfold_metric.mean())

            return loss
        return _hp_opt_objective_func
```

```
    @staticmethod
    def _loss_func(metric):
        raise NotImplementedError

    @staticmethod
    def _make_scoring():
        raise NotImplementedError

    @staticmethod
    def _results_parser(names, results):
        results_parse = {'ModelName': names}
        for result_key in results[0].keys():
            if result_key != 'name' and result_key[-15:] != 'ytrue_and_ypred':
                results_parse[result_key + '_mean'] =
[model_result[result_key].mean() for model_result in results]
                results_parse[result_key + '_std'] =
[model_result[result_key].std() for model_result in results]
            elif result_key[-15:] == 'ytrue_and_ypred':
                list_result = []
                for model_result in results:
                    list_result.append([k_fold_result for k_fold_result in
model_result[result_key]])
                results_parse[result_key] = list_result
        return pd.DataFrame(results_parse)

    def build_pipelines_for_models(
            self, models_enums: List[Model.ClassifierModelEnum],
            models_init_args: Dict[Model.ClassifierModelEnum, Dict] = None,
            transformer_enum: Transformer.TransformerEnum =
Transformer.TransformerEnum.NoTransform) -> None:
        transformer_func = self._get_transformer_func(transformer_enum)
        models = self._get_available_models()
        pipelines = []
        if len(models_enums) == 0:
            raise Exception('No model was chosen')
        for model_enum in models_enums:
            if model_enum not in models:
                raise Exception(
                    f'The model {model_enum} is not mapped in the class, check the
name or build the code for it.'
                    f'\nAvailable models: {list(models.keys())}\n\n')
            if models_init_args is not None and model_enum in models_init_args:
                model = models[model_enum](**models_init_args[model_enum])
                print(f'Done {models_init_args[model_enum]}')
                del models_init_args[model_enum]
            else:
                model = models[model_enum]()
            pipelines.append((
                model_enum,
                Pipeline([(f'Scaler{model_enum}', transformer_func()),
(f'Model{model_enum}', model)])
            ))
        if models_init_args:
            raise Exception(
                f'Some model arguments were not used:
{list(models_init_args.keys())}.'
                f'\nRemove them or check their names.')
```

```
        self._pipelines = pipelines

    @staticmethod
    def _get_transformer_func(transformer_enum: Transformer.TransformerEnum):
        return Transformer.get_transformer_func(transformer_enum)

    def _get_available_models(self):
        if self.MODELS is None:
            raise NotImplementedError
        else:
            return self.MODELS

    @staticmethod
    def get_available_models():
        raise NotImplementedError


class HPClassifierOptimizer(_HPOptimizer):
    MODELS = {
        Model.ClassifierModelEnum.LogisticRegression: LogisticRegression,
        Model.ClassifierModelEnum.RandomForestClassifier: RandomForestClassifier,
        Model.ClassifierModelEnum.ExtraTreesClassifier: ExtraTreesClassifier,
        # Model.ClassifierModelEnum.AdaBoostClassifier: AdaBoostClassifier, # Poor
performance
        # Model.ClassifierModelEnum.KNeighborsClassifier: KNeighborsClassifier, #
Not working
        # Model.ClassifierModelEnum.RidgeClassifier: RidgeClassifier, # Has no
probability prediction
        # Model.ClassifierModelEnum.SVC: SVC, # Has no probability prediction
        Model.ClassifierModelEnum.LGBMClassifier: LGBMClassifier,
        # Model.ClassifierModelEnum.CatBoostClassifier: CatBoostClassifier,
        Model.ClassifierModelEnum.XGBClassifier: XGBClassifier,
    }

    def __init__(self, n_features, x, y, hp_tuning: bool = True, n_folds=10,
experiment_name: str = 'default',
                 cv_jobs=4, hp_max_evals=60, parallel=True, parallel_jobs=4,
hp_metric_opt=None, params=None):
        super().__init__(
            n_features, x, y, hp_tuning=hp_tuning, n_folds=n_folds,
experiment_name=experiment_name, cv_jobs=cv_jobs,
            hp_max_evals=hp_max_evals, parallel=parallel,
parallel_jobs=parallel_jobs, hp_metric_opt=hp_metric_opt,
            params=params)

    def _create_opt_params(self):
        default_params = {
            LogisticRegression: {
                # 'penalty': hp.choice('penalty', ['l1', 'l2', 'elasticnet']),
                'tol': hp.uniform('tol', 1e-5, 1e-4),
                'C': hp.uniform('C', 1e-2, 2),
                'max_iter': hp.choice('max_iter', [10000]),
                'random_state': hp.choice('seed', [27])
            },
            KNeighborsClassifier: {
                'leaf_size ': hp.choice('leaf_size ', range(3, 40)),
                'n_neighbors ':  hp.choice('n_neighbors ', range(1, 15))
            },
```

```
            RidgeClassifier: {
                'alpha': hp.uniform('alpha', 0.001, 100),
                'tol': hp.uniform('tol', 0.0001, 0.001),
                'max_iter': hp.choice('max_iter', range(3000, 4000, 100)),
                'random_state': hp.choice('seed', [27])
            },
            SVC: {
                'C': hp.uniform('C', 1e-2, 2),
                'kernel': hp.choice('kernel', ['linear', 'poly', 'rbf',
'sigmoid']),
                'degree': hp.choice('degree', range(2, 6)),
                'gamma': hp.choice('gamma', ['scale', 'auto']),
                'coef0': hp.uniform('coef0', 0, 1),
                'tol': hp.uniform('tol', 1e-4, 1e-3),
                'max_iter': hp.choice('max_iter', [10000]),
                'cache_size': hp.choice('cache_size', [2000]),
                'random_state': hp.choice('seed', [27])
            },
            RandomForestClassifier: {
                'n_estimators': hp.choice('n_estimators', range(100, 500, 100)),
                'criterion': hp.choice('criterion', ['gini', 'entropy']),
                'max_depth': hp.choice('max_depth', range(1, 20)),
                'max_features': hp.choice('max_features', range(1,
self._n_features)),
                'random_state': hp.choice('seed', [27])
            },
            ExtraTreesClassifier: {
                'max_depth': hp.choice('max_depth', range(1, 20)),
                'max_features': hp.choice('max_features', range(1,
self._n_features)),
                'n_estimators': hp.choice('n_estimators', range(100, 500)),
                'random_state': hp.choice('seed', [27])
            },
            AdaBoostClassifier: {
                'learning_rate': hp.uniform('learning_rate', 0, 1),
                'n_estimators': hp.choice('n_estimators', range(100, 500)),
                'random_state': hp.choice('seed', [27])
            },
            LGBMClassifier: {
                'num_leaves': hp.choice('num_leaves', range(3, 40, 3)),
                'max_depth': hp.choice('max_depth', range(2, 9)),
                'learning_rate': hp.uniform('learning_rate', 0, 1),
                'colsample_bytree': hp.uniform('colsample_bytree', 0, 1),
                'reg_alpha': hp.uniform('reg_alpha', 0, 1),
                'reg_lambda': hp.uniform('reg_lambda', 0, 1),
                'n_estimators': hp.choice('n_estimators', range(10, 1000, 50)),
                # 'metric': hp.choice('metric', ['auc']),
                'random_state': hp.choice('random_state', [27])
            },
            CatBoostClassifier: {
                # The maximum number of trees that can be built when solving
machine learning problems.
                'iterations': hp.choice('iterations', range(1000, 2000, 100)),
                # Coefficient at the L2 regularization term of the cost function.
Any positive value is allowed.
                'l2_leaf_reg': hp.uniform('l2_leaf_reg', 2, 4),
                'allow_writing_files': hp.choice('allow_writing_files=', [False]),
                'verbose': hp.choice('verbose', [False]),
                'random_state': hp.choice('random_state', [27])
```

```
            },
            XGBClassifier: {
                # 'colsample_bylevel': hp.uniform('colsample_bylevel', 0, 1),
                # 'colsample_bynode': hp.uniform('colsample_bynode', 0, 1),
                'colsample_bytree': hp.uniform('colsample_bytree', 0, 1),
                'gamma': hp.uniform('gamma', 0, 1),
                'learning_rate': hp.uniform('learning_rate', 0, 1),
                'max_depth': hp.choice('max_depth', range(1, 10)),
                'reg_alpha': hp.uniform('reg_alpha', 0, 1),
                'reg_lambda': hp.uniform('reg_lambda', 0, 1),
                # 'subsample': hp.uniform('subsample', 0, 1),
                'n_estimators': hp.choice('n_estimators', range(10, 1000, 50)),
                'seed': hp.choice('seed', [42])
            }
        }

        return default_params

    @staticmethod
    def _get_cross_validator(n_folds):
        return StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=27)

    def _make_scoring(self):
        score_dict = {
            'accuracy': make_scorer(accuracy_score),
            'neg_brier_score': make_scorer(brier_score_loss),
            'f1_macro': make_scorer(f1_score, average='macro', zero_division=0),
            'f1_weighted': make_scorer(f1_score, average='weighted',
zero_division=0),
            'neg_log_loss': make_scorer(log_loss),
            'precision_macro': make_scorer(precision_score, average='macro',
zero_division=0),
            'recall_macro': make_scorer(recall_score, average='macro',
zero_division=0),
            'jaccard_macro': make_scorer(jaccard_score, average='macro'),
            'roc_auc_ovr': make_scorer(roc_auc_score, multi_class='ovr'),
            'roc_auc_ovo': make_scorer(roc_auc_score, multi_class='ovo'),
        }
        # Add the F1 score for each label of the classification
        for label in np.unique(self._y):
            score_dict[f'f1_{label}'] = make_scorer(f1_score, labels=[label],
average='macro', zero_division=0)
            score_dict[f'precision_{label}'] = \
                make_scorer(precision_score, labels=[label], average='macro',
zero_division=0)
            score_dict[f'recall_{label}'] = \
                make_scorer(recall_score, labels=[label], average='macro',
zero_division=0)
        return score_dict

    def _get_hp_opt_objective_metric(self, hp_metric_opt):
        if hp_metric_opt is None:
            hp_metric_opt = 'f1_macro'
        return self._make_scoring()[hp_metric_opt]

    @staticmethod
    def _loss_func(metric):
        if metric == 0:
            return 1/0.001
```

```
        return 1.0/metric

    @staticmethod
    def get_available_models() -> List[Model.ClassifierModelEnum]:
        return list(HPClassifierOptimizer.MODELS.keys())


class _ModelTournament:
    KEY_DICT_DATA_DF = 'DataFrame'
    KEY_DICT_DATA_PREDICTORS = 'Predictors'

    def __init__(self, dict_data_group: Dict[str, Dict[str, Union[pd.DataFrame,
List[List]]]], targets_list: List,
                 models_enums: List[Model.ClassifierModelEnum],
                 transformers_list: List[Transformer.TransformerEnum],
                 hp_tuning: bool = True, n_folds=10, experiment_prefix: str =
'default', cv_jobs=4,
                 hp_max_evals=60, parallel=True, parallel_jobs=4,
hp_metric_opt=None, params=None):
        self._dict_data_group = dict_data_group
        self._targets_list = targets_list
        self._models_enums = models_enums
        self._transformers_list = transformers_list
        self._hp_tuning = hp_tuning
        self._n_folds = n_folds
        self._experiment_prefix = experiment_prefix
        self._cv_jobs = cv_jobs
        self._hp_max_evals = hp_max_evals
        self._parallel = parallel
        self._parallel_jobs = parallel_jobs
        self._hp_metric_opt = hp_metric_opt
        self._params = params

        # Find number of executions
        n_exec = 0
        for d in self._dict_data_group:
            n_exec += len(self._dict_data_group[d][self.KEY_DICT_DATA_PREDICTORS])
* \
                          len(self._targets_list) * len(self._transformers_list)
        self.n_exec = n_exec

        self._list_df_results = None

    def init_tournament(self):
        # Init list to keep the results
        list_df_results = []
        i = 0
        minutes_elapsed = np.nan
        # For each data group
        for data_group_name in self._dict_data_group:
            # Get the dataframe
            df_all = self._dict_data_group[data_group_name][self.KEY_DICT_DATA_DF]
            # For each target variable
            for target in self._targets_list:
                # For each list of predictors
                for p_i, predictors in enumerate(

self._dict_data_group[data_group_name][self.KEY_DICT_DATA_PREDICTORS]):
```

```
                    if not predictors:
                        continue
                    # Drop NaNs
                    df = df_all[[target] + predictors].dropna()
                    for transformer in self._transformers_list:
                        # Get the right optimizer class, init the optimizer
                        hp_optimizer = self._get_hp_optimizer_class()(
                            len(predictors),
                            df[predictors],
                            df[target],
                            hp_tuning=self._hp_tuning,
                            n_folds=self._n_folds,
                            experiment_name=self._experiment_prefix + '',
                            cv_jobs=self._cv_jobs,
                            hp_max_evals=self._hp_max_evals,
                            parallel=self._parallel,
                            parallel_jobs=self._parallel_jobs,
                            hp_metric_opt=self._hp_metric_opt,
                            params=self._params
                        )
                        hp_optimizer.build_pipelines_for_models(
                            self._models_enums,
                            models_init_args=None,
                            transformer_enum=transformer
                        )
                        t_init = datetime.datetime.now()
                        i += 1
                        print(f'\n\nData group name: {data_group_name}')
                        print(f'Target: {target}')
                        print(f'Samples: {df.shape[0]}')
                        print(f'Predictors idx: {p_i}. Length :
{len(predictors)}')
                        print(f'Transformer: {transformer}')
                        df = df_all[[target] + predictors].dropna()
                        # Run cross validation with hyperparameters optimization
when available
                        df_results, best_params, trained_models,
trained_models_score = hp_optimizer.run_models()
                        # Register the test type in the cross validation results
                        df_results['Samples'] = df.shape[0]
                        df_results['PredictorsNumber'] = len(predictors)
                        df_results['DataGroup'] = data_group_name
                        df_results['Target'] = [target] * df_results.shape[0]
                        df_results['Predictors'] = [predictors] *
df_results.shape[0]
                        df_results['Transformer'] = transformer
                        df_results['Score'] = trained_models_score
                        df_results['BestParams'] = best_params
                        df_results['TrainedModel'] = trained_models
                        # Append the test results for later concatenation
                        list_df_results.append(df_results)
                        minutes_elapsed = float(np.nanmean(
                            [(datetime.datetime.now() - t_init).total_seconds() /
60.0, minutes_elapsed]))
                        print(f'\nMean minutes per execution:
{round(minutes_elapsed, 1)}')
                        print(f'ETA: {round(minutes_elapsed * (self.n_exec - i),
1)} minutes. '
                              f'{datetime.datetime.now()}')
```

```
        self._list_df_results = list_df_results

    def save_results(self, suffix=''):
        # Concat all resulting dataframes and save them
        df_all_results = pd.concat(self._list_df_results, axis=0)
        df_all_results[
            [col for col in df_all_results.columns if col not in
             ['TrainedModel']
             ]
        ].to_csv(
            f'{self._experiment_prefix}-ModelScores{suffix}.csv',
            index=False,
            encoding='latin-1'
        )
        df_all_results.to_pickle(
            f'{self._experiment_prefix}-ModelScores{suffix}.df'
        )

    @staticmethod
    def _get_hp_optimizer_class() -> Type[_HPOptimizer]:
        raise NotImplementedError


class ClassifierModelTournament(_ModelTournament):

    @staticmethod
    def _get_hp_optimizer_class() -> Type[HPClassifierOptimizer]:
        return HPClassifierOptimizer
```

Apêndice C - Área piloto do estudo em formato GeoJSON

{
"type": "FeatureCollection",
"name": "masters_area",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:EPSG::4674" } },
"features": [
{ "type": "Feature", "properties": { "id": 1 }, "geometry": { "type": "MultiPolygon",
"coordinates": [[[[-46.556190726861843, -23.510866673154286], [-
46.555703039253807, -23.503610261091183], [-46.549819324241327, -
23.505182758067658], [-46.548828217167035, -23.506163755848426], [-
46.547616864076232, -23.505918507088097], [-46.546311119835494, -
23.503076473446914], [-46.546295387977175, -23.502672524602907], [-
46.546515633993685, -23.501835769486284], [-46.547459545493012, -
23.501604939553442], [-46.548120283542531, -23.501287547735483], [-
46.547711255226154, -23.499094637922962], [-46.546547097710317, -
23.498604113628293], [-46.542205104813405, -23.494650998255274], [-
46.542330959679987, -23.493871903136778], [-46.541166802164149, -
23.492255987469772], [-46.539499225182006, -23.491419166214047], [-
46.537894575633146, -23.490582339646181], [-46.538177749082948, -
23.489254948677488], [-46.538083357933004, -23.487725546860936], [-
46.537233837583607, -23.486600126646632], [-46.536006752634499, -
23.486484698389216], [-46.534999913701881, -23.487523549068271], [-
46.534181857069129, -23.488129541517822], [-46.533458191586313, -
23.487869830809068], [-46.533175018136518, -23.483945251161686], [-
46.532545743803631, -23.483310381713579], [-46.531885005754098, -
23.483108377157325], [-46.530878166821481, -23.483454670492893], [-
46.531066949121353, -23.484291542299673], [-46.530563529655041, -
23.484839834741727], [-46.526693492507803, -23.481319180437755], [-
46.526378855341363, -23.479991696218889], [-46.525466407558675, -
23.47817359832673], [-46.524239322609553, -23.477452123964017], [-
46.522571745627403, -23.477538701095892], [-46.519283787238102, -
23.477682996189408], [-46.516924008489788, -23.475056800782848], [-
46.515744119115617, -23.474393028832026], [-46.517757796980852, -
23.473931272461805], [-46.517915115564072, -23.472531563590579], [-
46.516782421764887, -23.47173790763393], [-46.515303627082595, -
23.471478164647657], [-46.514926062482871, -23.473325214773347], [-
46.513541658950516, -23.472690294209048], [-46.510709924452534, -
23.47103082830608], [-46.507547820929787, -23.471420443914607], [-
46.506714032438708, -23.473440654547691], [-46.508240022695965, -
23.474768204692836], [-46.508035508537773, -23.476196749166498], [-
46.50726464748, -23.478577622246203], [-46.504118275815571, -
23.476947089562248], [-46.50199447494208, -23.475908155570195], [-
46.499870674068596, -23.473830263040394], [-46.497526627178594, -
23.47489807301147], [-46.496960280278991, -23.475850436775175], [-
46.496913084704033, -23.476586349520073], [-46.497652482045169, -
23.477365546775321], [-46.491611448449476, -23.477307828617953], [-
46.488763982093154, -23.477581989640512], [-46.483462345838589, -
23.477351117238342], [-46.479529381258068, -23.476773934465111], [-
46.476320082160342, -23.475879296175851], [-46.474259208720142, -
23.476600779140689], [-46.472686022887935, -23.478534334028314], [-

46.472748950321233, -23.479804115849735], [-46.471694915813643, -23.480309139312808], [-46.470766736172635, -23.479804115849735], [-46.47010599812311, -23.479775257307736], [-46.467557437074923, -23.480251422443462], [-46.465952787526057, -23.4808285900003], [-46.462114214095465, -23.480179276321273], [-46.460289318530101, -23.480987310635626], [-46.45937687074742, -23.48218492381692], [-46.459361138889086, -23.483093948248609], [-46.455868666341573, -23.482603364412963], [-46.448757866379971, -23.481290322227242], [-46.446413819489976, -23.480886306617059], [-46.444148431891584, -23.48019370554886], [-46.442008899159767, -23.480006125467007], [-46.440419981469248, -23.480323568526181], [-46.439979489436212, -23.478794063122585], [-46.437934347854352, -23.478794063122589], [-46.437116291221585, -23.478390039866056], [-46.435794815122534, -23.478188027773726], [-46.436203843438918, -23.47817359832672], [-46.436848849630124, -23.477567560127223], [-46.436896045205103, -23.476499771763148], [-46.438107398295898, -23.475532982951158], [-46.435653228397641, -23.474017851904318], [-46.428794138169195, -23.470670072087273], [-46.42080235414155, -23.470583490447996], [-46.417309881594051, -23.471247281563958], [-46.415359131162099, -23.469313619878587], [-46.41158348516479, -23.469025011226766], [-46.407430274567751, -23.470035138746677], [-46.40403219317016, -23.473988992096452], [-46.401546559555257, -23.477105814868288], [-46.401326313538753, -23.479443383620794], [-46.402396079904662, -23.483050661512991], [-46.401357777255399, -23.483512385960417], [-46.399218244523581, -23.481867485237757], [-46.397078711791778, -23.480597723280642], [-46.394152586143861, -23.479760828034355], [-46.391950125978767, -23.479703110925072], [-46.389936448113538, -23.47970311092508], [-46.389967911830169, -23.477019237452307], [-46.388772290597686, -23.47375811340633], [-46.385594455216626, -23.472950034809188], [-46.382699793285347, -23.47165132669533], [-46.381441244619587, -23.471824488515733], [-46.379584885337572, -23.474104431290019], [-46.376564368539718, -23.481117172831567], [-46.377130715439314, -23.481780910947201], [-46.375651920757029, -23.484695547477994], [-46.374928255274213, -23.486080698692479], [-46.373827025191659, -23.48792754434411], [-46.373197750858779, -23.489543513068444], [-46.371530173876629, -23.491303742175674], [-46.370303088927507, -23.493323648255828], [-46.369170395128315, -23.4955166540969], [-46.368824294245215, -23.497305658152499], [-46.367062326113142, -23.498488695881843], [-46.366558906646844, -23.49955630617939], [-46.365017184531276, -23.50218201362733], [-46.364356446481736, -23.50385551414859], [-46.364671083648197, -23.506077197514568], [-46.364954257097985, -23.507866058189165], [-46.365331821697708, -23.509885710398251], [-46.365205966831141, -23.511241745218353], [-46.365772313730737, -23.512915130643886], [-46.367471354429526, -23.512770874117113], [-46.371152609276912, -23.512020737630621], [-46.373480924308581, -23.512338103588679], [-46.374236053508049, -23.512915130643886], [-46.373512388025226, -23.513665262035769], [-46.37341799687529, -23.5151943627937], [-46.374645081824418, -23.516521492454086], [-46.376910469422796, -23.517127351547469], [-46.379490494187635, -23.516838847564976], [-

46.380623187986821, -23.517185052268104], [-46.382479547268844, -23.519550760041575], [-46.383391995051511, -23.522118370369434], [-46.383769559651242, -23.523935862013737], [-46.387545205648564, -23.527426529543245], [-46.389149855197417, -23.527974642480395], [-46.391352315362518, -23.530686325529384], [-46.392516472878349, -23.531378661182288], [-46.393932340127343, -23.530859409784135], [-46.393303065794456, -23.533513339845161], [-46.393114283494597, -23.534897977755367], [-46.392390618011774, -23.538647965575457], [-46.390628649879694, -23.540840216626819], [-46.388080088831515, -23.542195932542725], [-46.387796915381713, -23.543955457890576], [-46.387199104765472, -23.545772647879815], [-46.387419350781975, -23.547243688048187], [-46.392610864028285, -23.550185719007327], [-46.394813324193386, -23.549753071524911], [-46.392547936595008, -23.551858609208335], [-46.39185573482883, -23.553877586269856], [-46.391289387929227, -23.555723481014734], [-46.389810593246949, -23.559501715973244], [-46.383675168501327, -23.562443472548996], [-46.379175857021181, -23.561318691047394], [-46.378924147288032, -23.561578256863964], [-46.379490494187621, -23.564721846594157], [-46.379679276487494, -23.565529362087723], [-46.381346853469644, -23.56599079728387], [-46.381692954352722, -23.567029020548642], [-46.381598563202793, -23.568817052490104], [-46.382857111868567, -23.569393831794883], [-46.382322228685616, -23.571152993029884], [-46.381315389752999, -23.572162337197774], [-46.38056026055353, -23.57308515936225], [-46.379553421620912, -23.573344701927546], [-46.378263409238507, -23.573575405999357], [-46.377476816322392, -23.57507497258602], [-46.378231945521854, -23.576343823244276], [-46.380025377370579, -23.577756847045514], [-46.380969288869906, -23.577756847045514], [-46.38301443045178, -23.578593119274156], [-46.378924147288032, -23.583927833958665], [-46.379144393304536, -23.584908244422994], [-46.377571207472329, -23.586667197780695], [-46.376816078272874, -23.58776292729868], [-46.37700486057274, -23.589493007908917], [-46.378263409238507, -23.589954358885713], [-46.379962449937302, -23.589608345805189], [-46.381158071169786, -23.590992392652325], [-46.381787345502673, -23.593587441138023], [-46.383171749035029, -23.596240104310802], [-46.382479547268844, -23.596701431560614], [-46.381850272935971, -23.598604389314996], [-46.383738095934618, -23.601314614814683], [-46.38940156493058, -23.602583211761058], [-46.388835218030991, -23.610598154168198], [-46.395002106493266, -23.613711740637999], [-46.397204566658353, -23.613596423939601], [-46.414572538246006, -23.629912727989346], [-46.422690177140225, -23.638819534534889], [-46.428794138169216, -23.637695407897208], [-46.430933670901013, -23.639424829495237], [-46.43307320363283, -23.639338358957975], [-46.433167594782759, -23.640693057478998], [-46.435401518664499, -23.641240697582372], [-46.435558837247719, -23.645996423178048], [-46.436628603613627, -23.646774616367228], [-46.438107398295912, -23.64599642317804], [-46.439869366427992, -23.643431235082129], [-46.44235500004288, -23.644641779550078], [-46.444651851357911, -23.644238265971584], [-46.444777706224492, -23.647062834892495], [-46.446571138073217, -23.646716972585995], [-46.447798223022346, -23.645448802974467], [-

46.448207251338708, -23.646659328779371], [-46.448616279655099, -23.647725735091377], [-46.451007522120051, -23.648417453457473], [-46.451322159286498, -23.65034848120867], [-46.449088235404751, -23.651645424146537], [-46.446980166389586, -23.652423583737747], [-46.445375516840734, -23.650694333915908], [-46.444337214191478, -23.65129957395358], [-46.44452599649135, -23.654325732140578], [-46.448993844254836, -23.657265361629594], [-46.451888506186108, -23.661357676979733], [-46.456293426516304, -23.661300039624503], [-46.462617633561798, -23.670319976560002], [-46.465386440626489, -23.668360426562916], [-46.464190819394013, -23.666227941718272], [-46.467274263625157, -23.664787053887704], [-46.476021176852264, -23.660204925048305], [-46.474888483053057, -23.655305615977674], [-46.482628557347539, -23.654642754185264], [-46.489739357309141, -23.658389320150405], [-46.488449344926728, -23.664556510361137], [-46.493168902423378, -23.665190504081], [-46.495685999754919, -23.663807241092222], [-46.49511965285533, -23.662279871208174], [-46.491627180307809, -23.662971512611328], [-46.491344006858014, -23.662106960285758], [-46.491564252874504, -23.657438278953801], [-46.493043047556803, -23.653662865382547], [-46.496818693554111, -23.657524737530178], [-46.499367254602298, -23.65769765451147], [-46.506824155446992, -23.654959775461606], [-46.513588854525509, -23.658302862145586], [-46.518182557155569, -23.656631329484867], [-46.519535496971272, -23.657841751821234], [-46.520762581920394, -23.658591055272712], [-46.52214698545275, -23.660608389380219], [-46.523625780135035, -23.661789956334115], [-46.524601155351007, -23.660954214959645], [-46.523720171284964, -23.659542088086358], [-46.524758473934227, -23.658533416698095], [-46.526017022600001, -23.658677513087014], [-46.527558744715563, -23.657265361629591], [-46.527841918165358, -23.656256672681909], [-46.529729741164019, -23.655305615977671], [-46.530799507529927, -23.655305615977671], [-46.5306421889467, -23.653662865382547], [-46.528974611964564, -23.652827072073865], [-46.530390479213551, -23.651847169670202], [-46.529415103997579, -23.650550228732335], [-46.527653135865513, -23.650953722846346], [-46.526426050916385, -23.649339738923864], [-46.529257785414359, -23.64726458748245], [-46.53422905264415, -23.644584134829334], [-46.540301549956496, -23.629595646038155], [-46.544769397719975, -23.626655395017583], [-46.5462167286856, -23.623570943107371], [-46.54819894283419, -23.61670993918116], [-46.556379509161701, -23.613365790238401], [-46.567517664853767, -23.611491876371264], [-46.575320666581554, -23.608753030988883], [-46.576925316130406, -23.606360098185466], [-46.57931655859538, -23.604860889115063], [-46.579285094878728, -23.601747092425175], [-46.58170780106034, -23.600680311739339], [-46.584099043525299, -23.597624081192571], [-46.584854172724768, -23.596009440077431], [-46.584856139206984, -23.59598060701973], [-46.587308342623018, -23.594221778515863], [-46.591776190386497, -23.590329205359897], [-46.594135969134825, -23.587503413239705], [-46.597534050532403, -23.58162095691803], [-46.597471123099112, -23.580957722264934], [-46.595520372667167, -23.580698194749321], [-46.595205735500727, -23.577064755655336], [-46.597125022216026, -23.57726661602101], [-

46.602536781478854, -23.567490450476669], [-46.604424604477501, -23.567778843358511], [-46.60602925402636, -23.569480348472137], [-46.607036092958971, -23.569970608566678], [-46.607382193842056, -23.569768736984685], [-46.607476584992, -23.572681425463276], [-46.608200250474816, -23.572537234484241], [-46.608798061091051, -23.571383700952218], [-46.608514887641256, -23.5698264146112], [-46.608326105341391, -23.568788213458372], [-46.609301480557363, -23.56757696840771], [-46.610969057539513, -23.564577646876522], [-46.61222760620528, -23.563164481267268], [-46.613045662838012, -23.561318691047394], [-46.613989574337353, -23.561751300456809], [-46.614367138937084, -23.563366362999293], [-46.615153731853184, -23.563222161793789], [-46.617293264584994, -23.56241463211796], [-46.617230337151703, -23.561722459873828], [-46.622830878714396, -23.560193899916598], [-46.62368039906378, -23.561145646884778], [-46.624246745963376, -23.560914920980107], [-46.624498455696539, -23.562645355388984], [-46.625914322945526, -23.562328110786872], [-46.625725540645654, -23.560568831363419], [-46.624813092862965, -23.557655874297993], [-46.625064802596128, -23.555117799659346], [-46.625568222062448, -23.5540794822703], [-46.626575060995066, -23.551224067155502], [-46.627550436211038, -23.550041503338072], [-46.627770682227549, -23.549205049336429], [-46.628871912310096, -23.547849405686609], [-46.629438259209678, -23.544878478166375], [-46.62924947690982, -23.542974741748356], [-46.628525811426996, -23.542224777410375], [-46.627770682227528, -23.540119085445646], [-46.628777521160153, -23.539253722808915], [-46.628525811426989, -23.537522980454614], [-46.628368492843769, -23.536282601091848], [-46.627644827360953, -23.535157595741079], [-46.628399956560422, -23.534292200462868], [-46.627739218510882, -23.533715267114744], [-46.626858234444846, -23.533542186616938], [-46.626228960111959, -23.532042146097854], [-46.625788468078945, -23.531984451890189], [-46.625914322945526, -23.529936291127775], [-46.624813092862965, -23.529359338678876], [-46.623145515880829, -23.528205426191992], [-46.623208443314113, -23.527426529543245], [-46.623523080480567, -23.52592641930471], [-46.623176979597474, -23.526012964590947], [-46.622610632697885, -23.52621490337086], [-46.622201604381502, -23.526705324832044], [-46.622516241547935, -23.526099509820288], [-46.623680399063787, -23.525724480082147], [-46.62358600791385, -23.524397443197074], [-46.622484777831296, -23.524368594420533], [-46.622044285798282, -23.524974417399878], [-46.621509402615324, -23.524743628022275], [-46.621415011465395, -23.523993559750185], [-46.620785737132501, -23.523676221886749], [-46.620439636249422, -23.522983845710876], [-46.621383547748749, -23.52252225957027], [-46.622170140664863, -23.521599082433131], [-46.621509402615324, -23.520906695331433], [-46.621068910582309, -23.52070474840745], [-46.618929377850506, -23.521801027985116], [-46.617419119451576, -23.519810408305872], [-46.616097643352511, -23.519723858941337], [-46.612731025671579, -23.521570233043271], [-46.61118930355601, -23.52102209343462], [-46.610056609756818, -23.521887575983985], [-46.609175625690774, -23.523156940096285], [-46.605211197393601, -23.522753052842901], [-46.602033362012527, -23.523849315361648], [-

46.59344376736864, -23.52909970925154], [-46.586521749706911, -23.528263122056654], [-46.581644873627056, -23.527541921930265], [-46.577837763913088, -23.526763021354398], [-46.574722855965305, -23.525811025501191], [-46.571954048900615, -23.525811025501191], [-46.570506717934983, -23.525637934606205], [-46.568146939186661, -23.525147509167507], [-46.566070333888135, -23.524022408608918], [-46.563993728589615, -23.523503128184231], [-46.561067602941698, -23.523387732256086], [-46.55792123127727, -23.52289729843281], [-46.556253654295119, -23.519695009140531], [-46.555939017128686, -23.516550342950321], [-46.55927417109298, -23.517011950030337], [-46.559368562242902, -23.516204136578278], [-46.556536827744928, -23.515829078647972], [-46.556348045445063, -23.514473091063696], [-46.557103174644517, -23.514271134271315], [-46.557575130394191, -23.513521006330713], [-46.557480739244255, -23.512886279351182], [-46.556159263145197, -23.512857428052143], [-46.556190726861843, -23.510866673154286]]]] } }
]
}

Apêndice D - *Notebook* Python para obtenção do IPF para cada consumidor da BDGD

A versão Python utilizada é 3.8.10. Esse *notebook* também se encontra no link em (PULZ, 2022) – arquivo “1.0-jp-indice_potencial_de_fraude.ipynb”. Os seguintes pacotes, com suas respectivas versões, foram utilizados para execução do *notebook* Python:

argon2-cffi==21.3.0
argon2-cffi-bindings==21.2.0
asttokens==2.0.5
attrs==21.4.0
backcall==0.2.0
bleach==4.1.0
certifi==2021.10.8
cffi==1.15.0
click==8.0.4
click-plugins==1.1.1
cligj==0.7.2
colorama==0.4.4
cycler==0.11.0
debugpy==1.5.1
decorator==5.1.1
defusedxml==0.7.1
entrypoints==0.4
executing==0.8.3
Fiona==1.8.20
fonttools==4.29.1
GDAL==3.4.1
geopandas==0.10.2
importlib-resources==5.4.0
ipykernel==6.9.1
ipython==8.1.1
ipython-genutils==0.2.0
ipywidgets==7.6.5
jedi==0.18.1
Jinja2==3.0.3
jsonschema==4.4.0
jupyter==1.0.0
jupyter-client==7.1.2
jupyter-console==6.4.0
jupyter-core==4.9.2
jupyterlab-pygments==0.1.2
jupyterlab-widgets==1.0.2
kiwisolver==1.3.2
MarkupSafe==2.1.0
matplotlib==3.5.1

matplotlib-inline==0.1.3
mistune==0.8.4
munch==2.5.0
nbclient==0.5.11
nbconvert==6.4.2
nbformat==5.1.3
nest-asyncio==1.5.4
notebook==6.4.8
numpy==1.22.2
packaging==21.3
pandas==1.4.1
pandocfilters==1.5.0
parso==0.8.3
pickleshare==0.7.5
Pillow==9.0.1
prometheus-client==0.13.1
prompt-toolkit==3.0.28
pure-eval==0.2.2
pycparser==2.21
Pygments==2.11.2
PyKrige==1.6.1
pyodbc==4.0.32
pyparsing==3.0.7
pyproj==3.3.0
pyrsistent==0.18.1
python-dateutil==2.8.2
pytz==2021.3
pywin32==303
pywinpty==2.0.3
pyzmq==22.3.0
qtconsole==5.2.2
QtPy==2.0.1
Rtree==0.9.7
scipy==1.8.0
Send2Trash==1.8.0
Shapely==1.8.1.post1
six==1.16.0
stack-data==0.2.0
terminado==0.13.2
testpath==0.6.0
tornado==6.1
traitlets==5.1.1
wcwidth==0.2.5
webencodings==0.5.1
widgetsnbextension==3.5.2
zipp==3.7.0

Segue o *notebook script* Python transformado em documento Word utilizando o software Pandoc:

Índice Potencial de Fraude (IPF)

2021-03-02 Jonatas Pulz

```
import pickle
import os

import pandas as pd
import numpy as np

import pyodbc
from pykrige.ok import OrdinaryKriging

import fiona
from shapely.geometry import box, Point
import geopandas
import rtree

import matplotlib
from matplotlib import pyplot as plt
from IPython import display
```

Configurações

```
pd.set_option('display.max_rows', 200)
pd.set_option('display.max_columns', 200)

font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 15}

matplotlib.rc('font', **font)
```

Conexão com o banco de dados

```
conn = pyodbc.connect('DRIVER={SQL Server Native Client
11.0};SERVER=localhost;DATABASE=InspectionEnel2016;Trusted_Connection=yes;')
```

Queries

```
sql_inspections_lv = f'\
SELECT \
    insp.* \
    ,insp.latitude AS y \
    ,insp.longitude AS x \
FROM \
    [InspectionEnel2016].[dbo].[InspectionLV] insp \
WHERE \
    insp.longitude IS NOT null and insp.latitude IS NOT null \
    AND \
    TRIM(insp.longitude) <> \'\' and TRIM(insp.latitude) <> \'\' \
    AND \
    TRY_CAST(insp.longitude AS float) <> 0 and TRY_CAST(insp.latitude AS float) <>
```

```
0 \
'

sql_bdgd_ucbt = f'\
SELECT \
   cod_id \
    ,shape.STX AS x \
    ,shape.STY AS y \
FROM \
    [BDGDEnel2019].[dbo].[ucbt] \
'
```

Carregamento dos dados

Dados privados de inspeções realizadas

```
df_inspections_lv_raw = pd.read_sql(sql_inspections_lv, conn)
df_inspections_lv = df_inspections_lv_raw.copy()
df_inspections_lv['hit_f'] = df_inspections_lv['HIT_F']

df_frauds_lv = df_inspections_lv.loc[
    df_inspections_lv['hit_f'].astype(int) == 1
].copy()
df_frauds_lv['y'] = df_frauds_lv['y'].astype(float)
df_frauds_lv['x'] = df_frauds_lv['x'].astype(float)
df_frauds_lv.describe()

df_inspections_lv_raw.to_csv(
    'all_inspections.csv'
    ,index=False
    ,encoding='utf-8'
)

df_frauds_lv[['hit_f', 'x', 'y']].to_csv(
    'found_frauds.csv'
    ,index=False
    ,encoding='utf-8'
)
```

Dados de consumidores de baixa tensão da BDGD

```
df_bdgd_ucbt_raw = pd.read_sql(sql_bdgd_ucbt, conn)

gdf_bdgd_ucbt = geopandas.GeoDataFrame(
    df_bdgd_ucbt_raw
    ,geometry=[
        Point(p[0], p[1])
        for p in np.column_stack((df_bdgd_ucbt_raw['x'], df_bdgd_ucbt_raw['y']))
    ]
    ,crs='EPSG:4674'
)
```

Histograma (mapa) de fraude

```
bins = 100
plot_size = 15
# https://matplotlib.org/stable/tutorials/colors/colormaps.html
cmap = 'plasma'

fig_size_factor = 1.115

hist_fraud_lv = np.histogram2d(df_frauds_lv['x'], df_frauds_lv['y'], bins=bins)

fig, ax = plt.subplots(figsize=(15, 6))
ax.hist(np.ravel(hist_fraud_lv[0]), bins=bins)
ax.set_xlabel('Número de fraudes')
ax.set_ylabel('Contagem')

x_min = min(hist_fraud_lv[1])
x_max = max(hist_fraud_lv[1])
y_min = min(hist_fraud_lv[2])
y_max = max(hist_fraud_lv[2])
relation_y_x = (y_max-y_min)/(x_max-x_min)

## (Início) Removendo o valor outlier com mais de 200 fraudes no ano, para melhor
visualização
hist_fraud_lv_values_plot = np.ravel(hist_fraud_lv[0])
index_max = np.argmax(hist_fraud_lv[0])
display.display(f'Número máximo de fraudes em um retângulo:
{np.ravel(hist_fraud_lv[0])[index_max]}')
value_second_max =
hist_fraud_lv_values_plot[np.arange(hist_fraud_lv_values_plot.shape[0])!=index_max
][np.argmax(hist_fraud_lv_values_plot[np.arange(hist_fraud_lv_values_plot.shape[0]
)!=index_max])]
display.display(f'Número máximo de fraudes considerado: {value_second_max}')
hist_fraud_lv_values_plot[index_max] = value_second_max
hist_fraud_lv_values_plot =
hist_fraud_lv_values_plot.reshape(hist_fraud_lv[0].shape)
## (Fim) Removendo o valor outlier com mais de 200 fraudes no ano, para melhor
visualização

fig, ax = plt.subplots(figsize=(15, 6))
ax.hist(np.ravel(hist_fraud_lv_values_plot), bins=bins)
ax.set_xlabel('Número de fraudes')
ax.set_ylabel('Contagem')

fig, ax = plt.subplots(figsize=(plot_size, relation_y_x*plot_size))
ax.pcolormesh(
    hist_fraud_lv[1]
    ,hist_fraud_lv[2]
    ,hist_fraud_lv_values_plot.T
    ,cmap=cmap
)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

xv, yv = np.meshgrid(np.convolve(hist_fraud_lv[1], np.ones(2)/2, mode='valid'),
np.convolve(hist_fraud_lv[2], np.ones(2)/2, mode='valid'))
x = np.ravel(xv)
y = np.ravel(yv)

# https://geostat-
framework.readthedocs.io/projects/pykrige/en/stable/examples/00_ordinary.html#sphx
```

```
-glr-examples-00-ordinary-py
OK = OrdinaryKriging(
    x,
    y,
    np.ravel(hist_fraud_lv_values_plot),
    variogram_model='gaussian',
    verbose=True,
    enable_plotting=True,
    coordinates_type="geographic"
)

kriging_axis_n = 500

kriging_axis_x_step = (np.max(hist_fraud_lv[1])-
np.min(hist_fraud_lv[1]))/kriging_axis_n
kriging_axis_x = np.append(np.arange(np.min(hist_fraud_lv[1]),
np.max(hist_fraud_lv[1]), kriging_axis_x_step), np.max(hist_fraud_lv[1]))

kriging_axis_y_step = (np.max(hist_fraud_lv[2])-
np.min(hist_fraud_lv[2]))/kriging_axis_n
kriging_axis_y = np.append(np.arange(np.min(hist_fraud_lv[2]),
np.max(hist_fraud_lv[2]), kriging_axis_y_step), np.max(hist_fraud_lv[2]))

if os.path.isfile(os.path.join('.', f'kriging_{kriging_axis_n}_z_o')) \
    and os.path.isfile(os.path.join('.', f'kriging_{kriging_axis_n}_ss_o')):
    print('Files exist')
    with open(os.path.join('.', f'kriging_{kriging_axis_n}_z_o'), 'rb') as f:
        z_o = pickle.load(f)
    with open(os.path.join('.', f'kriging_{kriging_axis_n}_ss_o'), 'rb') as f:
        ss_o = pickle.load(f)
else:
    print(f'Files don\'t exisit for {kriging_axis_n} bins. Kriging will be
executed.')
    z_o, ss_o = OK.execute('grid', kriging_axis_x, kriging_axis_y)
    with open(os.path.join('.', f'kriging_{kriging_axis_n}_z_o'), 'wb') as handle:
        pickle.dump(z_o, handle, protocol=pickle.HIGHEST_PROTOCOL)
    with open(os.path.join('.', f'kriging_{kriging_axis_n}_ss_o'), 'wb') as
handle:
        pickle.dump(ss_o, handle, protocol=pickle.HIGHEST_PROTOCOL)

kriging_values = z_o.data.copy()
kriging_values = kriging_values - np.min(kriging_values)
kriging_values = kriging_values/np.max(kriging_values)

fig, ax = plt.subplots(figsize=(fig_size_factor*plot_size,
relation_y_x*plot_size))

pcolor = ax.pcolormesh(
    hist_fraud_lv[1]
    ,hist_fraud_lv[2]
    ,hist_fraud_lv_values_plot.T
    ,cmap=cmap
)
fig.colorbar(pcolor, ax=ax, label='Número de fraudes')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

fig, ax = plt.subplots(figsize=(fig_size_factor*plot_size,
relation_y_x*plot_size))
```

```
pcolor = ax.pcolormesh(
    kriging_axis_x
    ,kriging_axis_y
    ,kriging_values.T
    ,cmap=cmap
)
fig.colorbar(pcolor, ax=ax, label='IPF')
ax.set_xlabel('Longitude', )
ax.set_ylabel('Latitude')

plt.show()

fig, ax = plt.subplots(figsize=(plot_size, 6))
display.display(round(hist_fraud_lv[2][int(bins/2)], 5))
display.display(round(hist_fraud_lv[2][int(bins/2)+1], 5))
ax.plot(hist_fraud_lv[1][:-1], hist_fraud_lv_values_plot[int(bins/2), :], c='b',
label='Número de fraudes')
ax_twin = ax.twinx()
ax_twin.plot(kriging_axis_x[:-1], kriging_values[int(kriging_axis_n/2), 1:],
c='r', label='IPF')
ax.set_xlabel(
    f'Longitiude para latitude entre ' +
    f'{round(kriging_axis_y[int(kriging_axis_n/2)],5)} e '
    f'{round(kriging_axis_y[int(kriging_axis_n/2)+1],5)}'
)
ax.set_ylabel(f'Número de fraudes')
ax_twin.set_ylabel(f'IPF')
fig.legend()
plt.show()
```

Cria e exporta shapefiles

Cria e exporta shapefile do IPF

```
axis_x = kriging_axis_x.copy()
axis_y = kriging_axis_y.copy()

boxes = []
values = []
for i in range(0, axis_x.shape[0]-1):
    for j in range(0, axis_y.shape[0]-1):
        boxes.append(box(axis_x[i], axis_y[j], axis_x[i+1], axis_y[j+1]))
        values.append((i, j, kriging_values[i, j]))
gdf_boxes = geopandas.GeoDataFrame(
    values
    ,columns=['i', 'j', 'value']
    ,geometry=boxes
    ,crs='EPSG:4326'
)

gdf_boxes.to_file(
    f'kriging_{kriging_axis_n}.shp'
    ,encoding='utf-8'
)
```

Obtém IPF para cada consumidor da BDGD e exporta

```
gdf_bdgd_ucbt_ipf = gdf_bdgd_ucbt.sjoin(
    gdf_boxes
    ,how='left'
    ,predicate='intersects'
)

gdf_bdgd_ucbt_ipf.to_file(
    f'ipf_kriging_{kriging_axis_n}.shp'
    ,encoding='utf-8'
)
```

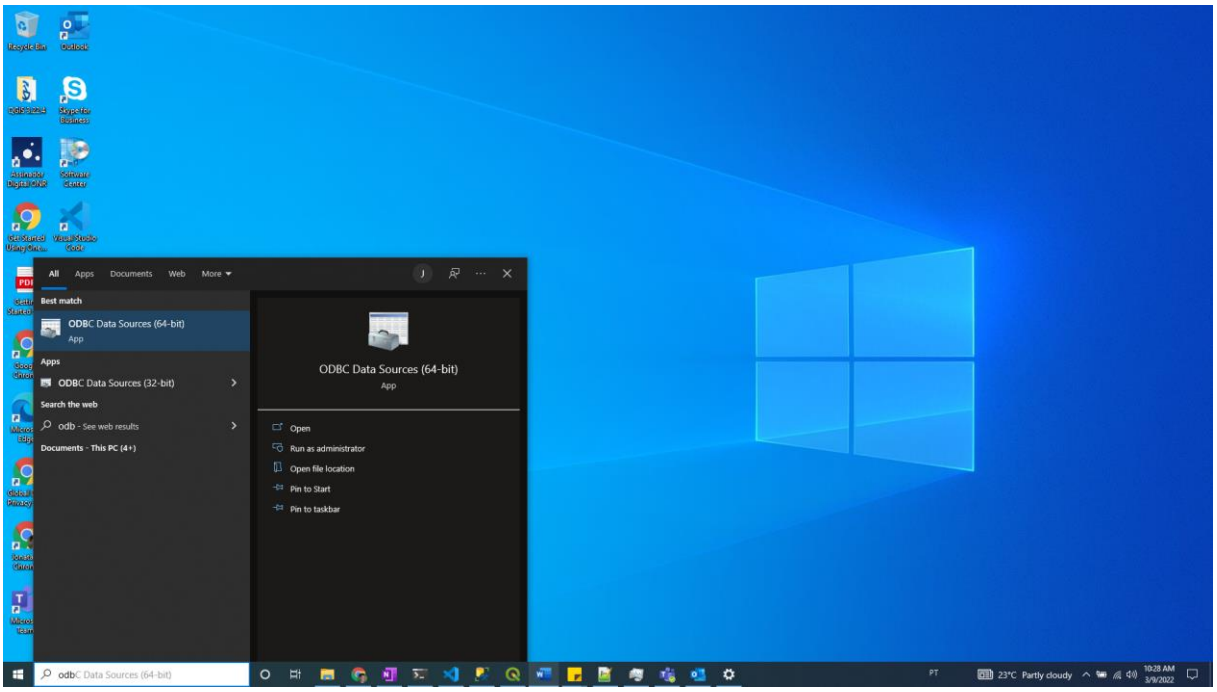
Apêndice E - Passo-a-passo no QGIS para obtenção dos alimentadores da área piloto

O ambiente utilizado em todo este trabalho é um ambiente Windows 10. Primeiramente, é necessário instalar o QGIS e recomenda-se instalar a versão *Long Term Release* (LTR). A versão aqui utilizada é a 3.22.4-Białowieża.

Em seguida, é necessário criar uma conexão ODBC para que o QGIS possa acessar os dados da BDGD diretamente no SQL Server. Deve-se seguir os seguintes passos:

- Abrir o ODBC Data Sources (64-bit), conforme Figura 26;

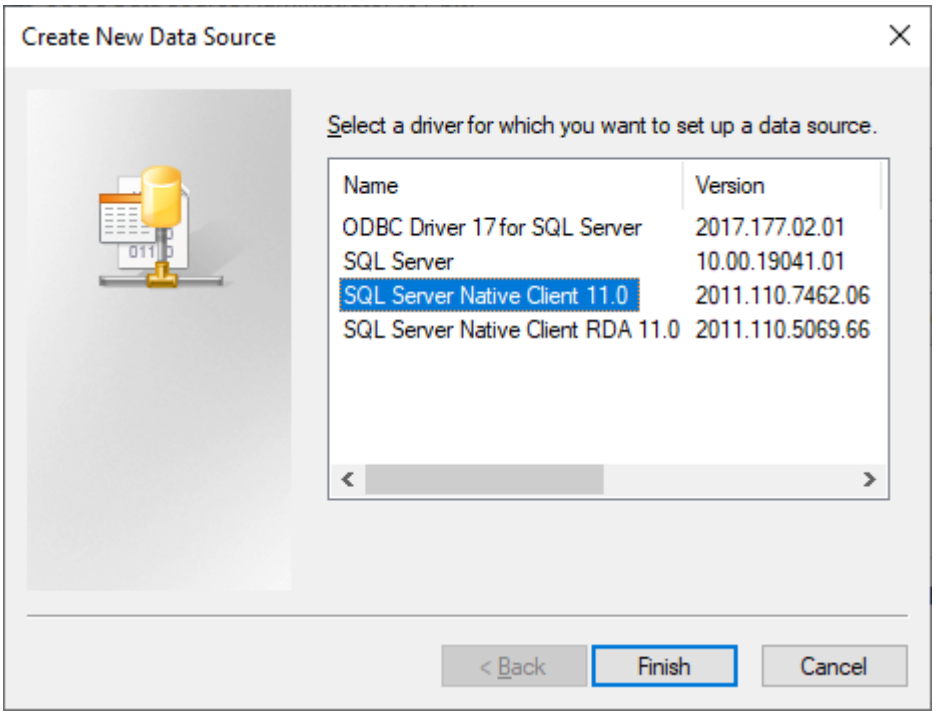
Figura 26 - Abertura do ODBC Data Sources



Fonte: Autor

- Clicar em "*Add...*" e criar uma conexão SQL Server Native Client 11.0, conforme

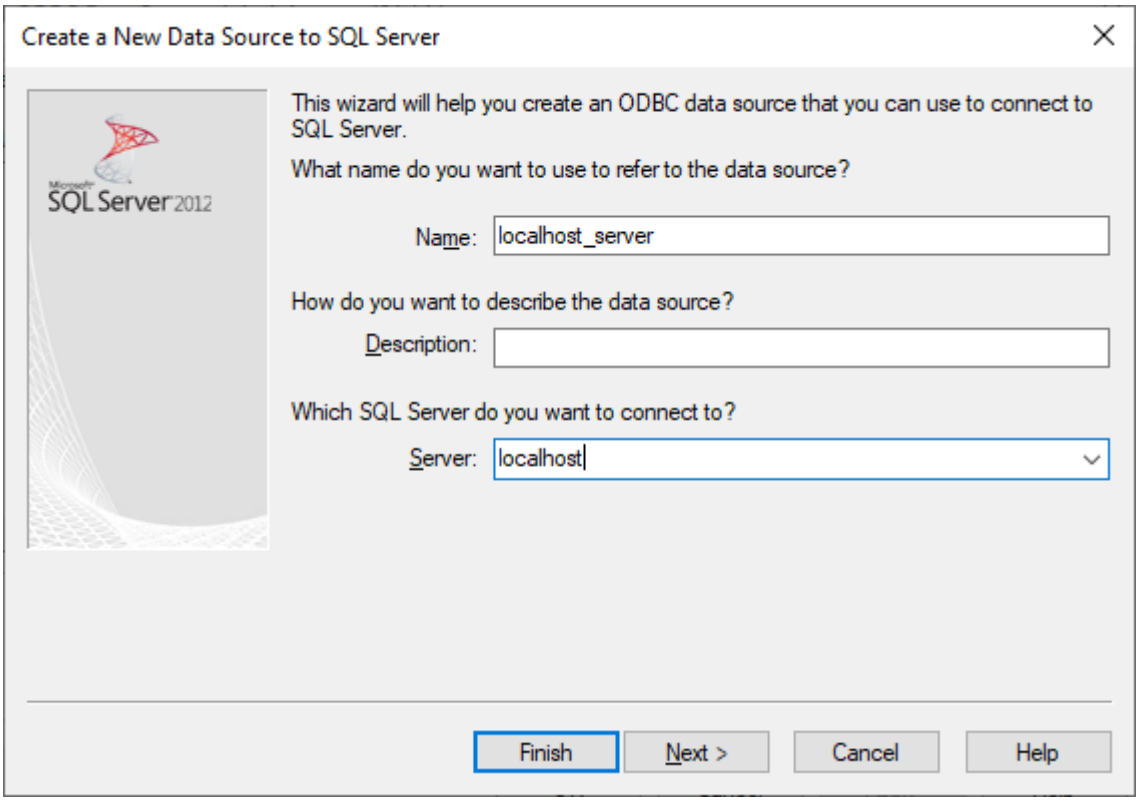
Figura 27 - Criação da conexão SQL Server



Fonte: Autor

- Configurar a conexão conforme Figura 28 e clicar em "*Finish*"

Figura 28 - Configuração da conexão ODBC com SQL Server

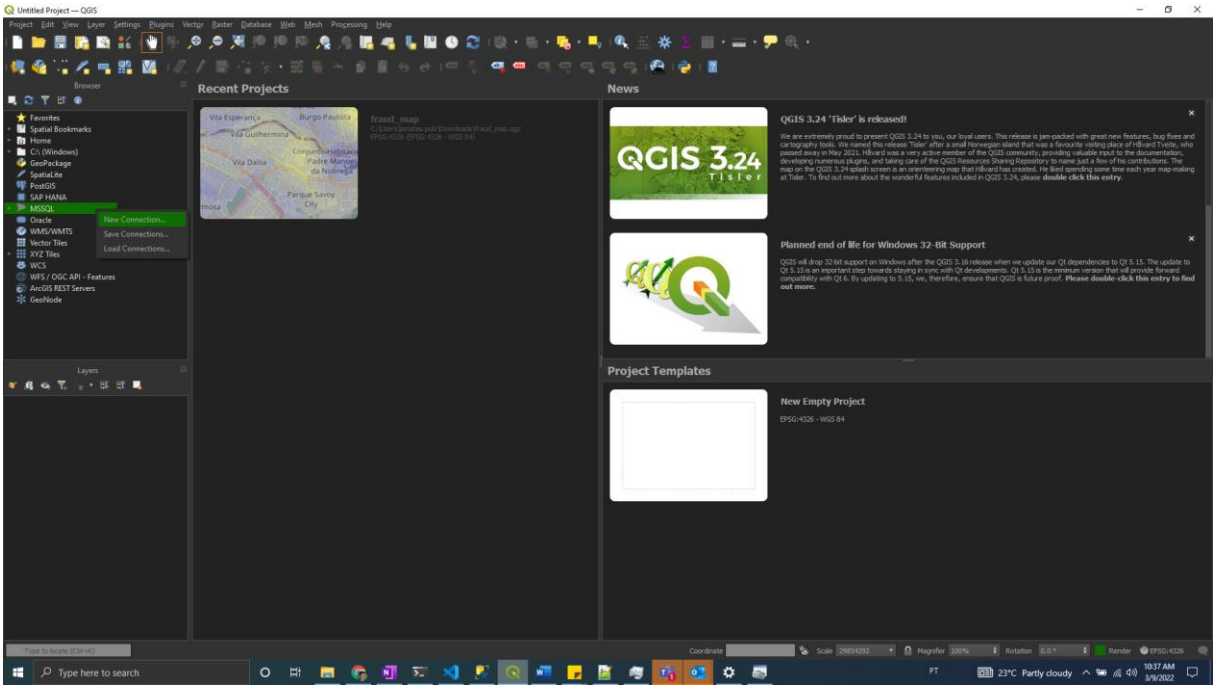


Fonte: Autor

Depois de a conexão ODBC chamada “localhost_server” ter sido criada, deve-se seguir os seguintes passos para adicionar essa conexão ao QGIS:

- Abra o QGIS, clique com o botão direito do mouse em MSSQL e depois em “New Connection...”, conforme Figura 29;

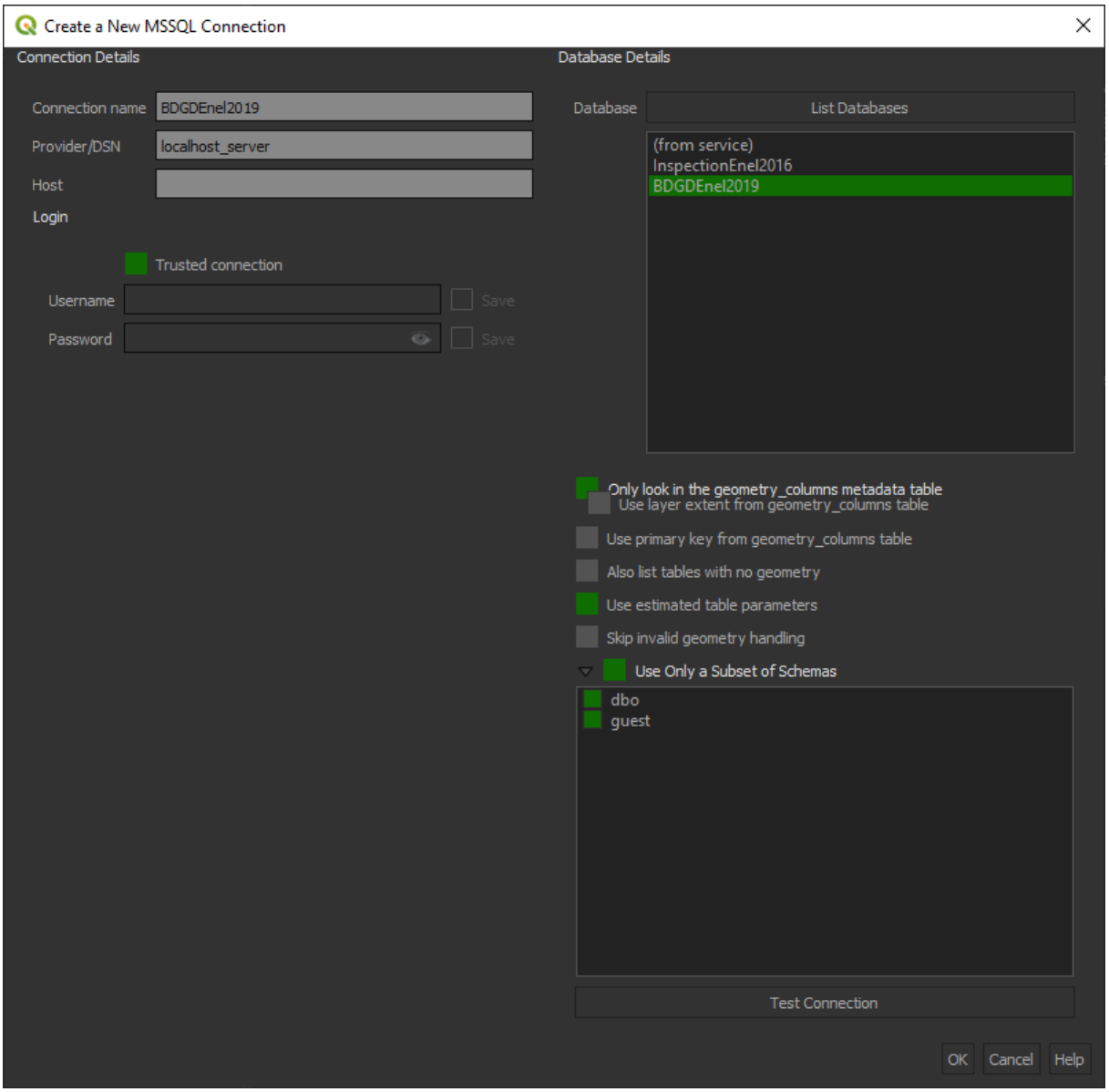
Figura 29 - Criação da conexão no QGIS, primeiro passo



Fonte: Autor

- Preencha os campos “Connection name” e “Provider/DSN” conforme Figura 30, depois clique em “List Databases” e selecione o banco BDGDEnel2019. Após isso, basta clicar em “OK” e a conexão BDGDEnel2019 estará disponível dentro de “MSSQL” no QGIS.

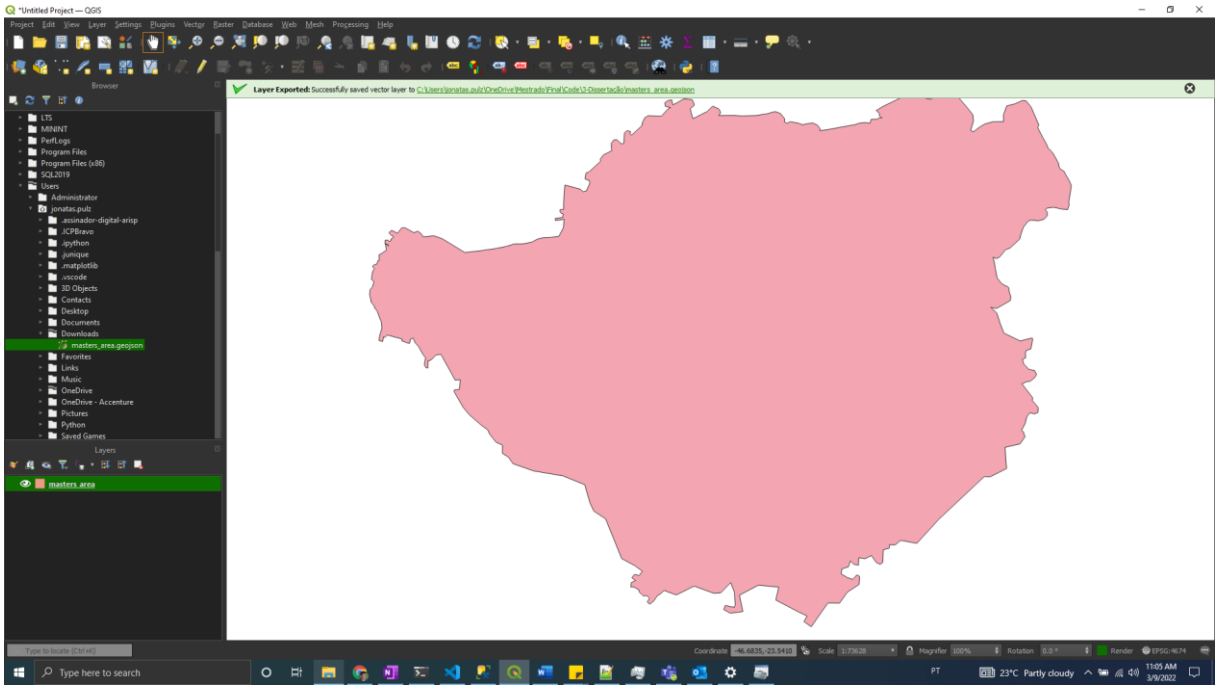
Figura 30 - Criação da conexão no QGIS, segundo passo



Fonte: Autor

Além disso, deve-se salvar o conteúdo do 0 num arquivo chamado “masters_area.geojson”, depois abrir esse através de dois cliques nele no “Browser” do QGIS, conforme Figura 31.

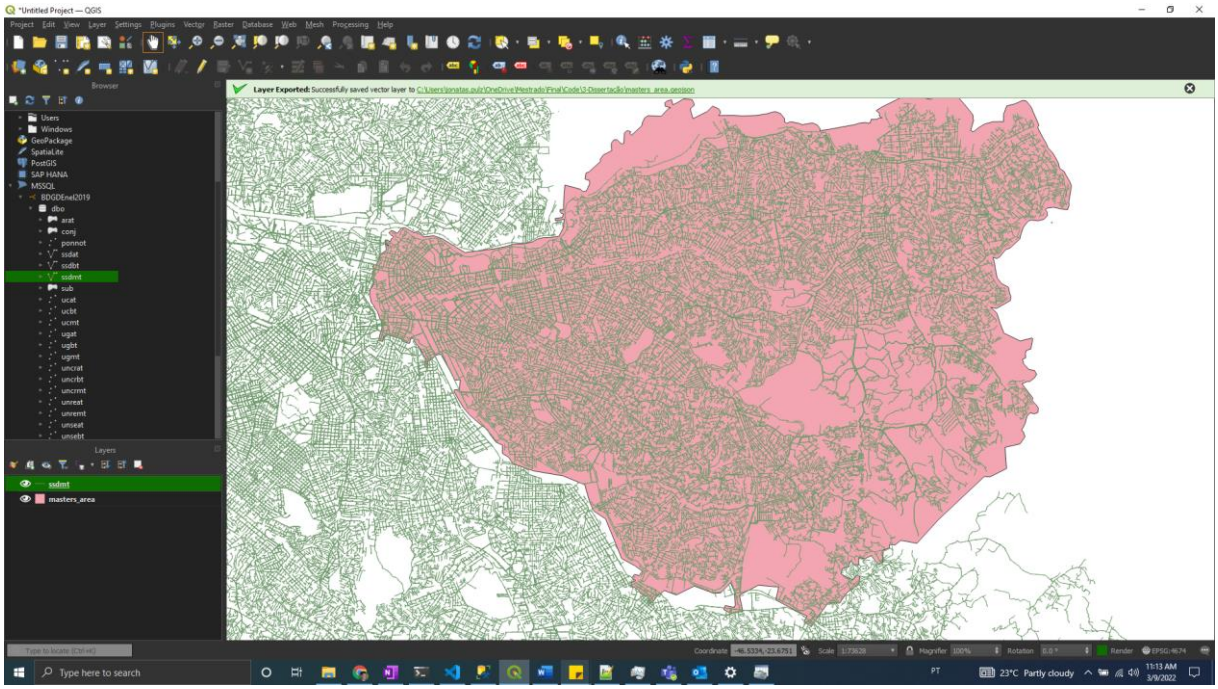
Figura 31 - Inclusão da área piloto no QGIS



Fonte: Autor

Depois, ainda utilizando o “Browser” do QGIS, deve-se expandir o item “MSSQL”, dentre deste, deve-se expandir a conexão com o SQL Server chamada “BDGDEnel2019”, dentro deste, deve-se expandir o item “dbo”, e aí apareceram todas as tabelas georreferenciadas da BDGD. O interesse está na tabela “ssdmt”, que representa os segmentos de média tensão da distribuidora, basta dar dois cliques para incluir esses segmentos no QGIS. Tudo isso pode ser visto na Figura 32.

Figura 32 - Inclusão dos segmentos de média tensão no QGIS

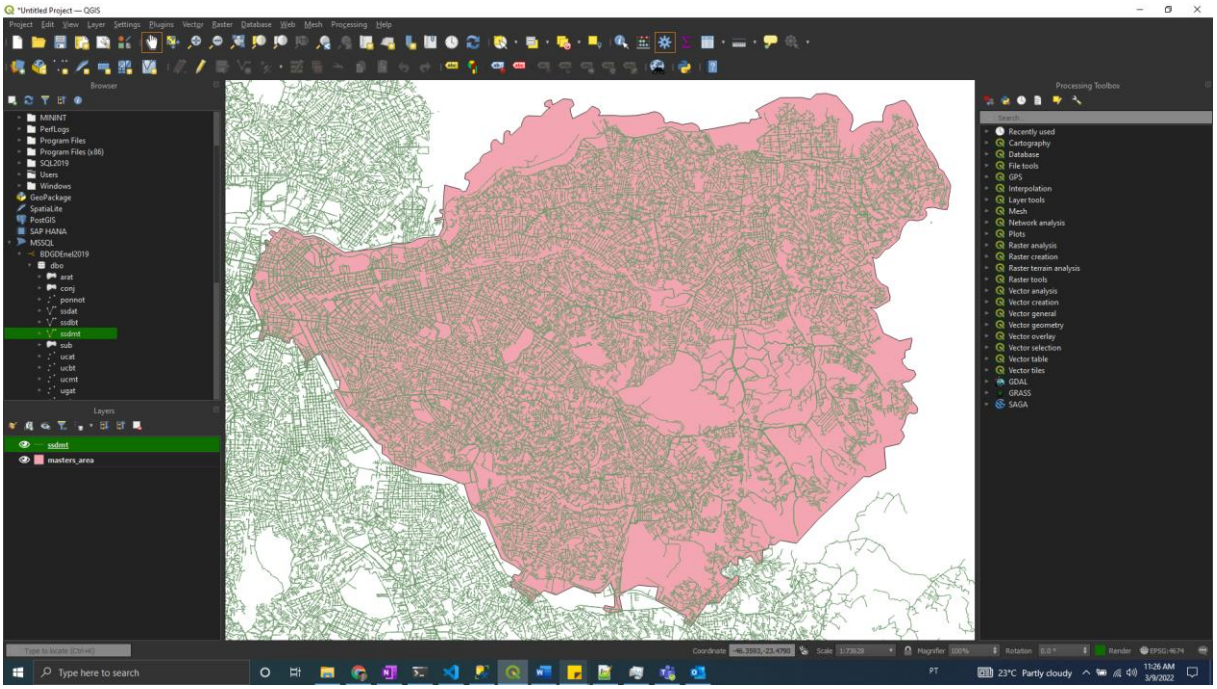


Fonte: Autor

Dados que estão carregadas as *layers* dos segmentos de média tensão (ssmt) e da área piloto, pode-se calcular a intersecção entre ambos através do QGIS, através dos seguintes passos:

- Caso já não esteja ativa, deve-se ativar a “Processing Toolbox”, clicando-se em “Processing” e depois em “Toolbox”. Aparecerá um quadro chamando “Processing Toolbox” em algum dos cantos da janela do QGIS, conforme Figura 33;

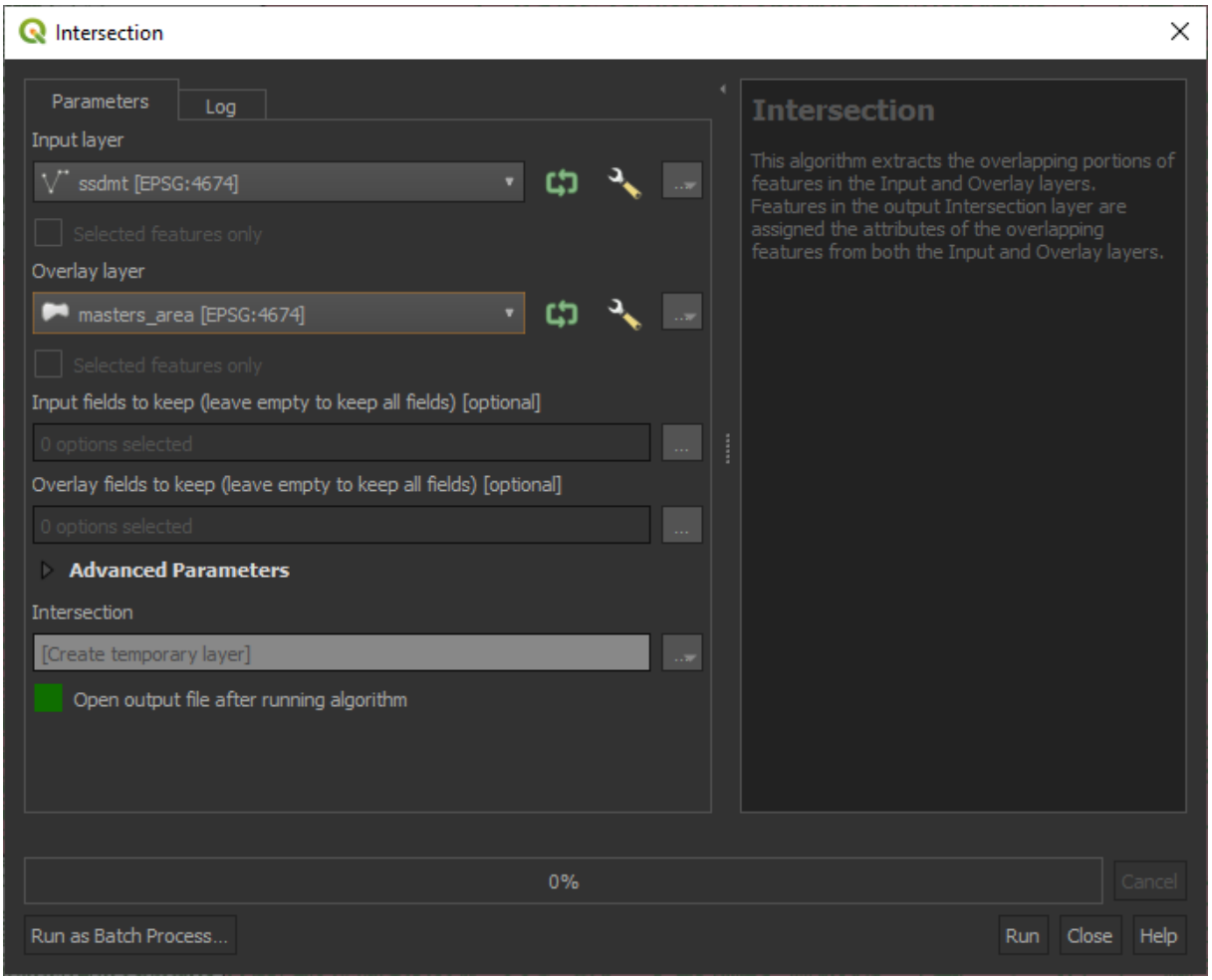
Figura 33 - Processing Toolbox no QGIS



Fonte: Autor

- No quadro “Processing Toobox”, deve-se clicar em “Vector overlay” e depois clicar duas vezes em “Intersection” e configurar a janela que se abre conforme Figura 34.Deve-se clicar em “Run” e aguardar o processamento;

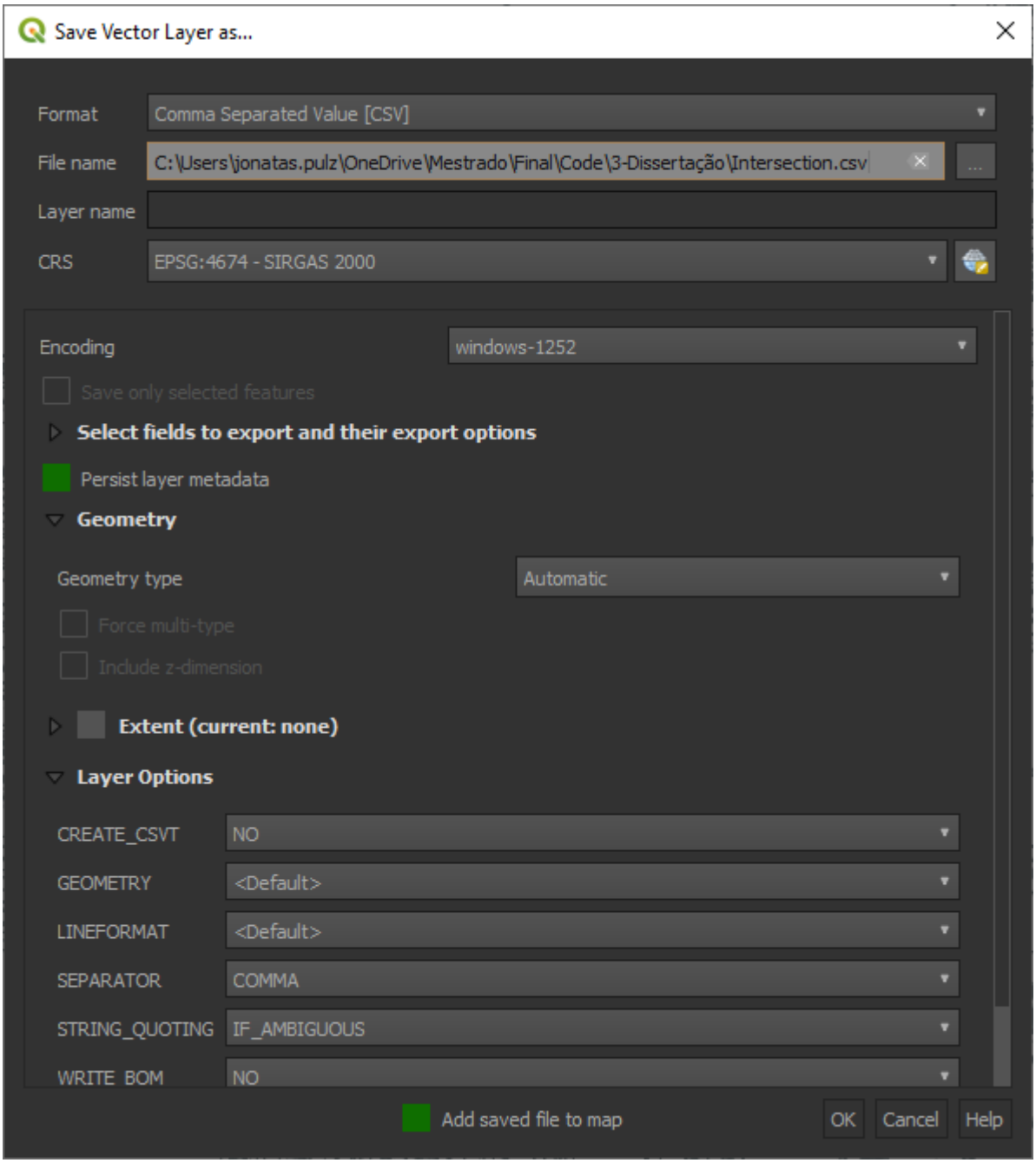
Figura 34 - Janela para cálculo da interseção entre ssmt e área piloto



Fonte: Autor

- Depois de processada a intersecção, uma nova *layer* será criada de nome “Intersection”, esta deverá ser exportada em formato CSV, e através do Excel, pode obter os valores únicos dos códigos dos alimentadores no campo “ctmt”. Para exportar a *layer* no formato CSV, basta clicar com o botão direito na *layer* criada, depois em “Export” e depois em “Save Features As...”, e basta escolher as configurações de exportação conforme Figura 35;

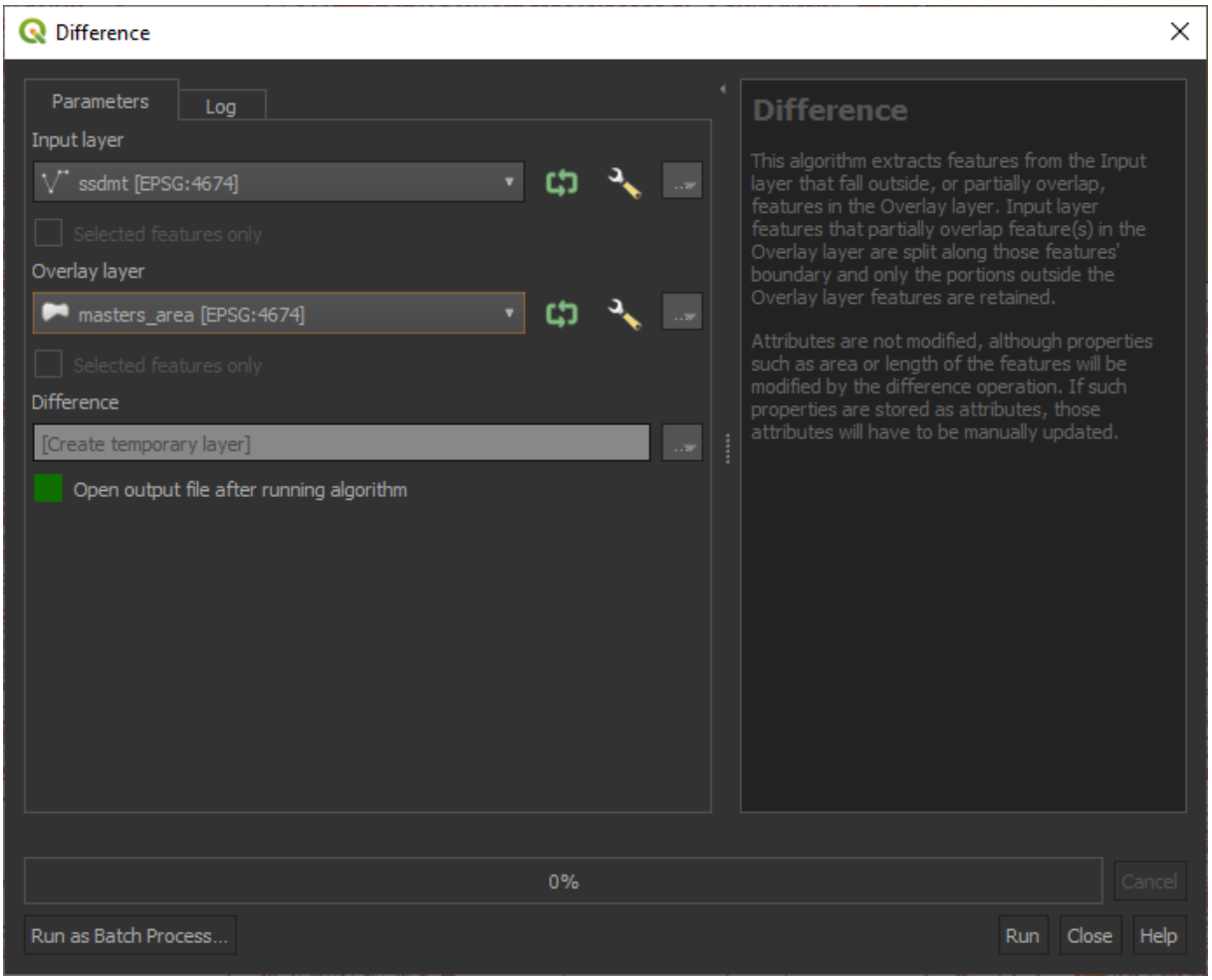
Figura 35 - Exportação da intersecção entre ssdmt e área piloto



Fonte: Autor

- Realiza-se também o processo para se determinar quais alimentadores não estão na área piloto. É importante salientar que alguns alimentadores têm intersecção com a área piloto, mas não estão completamente dentro dela. Para isso, no quadro “Processing Toobox”, deve-se clicar em “Vector overlay” e depois clicar duas vezes em “Difference” e configurar a janela que se abre conforme Figura 36;

Figura 36 - Janela para cálculo da diferença entre ssmt e área piloto



Fonte: Autor

- Exporta-se a *layer* “Difference” para um arquivo CSV, e com uma manipulação através do Excel, pode-se obter os valores únicos do campo “ctmt”. O processo de exportação é o mesmo já explicado anteriormente;
- Em posse da intersecção e da diferença com a área piloto, pode-se usar o Excel e obter os alimentadores (campo “ctmt”) que estão inteiramente na área piloto.

Apêndice F - Alocação das curvas de carga

Para o cálculo do fluxo de potência através do ProgGeoPerdas, é necessário que cada carga esteja associada a certa curva de carga. Primeiramente, deve-se obter as curvas de carga da Enel (antiga Eletropaulo) de 2019, disponíveis no arquivo compactado no link https://www.aneel.gov.br/audiencias-publicas-antigas?p_p_id=participacaopublica_WAR_participacaopublicaportlet&p_p_lifecycle=2&p_p_state=normal&p_p_mode=view&p_p_cacheability=cacheLevelPage&p_p_col_id=column-2&p_p_col_pos=1&p_p_col_count=2&_participacaopublica_WAR_participacaopublicaportlet_ideDocumento=32299&_participacaopublica_WAR_participacaopublicaportlet_tipoFaseReuniao=fase&_participacaopublica_WAR_participacaopublicaportlet_jspPage=%2Fhtml%2Fpp%2Fvisualizar.jsp , deve-se descompactar esse arquivo e o arquivo de interesse se chama “relatorioConsumidorTipo.xls”. Deve-se abrir esse arquivo através do Excel e salvá-lo em formato “xlsx” com o nome “relatorioConsumidorTipo.xls.xlsx”.

Dado que se tenha o arquivo com as curvas de carga, basta executar o notebook que está no final deste apêndice. Esse notebook realiza uma otimização para alocar da melhor forma as curvas de carga a cada consumidor. Essa otimização leva em conta que as curvas de carga de cada tipo (residencial, rural, iluminação pública, serviço público, comercial, industrial e média tensão) devem representar todos os consumidores do mesmo tipo, ou seja, supondo que existam 10 curvas de carga, a proporção de energia anual de cada curva (considerando 252 dias úteis, 52 sábados e 61 domingos e feriados) deve ser refletido na energia anual dos consumidores que são alocados para aquela curva, por exemplo: se a curva A residencial representa 20% da energia das curvas residenciais, os consumidores residenciais que são alocados nessa curva devem representar 20% da energia total dos consumidores residenciais. Para que a otimização pudesse ser realizada em tempo hábil, permitiu-se uma tolerância para a alocação de 5%, ou seja, a energia alocada para cada curva pode variar 5% para cima ou para baixo com relação a energia que ela representa dentre as curvas do mesmo tipo.

O notebook, que está no final deste apêndice, irá criar arquivos csv com os nomes “ucs_[TENSÃO]_[TIPO DE CURVA]_relaxacao-0.05.csv” e

“curves__[TENSÃO]_[TIPO DE CURVA].csv”, em que [TENSÃO] pode ser “baixa_tensao” (curvas de baixa tensão), “baixa_tensao_pip” (curvas de iluminação pública) ou “media_tensão” (curvas de média tensão); e [TIPO DE CURVA] pode ser MT (média tensão), RES (residencial), COM (comercial), IND (industrial), SP (serviço público), RUR (rural) ou PIP (iluminação pública). O primeiro tipo de arquivo contém o id de cada consumidor, coluna “cod_id”, e o id da curva alocada, coluna “curve_id_allocated”; e o segundo tipo de arquivo contém a curva em sim com 96 pontos com valores de demanda em kW.

Depois de executado o notebook (levou cerca de 50 horas para executar em um sistema i7, com 32GB de RAM e SSD), deve-se realizar a importação dos CSVs no SQL Server, para isso pode-se usar o SSMS, no banco “BDGDEnel2019” (a criação deste banco é detalhada em 5.3), gerando uma tabela para cada arquivo com o mesmo nome do arquivo (sem o “.csv”) – para isso, pode-se clicar com o botão direito no banco, depois em “Tasks” e depois em “Import Data...”. Depois, basta executar os comandos SQL abaixo para atualizar as tabelas da BDGD com as curvas corretas:

```
USE [BDGDEnel2019]
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
	[BDGDEnel2019].[dbo].[ucbt] AS T
INNER JOIN
	[BDGDEnel2019].[dbo].[ucs_baixa_tensao_COM_relaxacao-0.05] AS C
ON T.cod_id = C.cod_id
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
	[BDGDEnel2019].[dbo].[ucbt] AS T
INNER JOIN
	[BDGDEnel2019].[dbo].[ucs_baixa_tensao_IND_relaxacao-0.05] AS C
ON T.cod_id = C.cod_id
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
	[BDGDEnel2019].[dbo].[ucbt] AS T
INNER JOIN
	[BDGDEnel2019].[dbo].[ucs_baixa_tensao_RES_relaxacao-0.05] AS C
```

```
ON T.cod_id = C.cod_id
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
        [BDGDEnel2019].[dbo].[ucbt] AS T
INNER JOIN
        [BDGDEnel2019].[dbo].[ucs_baixa_tensao_RUR_relaxacao-0.05] AS C
ON T.cod_id = C.cod_id
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
        [BDGDEnel2019].[dbo].[ucbt] AS T
INNER JOIN
        [BDGDEnel2019].[dbo].[ucs_baixa_tensao_SP_relaxacao-0.05] AS C
ON T.cod_id = C.cod_id
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
        [BDGDEnel2019].[dbo].[pip] AS T
INNER JOIN
        [BDGDEnel2019].[dbo].[ucs_baixa_tensao_pip_PIP_relaxacao-0.05] AS C
ON T.cod_id = C.cod_id
GO
UPDATE T
SET T.tip_cc = C.curve_id_allocated
FROM
        [BDGDEnel2019].[dbo].[ucmt] AS T
INNER JOIN
        [BDGDEnel2019].[dbo].[ucs_media_tensao_MT_relaxacao-0.05] AS C
ON T.cod_id = C.cod_id
GO
INSERT INTO
        [BDGDEnel2019].[dbo].[StoredCrvCrgBT]
(
   [CodBase]
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
```

```
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
)
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_baixa_tensao_COM]
UNION ALL
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
```

```
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_baixa_tensao_IND]
UNION ALL
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_baixa_tensao_pip_PIP]
UNION ALL
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
```

```
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_baixa_tensao_RES]
UNION ALL
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_baixa_tensao_RUR]
UNION ALL
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
```

```
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_baixa_tensao_SP]

GO

INSERT INTO
        [BDGDEnel2019].[dbo].[StoredCrvCrgMT]
(
   [CodBase]
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
```

```
   ,[PotAtv96_kW]
)
SELECT
   390
   ,[CodCrvCrg]
   ,[TipoDia]
   ,[PotAtv01_kW],[PotAtv02_kW],[PotAtv03_kW],[PotAtv04_kW],[PotAtv05_kW]
   ,[PotAtv06_kW],[PotAtv07_kW],[PotAtv08_kW],[PotAtv09_kW],[PotAtv10_kW]
   ,[PotAtv11_kW],[PotAtv12_kW],[PotAtv13_kW],[PotAtv14_kW],[PotAtv15_kW]
   ,[PotAtv16_kW],[PotAtv17_kW],[PotAtv18_kW],[PotAtv19_kW],[PotAtv20_kW]
   ,[PotAtv21_kW],[PotAtv22_kW],[PotAtv23_kW],[PotAtv24_kW],[PotAtv25_kW]
   ,[PotAtv26_kW],[PotAtv27_kW],[PotAtv28_kW],[PotAtv29_kW],[PotAtv30_kW]
   ,[PotAtv31_kW],[PotAtv32_kW],[PotAtv33_kW],[PotAtv34_kW],[PotAtv35_kW]
   ,[PotAtv36_kW],[PotAtv37_kW],[PotAtv38_kW],[PotAtv39_kW],[PotAtv40_kW]
   ,[PotAtv41_kW],[PotAtv42_kW],[PotAtv43_kW],[PotAtv44_kW],[PotAtv45_kW]
   ,[PotAtv46_kW],[PotAtv47_kW],[PotAtv48_kW],[PotAtv49_kW],[PotAtv50_kW]
   ,[PotAtv51_kW],[PotAtv52_kW],[PotAtv53_kW],[PotAtv54_kW],[PotAtv55_kW]
   ,[PotAtv56_kW],[PotAtv57_kW],[PotAtv58_kW],[PotAtv59_kW],[PotAtv60_kW]
   ,[PotAtv61_kW],[PotAtv62_kW],[PotAtv63_kW],[PotAtv64_kW],[PotAtv65_kW]
   ,[PotAtv66_kW],[PotAtv67_kW],[PotAtv68_kW],[PotAtv69_kW],[PotAtv70_kW]
   ,[PotAtv71_kW],[PotAtv72_kW],[PotAtv73_kW],[PotAtv74_kW],[PotAtv75_kW]
   ,[PotAtv76_kW],[PotAtv77_kW],[PotAtv78_kW],[PotAtv79_kW],[PotAtv80_kW]
   ,[PotAtv81_kW],[PotAtv82_kW],[PotAtv83_kW],[PotAtv84_kW],[PotAtv85_kW]
   ,[PotAtv86_kW],[PotAtv87_kW],[PotAtv88_kW],[PotAtv89_kW],[PotAtv90_kW]
   ,[PotAtv91_kW],[PotAtv92_kW],[PotAtv93_kW],[PotAtv94_kW],[PotAtv95_kW]
   ,[PotAtv96_kW]
FROM
        [BDGDEnel2019].[dbo].[curves_media_tensao_MT]
GO
```

O *notebook* Python é apresentado abaixo (logo após a lista de pacotes com suas respectivas versões), transformado em Word através do software Pandoc. Esse *notebook* também se encontra no link em (PULZ, 2022) – arquivo “2.0-jp-alocação_curvas_de_carga.ipynb”.

```
argon2-cffi==21.3.0
argon2-cffi-bindings==21.2.0
asttokens==2.0.5
attrs==21.4.0
backcall==0.2.0
bleach==4.1.0
certifi==2021.10.8
cffi==1.15.0
click==8.0.4
click-plugins==1.1.1
cligj==0.7.2
colorama==0.4.4
cycler==0.11.0
```

debugpy==1.5.1
decorator==5.1.1
defusedxml==0.7.1
entrypoints==0.4
et-xmlfile==1.1.0
executing==0.8.3
Fiona==1.8.20
fonttools==4.29.1
GDAL==3.4.1
geopandas==0.10.2
importlib-resources==5.4.0
ipykernel==6.9.1
ipython==8.1.1
ipython-genutils==0.2.0
ipywidgets==7.6.5
jedi==0.18.1
Jinja2==3.0.3
jsonschema==4.4.0
jupyter==1.0.0
jupyter-client==7.1.2
jupyter-console==6.4.0
jupyter-core==4.9.2
jupyterlab-pygments==0.1.2
jupyterlab-widgets==1.0.2
kiwisolver==1.3.2
MarkupSafe==2.1.0
matplotlib==3.5.1
matplotlib-inline==0.1.3
mistune==0.8.4
munch==2.5.0
nbclient==0.5.11
nbconvert==6.4.2
nbformat==5.1.3
nest-asyncio==1.5.4
notebook==6.4.8
numpy==1.22.2
openpyxl==3.0.9
packaging==21.3
pandas==1.4.1
pandocfilters==1.5.0
parso==0.8.3
pickleshare==0.7.5
Pillow==9.0.1
prometheus-client==0.13.1
prompt-toolkit==3.0.28
PuLP==2.6.0
pure-eval==0.2.2
pycparser==2.21
Pygments==2.11.2
PyKrige==1.6.1

pyodbc==4.0.32
pyparsing==3.0.7
pyproj==3.3.0
pyrsistent==0.18.1
python-dateutil==2.8.2
pytz==2021.3
pywin32==303
pywinpty==2.0.3
pyzmq==22.3.0
qtconsole==5.2.2
QtPy==2.0.1
Rtree==0.9.7
scipy==1.8.0
Send2Trash==1.8.0
Shapely==1.8.1.post1
six==1.16.0
stack-data==0.2.0
terminado==0.13.2
testpath==0.6.0
tornado==6.1
traitlets==5.1.1
wcwidth==0.2.5
webencodings==0.5.1
widgetsnbextension==3.5.2
zipp==3.7.0

Alocação de Curvas de Carga

2022-03-11 Jonatas Pulz

```
import datetime
import pickle
import os

import pandas as pd
import numpy as np

import pyodbc

# https://coin-or.github.io/pulp/main/index.html
import pulp as pl

import matplotlib
from matplotlib import pyplot as plt
from IPython import display
```

Configurações

```
pd.set_option('display.max_rows', 200)
pd.set_option('display.max_columns', 200)
```

```
font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 15}

matplotlib.rc('font', **font)
```

Constantes

```
LOW_VOLTAGE_KEY = 'baixa_tensao'
LOW_VOLTAGE_PIP_KEY = 'baixa_tensao_pip'
MED_VOLTAGE_KEY = 'media_tensao'

KWH_TO_MWH_FACTOR = 1/1000.0

DAY_WORK_DAY = 'Dia útil'
DAY_SATURDAY = 'Sábado'
DAY_SUNDAY = 'Domingo'
DAY_WORK_DAY_N = 252
DAY_SATURDAY_N = 52
DAY_SUNDAY_N = 61
CURVE_N_POINTS = 96
TO_ENERGY_FACTOR = 0.25

RELAXATION = 0.05

UCS_LIMIT = 1e6
UCS_SPLIT_SIZE = 10
```

Conexão com o banco de dados

```
conn = pyodbc.connect('DRIVER={SQL Server Native Client
11.0};SERVER=localhost;DATABASE=InspectionEnel2016;Trusted_Connection=yes;')
```

Queries

```
sql_bdgd_ucbt = f'\
SELECT \
    cod_id \
    ,clas_sub \
    ,ene_01 \
    ,ene_02 \
    ,ene_03 \
    ,ene_04 \
    ,ene_05 \
    ,ene_06 \
    ,ene_07 \
    ,ene_08 \
    ,ene_09 \
    ,ene_10 \
    ,ene_11 \
    ,ene_12 \
FROM \
    [BDGDEnel2019].[dbo].[ucbt] \
'
```

```
sql_bdgd_ucmt = f'\
SELECT \
    cod_id \
    ,ene_01 \
    ,ene_02 \
    ,ene_03 \
    ,ene_04 \
    ,ene_05 \
    ,ene_06 \
    ,ene_07 \
    ,ene_08 \
    ,ene_09 \
    ,ene_10 \
    ,ene_11 \
    ,ene_12 \
FROM \
    [BDGDEnel2019].[dbo].[ucmt] \
'

sql_bdgd_pip = f'\
SELECT \
    cod_id \
    ,ene_01 \
    ,ene_02 \
    ,ene_03 \
    ,ene_04 \
    ,ene_05 \
    ,ene_06 \
    ,ene_07 \
    ,ene_08 \
    ,ene_09 \
    ,ene_10 \
    ,ene_11 \
    ,ene_12 \
FROM \
    [BDGDEnel2019].[dbo].[pip] \
'
```

Carregamento dos dados

Dados das curvas de carga de 2019

```
df_load_curve_lv_raw = pd.read_excel(
    os.path.join('.', '5. Curvas de Carga', 'relatorioConsumidorTipo.xls.xlsx')
    ,header=28
    ,engine='openpyxl'
)

df_load_curve_mv_raw = pd.read_excel(
    os.path.join('.', '5. Curvas de Carga', 'relatorioConsumidorTipo.xls.xlsx')
    ,header=143
    ,engine='openpyxl'
)

dict_df_load_curve_raw = {
    LOW_VOLTAGE_KEY: df_load_curve_lv_raw
    ,LOW_VOLTAGE_PIP_KEY: df_load_curve_lv_raw.copy()
```

```
    ,MED_VOLTAGE_KEY: df_load_curve_mv_raw
}
```

Dados de Unidades Consumidoras da BDGD de 2019

```
dict_df_bdgd_uc_raw = {}

# UCs de baixa tensão
df_bdgd_ucbt_raw = pd.read_sql(sql_bdgd_ucbt, conn)
# Ponto de iluminação pública (baixa tensão)
df_bdgd_pip_raw = pd.read_sql(sql_bdgd_pip, conn)
# UCs de média tensão
df_bdgd_ucmt_raw = pd.read_sql(sql_bdgd_ucmt, conn)

dict_df_bdgd_uc_raw[LOW_VOLTAGE_KEY] = df_bdgd_ucbt_raw
dict_df_bdgd_uc_raw[LOW_VOLTAGE_PIP_KEY] = df_bdgd_pip_raw
dict_df_bdgd_uc_raw[MED_VOLTAGE_KEY] = df_bdgd_ucmt_raw
```

Preparação dos dados

Preparação dos dados das curvas de carga

```
dict_df_load_curve = {}

delta_time = datetime.timedelta(hours = 0.5)

for k, df in dict_df_load_curve_raw.items():

    df_new = df.copy()

    current_c = None
    dict_rename = {}
    for c in df_new.columns.copy():
        if c[:len('Unnamed:')] != 'Unnamed:':
            current_c = c
        if c[:len('Unnamed:')] == 'Unnamed:':
            dict_rename[c] = current_c
    df_new.rename(dict_rename, axis=1, inplace=True)


    current_value = np.nan
    for i in range(0, df_new.shape[1]):
        if pd.notna(df_new.iloc[0, i]):
            current_value = df_new.iloc[0, i]
        if pd.isna(df_new.iloc[0, i]):
            df_new.iloc[0, i] = current_value

    for c in ['Hora', 'POSTO']:
        for i in range(0, df_new.shape[0]):
            v = df_new.iloc[i, df_new.columns.get_loc(c)].values[0]
            if pd.isna(v):
                df_new.iloc[i, df_new.columns.get_loc(c)] = ''
            if pd.notna(v):
                break
```

```
    df_new = df_new.iloc[0:np.argmax(pd.isna(df_new['Hora']).values[:, 0]),
:].copy()

    df_new.columns = pd.MultiIndex.from_arrays([
            df_new.columns.values
            ,df_new.iloc[0]
            ,df_new.iloc[1]
            ,df_new.iloc[2]
    ])
    df_new.drop([0, 1, 2], axis=0, inplace=True)

    if df_new.values.shape[0] != CURVE_N_POINTS:
        raise Exception(f'Verifique o processamento da planilha de curvas. Deve-
se ter {CURVE_N_POINTS} pontos nas curvas')

    for i, c in enumerate(df_new.columns):
        v = df_new.iloc[0, df_new.columns.get_loc(c)]
        if isinstance(v, pd.Series):
            v = v.values[0]
        if pd.isna(v):
            break
    df_new.drop(df_new.columns[i+1:], axis=1, inplace=True)

    df_new[('Hora ajustada', '', '', 'Início')] = df_new[('Hora', '', '',
'Início')].apply(
        lambda x: (datetime.datetime.combine(datetime.date(2,2,2), x) -
delta_time).time()
    )

    df_new = df_new.sort_values(
        by=('Hora ajustada', '', '', 'Início')
    ).reset_index(
    ).drop(
        ('',  'index', '', '')
        ,axis=1
    )
    for curve in df_new.columns.levels[1]:
        if curve not in {'', 'index'}:
            df_curve = df_new.xs(curve, axis=1, level=0)
            curve_type = df_curve.columns[0][0]
            df_new[(curve_type, curve, 'Agg', 'MWh')] = 0

            work_day_found = 0
            saturday_found = 0
            sunday_found = 0
            for day in df_curve.columns.levels[1]:
                if day.strip() == DAY_WORK_DAY:
                    work_day_found = 1
                    df_new[(curve_type, curve, 'Agg', 'MWh')] += \
                        df_curve[(curve_type, day,
'MW')].values*DAY_WORK_DAY_N*TO_ENERGY_FACTOR
                if day.strip() == DAY_SATURDAY:
                    saturday_found = 1
                    df_new[(curve_type, curve, 'Agg', 'MWh')] += \
                        df_curve[(curve_type, day,
'MW')].values*DAY_SATURDAY_N*TO_ENERGY_FACTOR
                if day.strip() == DAY_SUNDAY:
                    sunday_found = 1
                    df_new[(curve_type, curve, 'Agg', 'MWh')] += \
```

```
                        df_curve[(curve_type, day,
'MW')].values*DAY_SUNDAY_N*TO_ENERGY_FACTOR


            if work_day_found + saturday_found + sunday_found != 3:
                raise Exception(f'Falta algum dia para a curva {curve}')

            #print(curve)
            #display.display(df_curve.head())

    df_new.columns = df_new.columns.remove_unused_levels()
    display.display(df_new)
    dict_df_load_curve[k] = df_new
```

Preparação dos dados de consumidores

```
dict_df_bdgd_uc = {}
for voltage_type in dict_df_bdgd_uc_raw:
    df_bdgd_uc = dict_df_bdgd_uc_raw[voltage_type].copy()
    df_bdgd_uc['ene_total_MWh'] = 0
    for c in df_bdgd_uc.columns:
        if c[0:len('ene_')] == 'ene_' and c != 'ene_total_MWh':
            df_bdgd_uc['ene_total_MWh'] = df_bdgd_uc['ene_total_MWh'] +
df_bdgd_uc[c]*KWH_TO_MWH_FACTOR

    df_bdgd_uc.set_index('cod_id', inplace=True)
    dict_df_bdgd_uc[voltage_type] = df_bdgd_uc
    df_bdgd_uc.head()
```

Otimização para alocação das curvas

```
print(pl.listSolvers())
#solver = pl.GLPK(
#    path=PATH_TO_GLPK, msg=1
#    #,options = ['--tmlim', '3600']
#)
```

Modelagem do problema

```
curve_to_uc_types = {
    'Residencial': {'0', 'IP', 'RE1', 'RE2', 'REBP', 'REBR', 'REQU'}
    ,'Comercial': {'CO1', 'CO2', 'CO3', 'CO4', 'CO5', 'CO6', 'CO7', 'CO8', 'CO9',
'CPR'}
    ,'Industrial': {'IN'}
    ,'Serviço Público': {'PP1', 'PP2', 'PP3', 'SP1', 'SP2'}
    ,'Rural': {'RU1', 'RU2', 'RU3', 'RU4', 'RU5', 'RU8'}
}
curve_type_to_name = {
    'A4': 'MT'
    ,'Residencial': 'RES'
    ,'Comercial': 'COM'
    ,'Industrial': 'IND'
    ,'Serviço Público': 'SP'
    ,'Rural': 'RUR'
    ,'Iluminação Pública': 'PIP'
```

```
}
day_type_map = {
    'Dia útil': 'DU'
    ,'Sábado': 'SA'
    ,'Domingo': 'DO'
}

for voltage_type in dict_df_bdgd_uc:
    print(f'\n\n{voltage_type}
==============================================================================')

    df_ucs = dict_df_bdgd_uc[voltage_type]
    df_ucs['curve_id_allocated'] = np.nan
    df_ucs['ene_total_ratio'] = np.nan

    curve_keys = list(
        dict_df_load_curve[
            voltage_type
        ].xs(
            'Agg'
            ,axis=1
            ,level=1
        ).columns.remove_unused_levels().levels[0]
    )

    if voltage_type == LOW_VOLTAGE_KEY:
        curve_keys = [_c for _c in curve_keys if _c.strip() != 'Iluminação
Pública']
    elif voltage_type == LOW_VOLTAGE_PIP_KEY:
        curve_keys = [_c for _c in curve_keys if _c.strip() == 'Iluminação
Pública']
    elif voltage_type == MED_VOLTAGE_KEY:
        pass
    else:
        raise Exception('Tipo de tensão não reconhecida')

    for curve_key in curve_keys:
        print(f'\n{curve_key}
******************************************************************************')

        file_uc_name =
f'ucs_{voltage_type}_{curve_type_to_name[curve_key.strip()]}_relaxacao-
{RELAXATION}.csv'
        file_curve_name =
f'curves_{voltage_type}_{curve_type_to_name[curve_key.strip()]}.csv'
        if os.path.isfile(file_uc_name) and os.path.isfile(file_curve_name):
            print(f'Otimização já existe')
            continue

        if voltage_type == LOW_VOLTAGE_KEY:
            curve_keys = [_c for _c in curve_keys if _c.strip() != 'Iluminação
Pública']
            mask = df_ucs['clas_sub'].isin(curve_to_uc_types[curve_key.strip()])
        elif voltage_type == LOW_VOLTAGE_PIP_KEY:
            curve_keys = [_c for _c in curve_keys if _c.strip() == 'Iluminação
Pública']
            mask = np.full((df_ucs.shape[0], ), True)
        elif voltage_type == MED_VOLTAGE_KEY:
```

```
            mask = np.full((df_ucs.shape[0], ), True)
        else:
            raise Exception('Tipo de tensão não reconhecida')
        mask_original = mask.copy()

        # Calcula proporções de energias das curvas
        curves_ene_total = dict_df_load_curve[
            voltage_type
        ].xs(
            'Agg'
            ,axis=1
            ,level=1
        ).xs(
            curve_key
            ,axis=1
            ,level=None
        ).sum(axis=0)
        curves_ene_total.index = curves_ene_total.index.droplevel(2)
        curve_ene_total_ratio = curves_ene_total/curves_ene_total.sum()
        df_curve_ene_total_ratio = pd.DataFrame(
            curve_ene_total_ratio
            ,columns=['curve_ene_total_ratio']
        )
        display.display(df_curve_ene_total_ratio)

        # Otimiza se houver mais do que uma curva
        if df_curve_ene_total_ratio.shape[0] > 1:
            t_init = datetime.datetime.now()

            df_ucs[
                'ene_total_ratio'
            ] = df_ucs.loc[
                (mask)
                ,'ene_total_MWh'
            ]/df_ucs.loc[
                (mask)
                ,'ene_total_MWh'
            ].sum()
            display.display(df_ucs.loc[(mask)])

            # Cria grupos de forma aleatória de houver muitas UCs para conseguir
processar a otimização
            n_ucs = df_ucs.loc[
                (df_ucs['ene_total_MWh'] > 0)
                &(mask)
            ].shape[0]
            if n_ucs > UCS_LIMIT:
                print(f'Há muitas UCS: {n_ucs:,}, será realizado um agrupamento
aleatório')
                df_ucs_grouped = df_ucs.loc[
                    (df_ucs['ene_total_MWh'] > 0)
                    &(mask)
                ].reset_index().copy()

                ucs_rand_sort = np.arange(n_ucs)
                np.random.shuffle(ucs_rand_sort)

                df_ucs_grouped = df_ucs_grouped.iloc[ucs_rand_sort]
                df_ucs_grouped['group_number'] = np.nan
```

```
                for group_n, reset_idxs in
enumerate(np.array_split(df_ucs_grouped.index, n_ucs//UCS_SPLIT_SIZE)):
                    df_ucs_grouped.loc[reset_idxs, 'group_number'] = group_n
                df_ucs_grouped = df_ucs_grouped[
                    ['cod_id', 'ene_total_MWh', 'ene_total_ratio', 'group_number']
                ].groupby(
                    'group_number'
                ).agg(
                    cod_id=('cod_id', lambda x: '__'.join(sum([], x)))
                    ,cod_id_list=('cod_id', lambda x: sum([], x))
                    ,ene_total_MWh=('ene_total_MWh', np.sum)
                    ,ene_total_ratio=('ene_total_ratio', np.sum)
                ).set_index('cod_id')

                df_ucs_grouped['curve_id_allocated'] = np.nan
                mask = np.full((df_ucs_grouped.shape[0], ), True)
            else:
                df_ucs_grouped = df_ucs

            prob = pl.LpProblem(voltage_type, sense=pl.const.LpMinimize)

            dict_uc_id_constraint = {}
            dict_uc_id_curve_id_var = {}
            dict_curve_id_uc_id_var = {}
            # Cria as variáveis que indicam a qual curva o consumidor foi
associado
            for uc_id in df_ucs_grouped.loc[
                (df_ucs_grouped['ene_total_MWh'] > 0)
                &(mask)
            ].index:
                #print(uc_id)
                dict_uc_id_curve_id_var[uc_id] = {}
                for curve_id in df_curve_ene_total_ratio.index:
                    # É importate não iniciar os nomes das variáveis com valores
numéricos
                    var = pl.LpVariable(f'x_{uc_id}__{curve_id}', lowBound=0,
upBound=1, cat=pl.const.LpInteger)
                    dict_uc_id_curve_id_var[uc_id][curve_id] = var
                    dict_curve_id_uc_id_var.setdefault(curve_id, {})[uc_id] = var

                if uc_id in dict_uc_id_constraint:
                    raise Exception(f'Verifique ids de ucs repetidos: {uc_id}')
                # Cria a restrição para que o consumidor esteja associado a apenas
uma curva
                dict_uc_id_constraint[uc_id] = [
                    pl.lpSum(dict_uc_id_curve_id_var[uc_id].values()) <= 1
                    ,pl.lpSum(dict_uc_id_curve_id_var[uc_id].values()) >= 1
                ]
                prob.addConstraint(
                    dict_uc_id_constraint[uc_id][0]
                )
                prob.addConstraint(
                    dict_uc_id_constraint[uc_id][1]
                )

            # Cria a restrição para que o percentual da energia dos consumidores
seja igual ao da curva associada
            dict_curve_id_constraint = {}
            for curve_id in dict_curve_id_uc_id_var:
```

```
                dict_var_coef = {
                    dict_curve_id_uc_id_var[curve_id][uc_id]:
df_ucs_grouped.at[uc_id, 'ene_total_ratio']
                    for uc_id in dict_curve_id_uc_id_var[curve_id]
                }
                le_const = pl.LpConstraint(
                    e=pl.LpAffineExpression(
                        e=dict_var_coef
                        ,constant=-
(1+RELAXATION)*df_curve_ene_total_ratio.at[curve_id, 'curve_ene_total_ratio']
                    )
                    ,sense=pl.LpConstraintLE
                )
                ge_const = pl.LpConstraint(
                    e=pl.LpAffineExpression(
                        e=dict_var_coef
                        ,constant=-(1-
RELAXATION)*df_curve_ene_total_ratio.at[curve_id, 'curve_ene_total_ratio']
                    )
                    ,sense=pl.LpConstraintGE
                )
                dict_curve_id_constraint[curve_id] = [le_const, ge_const]
                prob.addConstraint(le_const)
                prob.addConstraint(ge_const)

            # Cria as restrições do objetivo
            obj_var = pl.LpVariable('objective', cat=pl.const.LpContinuous)
            dict_obj_coef = {
                dict_curve_id_uc_id_var[curve_id][uc_id]: df_ucs_grouped.at[uc_id,
'ene_total_ratio']
                for uc_id in dict_curve_id_uc_id_var[curve_id]
                for curve_id in dict_curve_id_uc_id_var
            }
            obj_pos_const = pl.LpConstraint(
                e=pl.LpAffineExpression(
                    e={
                        **dict_obj_coef
                        ,**{
                            obj_var: -1
                        }
                    }
                    ,constant=-1
                )
                ,sense=pl.LpConstraintLE
            )
            obj_neg_const = pl.LpConstraint(
                e=pl.LpAffineExpression(
                    e={
                        **{
                            _k: -1*_v
                            for _k, _v in dict_obj_coef.items()
                        }
                        ,**{
                            obj_var: -1
                        }
                    }
                    ,constant=+1
                )
                ,sense=pl.LpConstraintLE
```

```
            )
            prob.addConstraint(obj_pos_const)
            prob.addConstraint(obj_neg_const)

            prob.setObjective(obj_var)
            #prob.writeMPS('problem.mps')
            print('Otimizando...')
            prob.solve()
            print(pl.LpStatus[prob.status])
            print(obj_var.varValue)

            for uc_id in dict_uc_id_curve_id_var:
                found_curve = False
                for curve_id in dict_uc_id_curve_id_var[uc_id]:
                    if dict_uc_id_curve_id_var[uc_id][curve_id].varValue == 1:
                        found_curve = 1
                        df_ucs_grouped.at[uc_id, 'curve_id_allocated'] = curve_id
                        break
                if not found_curve:
                    raise Exception(f'Não encontrou curva para {uc_id}')

            # Retorna os valores para o DataFrame original se houve agrupamento
            if n_ucs > UCS_LIMIT:
                print(f'Alocando curvas para as UCs de cada grupo')
                for gruped_idx in df_ucs_grouped.index:
                    df_ucs.loc[
                        df_ucs_grouped.at[gruped_idx, 'cod_id_list']
                        ,'curve_id_allocated'
                    ] = df_ucs_grouped.at[gruped_idx, 'curve_id_allocated']

            t_final = datetime.datetime.now()
            print(f'{round((t_final-t_init).total_seconds()/60.0, 1)} ' +
                  f'minutos para otimizar com {RELAXATION} de tolerância '
            )
        else:
            print('Há apenas uma curva, não será feita otimização')

        # Atribui a primeira curva aos consumidores que tenham energia zero
        df_ucs.loc[
            (pd.isna(df_ucs['curve_id_allocated']))
            &(mask_original)
            ,'curve_id_allocated'
        ] = df_curve_ene_total_ratio.index[0]

        # Salva alocação de curvas
        df_ucs_to_save = df_ucs.loc[
            (mask_original)
            #,['ene_total_MWh', 'ene_total_ratio', 'curve_id_allocated']
        ].merge(
            df_curve_ene_total_ratio
            ,how='left'
            ,left_on='curve_id_allocated'
            ,right_index=True
        )
        df_ucs_to_save['curve_id_allocated'] =
df_ucs_to_save['curve_id_allocated'].apply(
            lambda x: curve_type_to_name[curve_key.strip()] + '_' + x
        )
        df_ucs_to_save.to_csv(
```

```
            file_uc_name
            ,index=True
            ,encoding='utf-8'
        )

        # Salve a curva em formato correto para GeoPerdas
        df_curves_to_save = dict_df_load_curve[
            voltage_type
        ].xs(
            curve_key
            ,axis=1
            ,level=None
        ).copy()
        df_curves_to_save.columns = df_curves_to_save.columns.droplevel(2)
        df_curves_to_save = df_curves_to_save.T
        df_curves_to_save.columns = [f'PotAtv{n:02d}_kW' for n in range(1,
df_curves_to_save.columns.shape[0]+1)]
        df_curves_to_save = df_curves_to_save/KWH_TO_MWH_FACTOR
        df_curves_to_save.reset_index(inplace=True)
        df_curves_to_save.rename({0: 'CodCrvCrg', 1: 'TipoDia'}, axis=1,
inplace=True)
        df_curves_to_save = df_curves_to_save.loc[
            df_curves_to_save['TipoDia'] != 'Agg'
        ]
        df_curves_to_save['CodCrvCrg'] = df_curves_to_save['CodCrvCrg'].apply(
            lambda x: curve_type_to_name[curve_key.strip()] + '_' + x
        )
        df_curves_to_save['TipoDia'] = df_curves_to_save['TipoDia'].apply(lambda
x: day_type_map[x.strip()])
        df_curves_to_save.to_csv(
            file_curve_name
            ,index=False
            ,encoding='utf-8'
        )

        # Compara razão de energia das ucs por alocação com a razão de energia
entre curvas
        df_comparison = df_ucs.loc[
            (mask_original)
            ,['ene_total_ratio', 'curve_id_allocated']
        ].groupby(
            ['curve_id_allocated']
        ).agg(
            np.sum
        ).merge(
            df_curve_ene_total_ratio
            ,how='left'
            ,left_index=True
            ,right_index=True
        )
        df_comparison['diff_perc'] = ((
            df_comparison['ene_total_ratio'] -
df_comparison['curve_ene_total_ratio']
            )/df_comparison['curve_ene_total_ratio']).apply(lambda x: round(x*100,
1))
        display.display(df_comparison)
```

Apêndice G - Passo-a-passo para obtenção do banco para cálculo de perdas através da metodologia atual regulatória a partir da BDGD

Em (ANEEL, 2021a), obtém-se um arquivo zip, que após descompactado, possui uma estrutura de pastas com scritps SQL necessários para a transformação da BDGD em um banco de dados que possa ser lido pelo ProgGeoPerdas para cálculo do fluxo de potência. Esse arquivo também se encontra no link em (PULZ, 2022) – arquivo “7. GeoPerdas ProgGeoperdas e Curvas\ Manual_BDGD_e_ANEXOS.zip”. O arquivo foi baixado em 28 de fevereiro de 2021.

Dentro das pastas, existe o caminho Manual_BDGD_e_ANEXOS\Simulação Perdas Técnicas\GeoPerdas_SIGR_20200811\ onde se encontram os scripts SQL. Abaixo são detalhadas as funções de cada um:

- scripts_curvas.7z: arquivo compactado com um script SQL que contém curvas de carga para teste. Esse arquivo não é utilizado neste trabalho. Os detalhes para alocação de curvas de carga aos consumidores da distribuidora são apresentados no 0 (antes de realizar a alocação de curvas é necessário preparar o banco, conforme é detalhado a seguir);
- script_sigr_dda_m10.sql: script que contém constantes utilizadas pela BDGD;
- script_sigr.sql: script que cria a BDGD que é utilizada pelo script “script_GeoPerdas.sql”;
- script_GeoPerdas.sql: script que cria o banco de dados que é lido pelo ProgGeoPerdas.

Antes de executar o script “script_sigr.sql”, deve-se calcular nas tabelas “eqre”, “eqtrd” e “eqtrs” um campo chamado “CodBnc” que indica a forma de ligação dos religadores, transformadores de distribuição e transformado de subestação, respectivamente. Além disso, deve-se criar uma coluna chamada “data_base” tem todas as tabelas do banco e preencher com o valor “2019” como texto. Já o preenchimento do campo chamado “CodBnc” é feito pelo código SQL a seguir que segue a orientação dada no arquivo “\Simulação Perdas Técnicas\leiame.txt” em (ANEEL, 2021a):

USE [BDGDEnel2019]

```
GO
IF NOT EXISTS (
  SELECT
    *
  FROM
    INFORMATION_SCHEMA.COLUMNS
  WHERE
    TABLE_NAME = 'eqre' AND COLUMN_NAME = 'CodBnc')
BEGIN
  ALTER TABLE [dbo].[eqre]
    ADD CodBnc int
END;
GO
IF NOT EXISTS (
  SELECT
    *
  FROM
    INFORMATION_SCHEMA.COLUMNS
  WHERE
    TABLE_NAME = 'eqtrd' AND COLUMN_NAME = 'CodBnc')
BEGIN
  ALTER TABLE [dbo].[eqtrd]
    ADD CodBnc int
END;
GO
IF NOT EXISTS (
  SELECT
    *
  FROM
    INFORMATION_SCHEMA.COLUMNS
  WHERE
    TABLE_NAME = 'eqtrs' AND COLUMN_NAME = 'CodBnc')
BEGIN
  ALTER TABLE [dbo].[eqtrs]
    ADD CodBnc int
END;
GO
UPDATE
        [dbo].[eqre]
SET CodBnc = 0
GO
WITH W_EQRE AS
(
        SELECT * FROM
                [dbo].[eqre]
        WHERE
        PAC_1 <> PAC_2 AND PAC_1 <> '0' AND PAC_2 <> '0' AND PAC_1 <> ''
AND PAC_2 <> '' AND PAC_1 IS NOT NULL AND PAC_2 IS NOT NULL
)
UPDATE
```

```
        W_EQRE
SET
        W_EQRE.CodBnc = (CASE
                                        WHEN        EQRE_SATANOJ.EQRE_NUM
IS NOT NULL THEN EQRE_SATANOJ.EQRE_NUM
                                        ELSE 1
                                END)
FROM
W_EQRE
INNER JOIN
(
        SELECT
                COD_ID,
                ROW_NUMBER() OVER (PARTITION BY PAC_1, PAC_2 ORDER BY
(SELECT NULL)) AS EQRE_NUM,
                COUNT(1) OVER (PARTITION BY PAC_1, PAC_2 ORDER BY
(SELECT NULL)) AS EQRE_COUNT
        FROM
        W_EQRE
) EQRE_SATANOJ
ON W_EQRE.COD_ID = EQRE_SATANOJ.COD_ID
WHERE EQRE_SATANOJ.EQRE_COUNT > 1;
GO
UPDATE
        [dbo].[eqtrd]
SET CodBnc = 0;
GO
WITH W_UNTR AS
(
        SELECT * FROM
                [dbo].[untrd]
        WHERE
        PAC_1 <> PAC_2 AND PAC_1 <> '0' AND PAC_2 <> '0' AND PAC_1 <> ''
AND PAC_2 <> '' AND PAC_1 IS NOT NULL AND PAC_2 IS NOT NULL
),
W_EQTR AS
(
        SELECT * FROM
                [dbo].[eqtrd]
        WHERE
        PAC_1 <> PAC_2 AND PAC_1 <> '0' AND PAC_2 <> '0' AND PAC_1 <> ''
AND PAC_2 <> '' AND PAC_1 IS NOT NULL AND PAC_2 IS NOT NULL
),
EQ AS
(
        SELECT
        W_EQTR.*,
        ROW_NUMBER()        OVER        (PARTITION        BY        W_EQTR.PAC_1,
W_EQTR.PAC_2 ORDER BY (SELECT NULL)) AS EQTR_NUM,
```

```
        COUNT(1) OVER (PARTITION BY W_EQTR.PAC_1, W_EQTR.PAC_2
ORDER BY (SELECT NULL)) AS EQTR_COUNT
        FROM
        W_EQTR
),
TRAFO AS
(
        SELECT
        EQ.COD_ID,
        EQ.DIST,
        EQ.PAC_1,
        EQ.PAC_2,
        EQ.EQTR_NUM,
        EQ.DESCR,
        W_UNTR.COD_ID AS UN_COD_ID,
        W_UNTR.TIP_TRAFO,
        W_UNTR.BANC
        FROM
        EQ
        INNER JOIN
        W_UNTR
        ON EQ.PAC_1 = W_UNTR.PAC_1 AND EQ.PAC_2 = W_UNTR.PAC_2
        WHERE EQ.EQTR_COUNT > 1
)
UPDATE
        W_EQTR
SET
        W_EQTR.CodBnc =        (CASE
                                                WHEN   T.EQTR_NUM   >   1   AND
        T.TIP_TRAFO <> 'DA' AND T.TIP_TRAFO <> 'DF' AND T.BANC = 0 AND
T.UN_COD_ID = T.DESCR THEN T.EQTR_NUM + 2
                                                ELSE T.EQTR_NUM
                                        END)
FROM
        W_EQTR
INNER JOIN
        TRAFO T
ON
        W_EQTR.COD_ID = T.COD_ID;
GO
UPDATE
        [dbo].[eqtrs]
SET CodBnc = 0;
GO
WITH W_UNTR AS
(
        SELECT * FROM
                [dbo].[untrs]
        WHERE
```

```
	PAC_1 <> PAC_2 AND PAC_1 <> '0' AND PAC_2 <> '0' AND PAC_1 <> '' AND PAC_2 <> '' AND PAC_1 IS NOT NULL AND PAC_2 IS NOT NULL
),
W_EQTR AS
(
	SELECT * FROM
		[dbo].[eqtrs]
	WHERE
	PAC_1 <> PAC_2 AND PAC_1 <> '0' AND PAC_2 <> '0' AND PAC_1 <> '' AND PAC_2 <> '' AND PAC_1 IS NOT NULL AND PAC_2 IS NOT NULL
),
EQ AS
(
	SELECT
	W_EQTR.*,
	ROW_NUMBER() OVER (PARTITION BY W_EQTR.PAC_1, W_EQTR.PAC_2 ORDER BY (SELECT NULL)) AS EQTR_NUM,
	COUNT(1) OVER (PARTITION BY W_EQTR.PAC_1, W_EQTR.PAC_2 ORDER BY (SELECT NULL)) AS EQTR_COUNT
	FROM
	W_EQTR
),
TRAFO AS
(
	SELECT
	EQ.COD_ID,
	EQ.DIST,
	EQ.PAC_1,
	EQ.PAC_2,
	EQ.EQTR_NUM,
	EQ.DESCR,
	W_UNTR.COD_ID AS UN_COD_ID,
	W_UNTR.TIP_TRAFO,
	W_UNTR.BANC
	FROM
	EQ
	INNER JOIN
	W_UNTR
	ON EQ.PAC_1 = W_UNTR.PAC_1 AND EQ.PAC_2 = W_UNTR.PAC_2
	WHERE EQ.EQTR_COUNT > 1
)
UPDATE
	W_EQTR
SET
	W_EQTR.CodBnc =	(CASE
						WHEN T.EQTR_NUM > 1 AND T.TIP_TRAFO <> 'DA' AND T.TIP_TRAFO <> 'DF' AND T.BANC = 0 AND T.UN_COD_ID = T.DESCR THEN T.EQTR_NUM + 2
						ELSE T.EQTR_NUM
					END)
```

```
FROM
        W_EQTR
INNER JOIN
        TRAFO T
ON
        W_EQTR.COD_ID = T.COD_ID
```

São realizadas alterações no script “script_sigr.sql”, para que as tabelas dentro do *schema* “sde”, sejam *views* das tabelas originais dentro do *schema* “dbo”. Não convém incluir aqui todas as modificações realizadas nesse script por serem muitas, por isso será dado apenas um exemplo. Em vez de utilizar o comando:

```
CREATE TABLE [sde].[ARAT](
        [OBJECTID] [int] NOT NULL,
        [COD_ID] [nvarchar](20) NULL,
        [DIST] [int] NULL,
        [FUN_PR] [int] NULL,
        [FUN_TE] [int] NULL,
        [DESCR] [nvarchar](254) NULL,
        [DATA_BASE] [datetime2](7) NULL,
        [DATA_CARGA] [datetime2](7) NULL,
        [Shape_STArea__] [numeric](38, 8) NOT NULL,
        [Shape_STLength__] [numeric](38, 8) NOT NULL
) ON [PRIMARY]
```

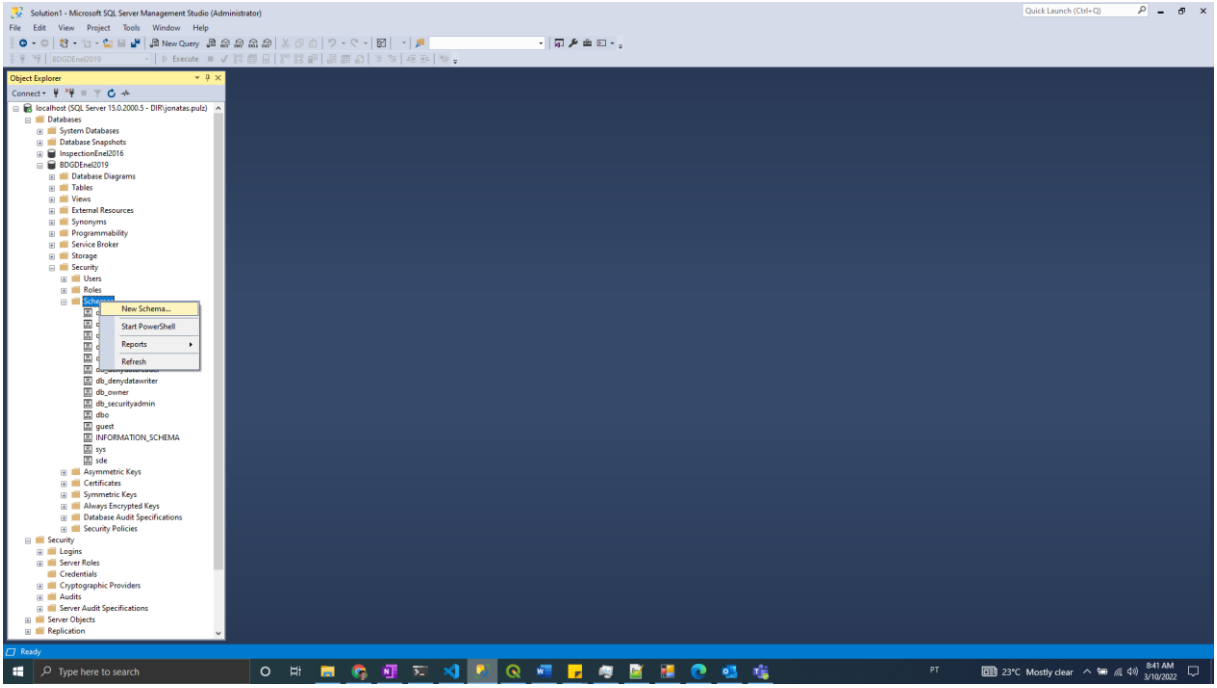
Utiliza-se o comando:

```
CREATE VIEW [sde].[ARAT] AS SELECT * FROM [dbo].[ARAT]
```

Além disso, deve-se alterar a primeira linha do script de “**USE [GEO_SIGR_PERDAS]**” para “**USE [BDGDEnel2019]**”.

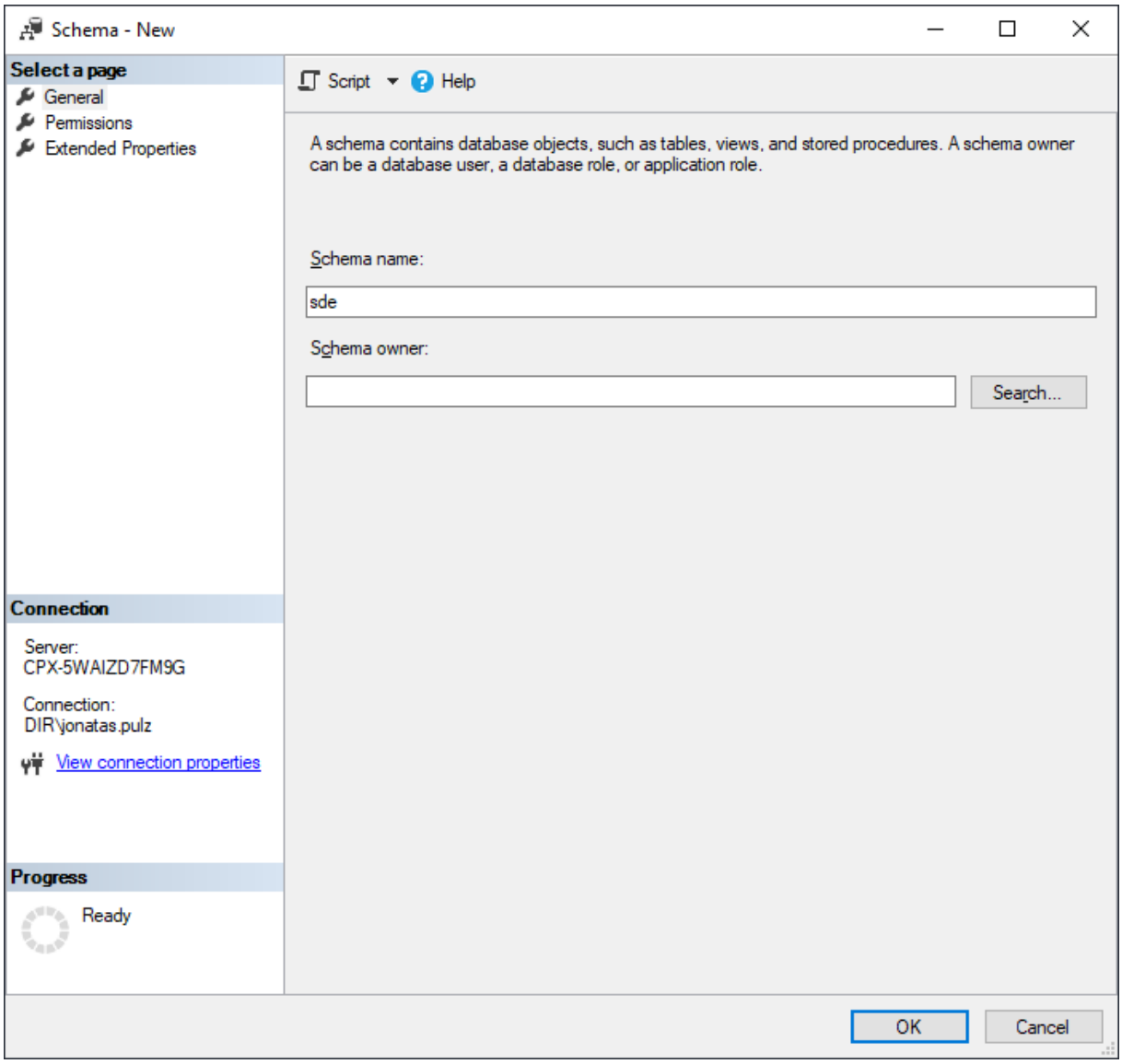
Nota-se que é necessário, antes de executar o script “script_sigr.sql”, criar o *schema* “sde” no banco de dados BDGDEnel2019 (a criação deste banco é descrita na seção 5.3). Isso pode ser feito através do SSMS, conforme Figura 37 e Figura 38. Depois de executado o script “script_sigr.sql”, é necessário alocar as curvas de carga da distribuidora a cada uma das unidades consumidoras de baixa tensão (“ucbt”, unidades consumidoras de média tensão (“ucmt”) e pontos de iluminação pública (“pip”) como detalhado no 0.

Figura 37 - Como criar o *schema* “sde” através do SSMS



Fonte: Autor

Figura 38 - Criação do *schema* “sde” no SSMS



Fonte: Autor

O próximo passo é criar o banco “GEO_SIGR_DDAD_M10”, para execução do script script_sigr_dda_m10.sql, que contém constantes utilizadas pela BDGD. Basta criar o banco através do SSMS, conforme já detalhado em outras seções deste trabalho; criar o *schema* “sde” neste banco e executar o script.

Agora é necessário criar o banco de dados que é utilizado pelo ProgGeoPerdas. Nesse banco, deve ser executado o script “script_GeoPerdas.sql”. Para o cálculo das perdas da forma regulatória, atualmente utilizada no momento de escrita deste trabalho, é criado um banco chamado GeoPerdasRegulatorio. É importante salientar

que a primeira linha do script “script_GeoPerdas.sql” deve ser alterada de “**USE [GeoPerdas2020]**” para “**USE [GeoPerdasRegulatorio]**”. Além disso, é necessário alterar a *procedure* “ExecutaCargaSIGR” do script, pois ela tem a tarefa de transformar a BDGD no formato de dados entendido pelo ProgGeoPerdas e, por padrão, a *procedure* aponta para um banco chamado GEO_SIGR_2020, por isso ela é alterada para apontar para o banco BDGDEnel2019, alterando as partes em que aparece “**GEO_SIGR_' + @AnoRef +** ’” para “**BDGDEnel' + @AnoRef + '**”. Além desses ajustes, é necessário modificar outras partes da *procedure* “ExecutaCargaSIGR”, para o correto funcionamento do script e do ProgGeoPerdas, para isso trocam-se as linhas iniciais da *procedure* “ExecutaCargaSIGR”

DECLARE @Dist bigint = CONVERT(BIGINT,SUBSTRING(ltrim(str(@CodBase)), 7, LEN(ltrim(str(@CodBase)))-6))
para
DECLARE @Dist bigint = CONVERT(BIGINT,ltrim(str(@CodBase)))
e
SET @data_base = eomonth(left(ltrim(str(@CodBase)),4) + '-' + substring(ltrim(str(@CodBase)),5,2) + '-01')
para
SET @data_base = @AnoRef”.

Altera-se também a parte da *procedure* “ExecutaCargaSIGR” onde está:

```
EXECUTE(
        'INSERT INTO [dbo].[StoredBase]
        ([CodBase]
        ,[CodDist]
        ,[Ano]
        ,[DatBas]
        ,[Descr]
        ,[COD_BDGD]
        ,[DataHora])
        SELECT DISTINCT
         ' + @CodBase + '
        ,B.[DIST]
        ,YEAR(B.[DATA_BASE])
        ,LTRIM(STR(DAY(B.[DATA_BASE]),2)) + "/" +
LTRIM(STR(MONTH(B.[DATA_BASE]),2)) + "/" +
LTRIM(STR(YEAR(B.[DATA_BASE]),4))
    ,[DESCR]
        ,N.COD_BDGD
        ,GETDATE()
        FROM GEO_SIGR_' + @AnoRef + '.sde.BASE as B LEFT JOIN
GEO_SIGR_' + @AnoRef + '.sde.NVAL_BDGD_CONSOLIDADA as N ON
(B.DIST = N.DIST AND B.DATA_BASE = N.DATA_BASE_DT)
        WHERE B.DIST = ' + @Dist + ' and B.DATA_BASE = '" +
@data_base + "'"
```

```
)
```

Para:

```
EXECUTE(
'INSERT INTO [dbo].[StoredBase]
([CodBase]
,[CodDist]
,[Ano]
,[DatBas]
,[Descr]
,[COD_BDGD]
,[DataHora])
SELECT DISTINCT
 ' + @CodBase + '
,B.[DIST]
,YEAR(B.[DATA_BASE])
,LTRIM(STR(DAY(B.[DATA_BASE]),2)) + "/" +
LTRIM(STR(MONTH(B.[DATA_BASE]),2)) + "/" +
LTRIM(STR(YEAR(B.[DATA_BASE]),4))
,[DESCR]
,"Enel2019"
,GETDATE()
FROM BDGDEnel' + @AnoRef + '.sde.BASE as B LEFT JOIN
BDGDEnel' + @AnoRef + '.sde.NVAL_BDGD_CONSOLIDADA as N ON (B.DIST =
N.DIST AND B.DATA_BASE = N.DATA_BASE_DT)
WHERE B.DIST = ' + @Dist + ' and B.DATA_BASE = '" +
@data_base + "'"
)
```

Altera-se também a parte da *procedure* “ExecutaCargaSIGR” onde está:

```
/*SET @formatted_datetime = CONVERT(char(23), GETDATE(), 121)
RAISERROR ('%s: Atualiza Ramal de StoredCargaBT...', 0, 1,
@formatted_datetime) WITH NOWAIT
EXECUTE(
'UPDATE [dbo].[StoredCargaBT]
SET [CodRmlBT] = R.COD_ID
FROM [dbo].[StoredCargaBT] AS U JOIN BDGDEnel' + @AnoRef +
'.sde.RAMLIG AS R ON (((U.CodPonAcopl = R.PAC_1) OR (U.CodPonAcopl =
R.PAC_2)) AND (U.CodBase = R.DIST))
WHERE U.CodBase = ' + @CodBase
)*/
```

Para (essa modificação atribui corretamente os ramais de ligação às cargas):

```
SET @formatted_datetime = CONVERT(char(23), GETDATE(), 121)
RAISERROR ('%s: Atualiza Ramal de StoredCargaBT, PAC_1...', 0,
1, @formatted_datetime) WITH NOWAIT
EXECUTE(
'UPDATE [dbo].[StoredCargaBT]
SET [CodRmlBT] = R.COD_ID
FROM [dbo].[StoredCargaBT] AS U JOIN BDGDEnel' + @AnoRef +
'.sde.RAMLIG AS R ON (U.CodPonAcopl = R.PAC_1) AND (U.CodBase =
R.DIST)
```

```
WHERE U.CodBase = ' + @CodBase
)
SET @formatted_datetime = CONVERT(char(23), GETDATE(), 121)
RAISERROR ('%s: Atualiza Ramal de StoredCargaBT, PAC_2...', 0, 1, @formatted_datetime) WITH NOWAIT
EXECUTE(
'UPDATE [dbo].[StoredCargaBT]
SET [CodRmlBT] = R.COD_ID
FROM [dbo].[StoredCargaBT] AS U JOIN BDGDEnel' + @AnoRef + '.sde.RAMLIG AS R ON (U.CodPonAcopl = R.PAC_2) AND (U.CodBase = R.DIST)
WHERE U.CodBase = ' + @CodBase
)
```

Além dessas modificações no script, também é necessário adicionar algumas colunas a algumas tabelas, provavelmente, a ANEEL não testou exaustivamente o script. Deve-se modificar:

```
CREATE TABLE [dbo].[StoredAuxResultado](
    [CodBase] [int] NOT NULL,
    [CodAlim] [nvarchar](100) NOT NULL,
    [TipoDiaMes] [nvarchar](4) NOT NULL,
    [TipoRodada] [nvarchar](100) NOT NULL,
    [Dias] [int] NULL,
    [Tolerancia] [decimal](18, 9) NULL,
    [PotenciaMaxInj_kW] [decimal](18, 9) NULL,
    [EnergiaInjBase_kWh] [decimal](18, 9) NULL,
    [EnergiaInj_kWh] [decimal](18, 9) NULL,
    [EnergiaForncBase_kWh] [decimal](18, 9) NULL,
    [EnergiaFornc_kWh] [decimal](18, 9) NULL,
    [PerdaPotenciaMaxTecnica_kW] [decimal](18, 9) NULL,
    [PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
    [PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
```

```
        [PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
        [EnergiaForncMT_kWh] [decimal](18, 9) NULL,
        [EnergiaForncBT_kWh] [decimal](18, 9) NULL,
        [EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
        [EnergiaPassBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
        [Convergencia] [int] NULL,
        [NumInter] [int] NULL,
 CONSTRAINT [PK_StoredAuxResultado] PRIMARY KEY CLUSTERED
(
        [CodBase] ASC,
        [CodAlim] ASC,
        [TipoDiaMes] ASC,
        [TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Para:

```
CREATE TABLE [dbo].[StoredAuxResultado](
        [CodBase] [int] NOT NULL,
        [CodAlim] [nvarchar](100) NOT NULL,
        [TipoDiaMes] [nvarchar](4) NOT NULL,
        [TipoRodada] [nvarchar](100) NOT NULL,
        [Dias] [int] NULL,
        [Tolerancia] [decimal](18, 9) NULL,
        [PotenciaMaxInj_kW] [decimal](18, 9) NULL,
        [EnergiaInjBase_kWh] [decimal](18, 9) NULL,
        [EnergiaInj_kWh] [decimal](18, 9) NULL,
        [EnergiaForncBase_kWh] [decimal](18, 9) NULL,
        [EnergiaFornc_kWh] [decimal](18, 9) NULL,
        [PerdaPotenciaMaxTecnica_kW] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
```

```
	[PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
	[EnergiaForncMT_kWh] [decimal](18, 9) NULL,
	[EnergiaForncBT_kWh] [decimal](18, 9) NULL,
	[EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
	[EnergiaPassBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaInj_per] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaFornc_per] [decimal](18, 9) NULL,
	[Convergencia] [int] NULL,
	[NumInter] [int] NULL,
 CONSTRAINT [PK_StoredAuxResultado] PRIMARY KEY CLUSTERED
(
	[CodBase] ASC,
	[CodAlim] ASC,
	[TipoDiaMes] ASC,
	[TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Também o código:

```
CREATE TABLE [dbo].[StoredAuxResultadoAno](
	[CodBase] [int] NOT NULL,
	[CodAlim] [nvarchar](100) NOT NULL,
	[TipoRodada] [nvarchar](100) NOT NULL,
	[Dias] [int] NULL,
	[EnergiaInjBase_kWh] [decimal](18, 9) NULL,
	[EnergiaInj_kWh] [decimal](18, 9) NULL,
	[DifEnergiaBaseInj_%] [decimal](18, 9) NULL,
	[EnergiaForncBase_kWh] [decimal](18, 9) NULL,
	[EnergiaFornc_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
```

```
        [PerdaEnergiaNTecnica_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaCobreTrafo_%] [decimal](18, 9) NULL,
        [PerdaEnergiaCobreTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroTrafo_%] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasBTTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
        [EnergiaForncMT_kWh] [decimal](18, 9) NULL,
        [EnergiaForncBT_kWh] [decimal](18, 9) NULL,
        [EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
        [EnergiaPassBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
        [PropAcrsBT] [decimal](18, 9) NULL,
        [NumConvergiu] [int] NULL,
        [NumDivergiu] [int] NULL,
        [NumOverflow] [int] NULL,
        [ResumoResultado] [int] NULL,
 CONSTRAINT [PK_StoredAuxResultadoAno] PRIMARY KEY CLUSTERED
(
        [CodBase] ASC,
```

```
        [CodAlim] ASC,
        [TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Para:

```
CREATE TABLE [dbo].[StoredAuxResultadoAno](
        [CodBase] [int] NOT NULL,
        [CodAlim] [nvarchar](100) NOT NULL,
        [TipoRodada] [nvarchar](100) NOT NULL,
        [Dias] [int] NULL,
        [EnergiaInjBase_kWh] [decimal](18, 9) NULL,
        [EnergiaInj_kWh] [decimal](18, 9) NULL,
        [DifEnergiaBaseInj_%] [decimal](18, 9) NULL,
        [EnergiaForncBase_kWh] [decimal](18, 9) NULL,
        [EnergiaFornc_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaCobreTrafo_%] [decimal](18, 9) NULL,
        [PerdaEnergiaCobreTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroTrafo_%] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTTotal_%] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
```

```
	[PerdaEnergiaLinhasBTTotal_%] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
	[EnergiaForncMT_kWh] [decimal](18, 9) NULL,
	[EnergiaForncBT_kWh] [decimal](18, 9) NULL,
	[EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
	[EnergiaPassBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaInj_per] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_EnergiaInj_per] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaFornc_per] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_EnergiaFornc_per] [decimal](18, 9) NULL,
	[PropAcrsBT] [decimal](18, 9) NULL,
	[NumConvergiu] [int] NULL,
	[NumDivergiu] [int] NULL,
	[NumOverflow] [int] NULL,
	[ResumoResultado] [int] NULL,
 CONSTRAINT [PK_StoredAuxResultadoAno] PRIMARY KEY CLUSTERED
(
	[CodBase] ASC,
	[CodAlim] ASC,
	[TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Também o código:

```
CREATE TABLE [dbo].[StoredAuxResultadoMes](
	[CodBase] [int] NOT NULL,
	[CodAlim] [nvarchar](100) NOT NULL,
	[TipoMes] [nvarchar](2) NOT NULL,
	[TipoRodada] [nvarchar](100) NOT NULL,
	[Dias] [int] NULL,
	[EnergiaInjBase_kWh] [decimal](18, 9) NULL,
	[EnergiaInj_kWh] [decimal](18, 9) NULL,
	[EnergiaForncBase_kWh] [decimal](18, 9) NULL,
	[EnergiaFornc_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
```

```
	[PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
	[EnergiaForncMT_kWh] [decimal](18, 9) NULL,
	[EnergiaForncBT_kWh] [decimal](18, 9) NULL,
	[EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
	[EnergiaPassBT_kWh] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
	[PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
	[PerdaEnergiaNTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
	[PropAcrsBT] [decimal](18, 9) NULL,
	[NumConvergiu] [int] NULL,
	[NumDivergiu] [int] NULL,
	[NumOverflow] [int] NULL,
 CONSTRAINT [PK_StoredAuxResultadoMes] PRIMARY KEY CLUSTERED
(
	[CodBase] ASC,
	[CodAlim] ASC,
	[TipoMes] ASC,
	[TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Para:

```
CREATE TABLE [dbo].[StoredAuxResultadoMes](
	[CodBase] [int] NOT NULL,
	[CodAlim] [nvarchar](100) NOT NULL,
	[TipoMes] [nvarchar](2) NOT NULL,
	[TipoRodada] [nvarchar](100) NOT NULL,
	[Dias] [int] NULL,
	[EnergiaInjBase_kWh] [decimal](18, 9) NULL,
```

```
        [EnergiaInj_kWh] [decimal](18, 9) NULL,
        [EnergiaForncBase_kWh] [decimal](18, 9) NULL,
        [EnergiaFornc_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
        [EnergiaForncMT_kWh] [decimal](18, 9) NULL,
        [EnergiaForncBT_kWh] [decimal](18, 9) NULL,
        [EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
        [EnergiaPassBT_kWh] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_EnergiaFornc_%] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaInj_per] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_EnergiaInj_per] [decimal](18, 9) NULL,
        [PerdaEnergiaTecnica_EnergiaFornc_per] [decimal](18, 9) NULL,
        [PerdaEnergiaNTecnica_EnergiaFornc_per] [decimal](18, 9) NULL,
        [PropAcrsBT] [decimal](18, 9) NULL,
        [NumConvergiu] [int] NULL,
        [NumDivergiu] [int] NULL,
        [NumOverflow] [int] NULL,
 CONSTRAINT [PK_StoredAuxResultadoMes] PRIMARY KEY CLUSTERED
(
        [CodBase] ASC,
```

```
        [CodAlim] ASC,
        [TipoMes] ASC,
        [TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Altera-se também a *procedure* “iadTabelaOperacao”, para incluir colunas que faltam às tabelas, mudando o código:

```
EXECUTE (
                'CREATE TABLE [dbo].[' + @CodBase + 'AuxResultadoMes](
                [CodBase] [int] NOT NULL,
                [CodAlim] [nvarchar](100) NOT NULL,
                [TipoMes] [nvarchar](2) NOT NULL,
                [TipoRodada] [nvarchar](100) NOT NULL,
                [Dias] [int] NULL,
                [EnergiaInjBase_kWh] [decimal](18, 9) NULL,
                [EnergiaInj_kWh] [decimal](18, 9) NULL,
                [EnergiaForncBase_kWh] [decimal](18, 9) NULL,
                [EnergiaFornc_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaNTecnica_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
                [PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
```

```
[EnergiaForncMT_kWh] [decimal](18, 9) NULL,
[EnergiaForncBT_kWh] [decimal](18, 9) NULL,
[EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
[EnergiaPassBT_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
[PerdaEnergiaNTecnica_EnergiaInj_%] [decimal](18, 9)
NULL,
[PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9)
NULL,
[PerdaEnergiaNTecnica_EnergiaFornc_%] [decimal](18, 9)
NULL,
[PropAcrsBT] [decimal](18, 9) NULL,
[NumConvergiu] [int] NULL,
[NumDivergiu] [int] NULL,
[NumOverflow] [int] NULL,
CONSTRAINT [PK_AuxResultadoMes' + @CodBase + ']
PRIMARY KEY CLUSTERED
(
[CodBase] ASC,
[CodAlim] ASC,
[TipoMes] ASC,
[TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]'
)
```

Para:

```
EXECUTE (
'CREATE TABLE [dbo].[' + @CodBase + 'AuxResultadoMes](
[CodBase] [int] NOT NULL,
[CodAlim] [nvarchar](100) NOT NULL,
[TipoMes] [nvarchar](2) NOT NULL,
[TipoRodada] [nvarchar](100) NOT NULL,
[Dias] [int] NULL,
[EnergiaInjBase_kWh] [decimal](18, 9) NULL,
[EnergiaInj_kWh] [decimal](18, 9) NULL,
[EnergiaForncBase_kWh] [decimal](18, 9) NULL,
[EnergiaFornc_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTecnica_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaNTecnica_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaCobre_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerro_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafo_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroA3aA3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoA3aA3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroA3aA4_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoA3aA4_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroA3aB_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoA3aB_kWh] [decimal](18, 9) NULL,
```

```
[PerdaEnergiaFerroA4A4_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoA4A4_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroA4B_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoA4B_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroA4A3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoA4A3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroBA3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoBA3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaFerroBA4_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTrafoBA4_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaLinhas_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaMT_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaBT_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaLinhasMT_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaLinhasBT_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaLinhasMTA3a_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaLinhasMTA4_kWh] [decimal](18, 9) NULL,
[EnergiaForncMT_kWh] [decimal](18, 9) NULL,
[EnergiaForncBT_kWh] [decimal](18, 9) NULL,
[EnergiaPassTrafo_kWh] [decimal](18, 9) NULL,
[EnergiaPassBT_kWh] [decimal](18, 9) NULL,
[PerdaEnergiaTecnica_EnergiaInj_%] [decimal](18, 9) NULL,
[PerdaEnergiaNTecnica_EnergiaInj_%] [decimal](18, 9)
NULL,
[PerdaEnergiaTecnica_EnergiaFornc_%] [decimal](18, 9)
NULL,
[PerdaEnergiaNTecnica_EnergiaFornc_%] [decimal](18, 9)
NULL,
[PerdaEnergiaTecnica_EnergiaInj_per] [decimal](18, 9)
NULL,
[PerdaEnergiaNTecnica_EnergiaInj_per] [decimal](18, 9)
NULL,
[PerdaEnergiaTecnica_EnergiaFornc_per] [decimal](18, 9)
NULL,
[PerdaEnergiaNTecnica_EnergiaFornc_per] [decimal](18, 9)
NULL,
[PropAcrsBT] [decimal](18, 9) NULL,
[NumConvergiu] [int] NULL,
[NumDivergiu] [int] NULL,
[NumOverflow] [int] NULL,
CONSTRAINT [PK_AuxResultadoMes' + @CodBase + ']
PRIMARY KEY CLUSTERED
(
[CodBase] ASC,
[CodAlim] ASC,
[TipoMes] ASC,
[TipoRodada] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
                ) ON [PRIMARY]'
                )
```

Altera-se também a *procedure* “iadAuxTensaoBaseBT”, fazendo com que o comando *insert* seja:

```
EXECUTE(
        'INSERT INTO [dbo].[AuxTensaoBaseBT]
        ([CodBase]
        ,[CodAlim]
        ,[CodPonAcopl]
        ,[TnsFasBas_kV])
        SELECT DISTINCT
            [CodBase]
            ,[CodAlim]
            ,[CodPonAcopl]
            ,[TnsFasBas_kV]
        FROM
            (
                SELECT
                    T1.*
                    ,ROW_NUMBER() OVER (PARTITION BY [CodBase], [CodAlim], [CodPonAcopl] ORDER BY [TnsFasBas_kV] DESC) AS rn
                FROM
                    (
                        SELECT DISTINCT [CodBase]
                        ,[CodAlim]
                        ,[De] AS [CodPonAcopl]
                        ,[TnsFasBas1_kV] AS [TnsFasBas_kV]
                        FROM [dbo].[' + @CodBase + 'AuxTramo]
                        WHERE [CodBase] =' + @CodBase + ' AND [TnsFasBas1_kV] IS NOT NULL
                        UNION
                        SELECT DISTINCT [CodBase]
                        ,[CodAlim]
                        ,[Para] AS [CodPonAcopl]
                        ,[TnsFasBas2_kV] AS [TnsFasBas_kV]
                        FROM [dbo].[' + @CodBase + 'AuxTramo]
                        WHERE [CodBase] = ' + @CodBase + ' AND [TnsFasBas2_kV] IS NOT NULL
                    ) AS T1
            ) AS T2
        WHERE
            rn = 1'
        )
```

Também, deve-se alterar a função “Max24Patamares”, para que seja computacionalmente mais eficiente, trocando-se todos os “**IF (@Var...**” por:

```
SELECT @Resultado = MAX(result)
        FROM
        (VALUES
                (0.0)
                ,(@Var01)
                ,(@Var02)
                ,(@Var03)
                ,(@Var04)
                ,(@Var05)
                ,(@Var06)
                ,(@Var07)
                ,(@Var08)
                ,(@Var09)
                ,(@Var10)
                ,(@Var11)
                ,(@Var12)
                ,(@Var13)
                ,(@Var14)
                ,(@Var15)
                ,(@Var16)
                ,(@Var17)
                ,(@Var18)
                ,(@Var19)
                ,(@Var20)
                ,(@Var21)
                ,(@Var22)
                ,(@Var23)
                ,(@Var24)
        ) X(result)
```

Também, deve-se alterar a *procedure* “ExecutaCargaOperacao”, trocando-se o comando de:

```
EXECUTE(
                'INSERT INTO [dbo].[' + @CodBase + 'Base]
                ([CodBase]
                ,[CodDist]
                ,[Ano]
                ,[DatBas]
                ,[Descr])
                SELECT [CodBase]
                ,[CodDist]
                ,[Ano]
                ,[DatBas]
                ,[Descr]
                FROM [dbo].[StoredBase]
                WHERE [CodBase] = ' + @CodBase
                )
```

Para:

```
EXECUTE(
        'INSERT INTO [dbo].[' + @CodBase + 'Base]
        ([CodBase]
        ,[CodDist]
        ,[Ano]
        ,[DatBas]
        ,[Descr]
        ,[COD_BDGD])
        SELECT [CodBase]
        ,[CodDist]
        ,[Ano]
        ,[DatBas]
        ,[Descr]
        ,[COD_BDGD]
        FROM [dbo].[StoredBase]
        WHERE [CodBase] = ' + @CodBase
        )
```

Essas modificações são necessárias para a correta execução dos códigos SQL e do ProgGeoPerdas.

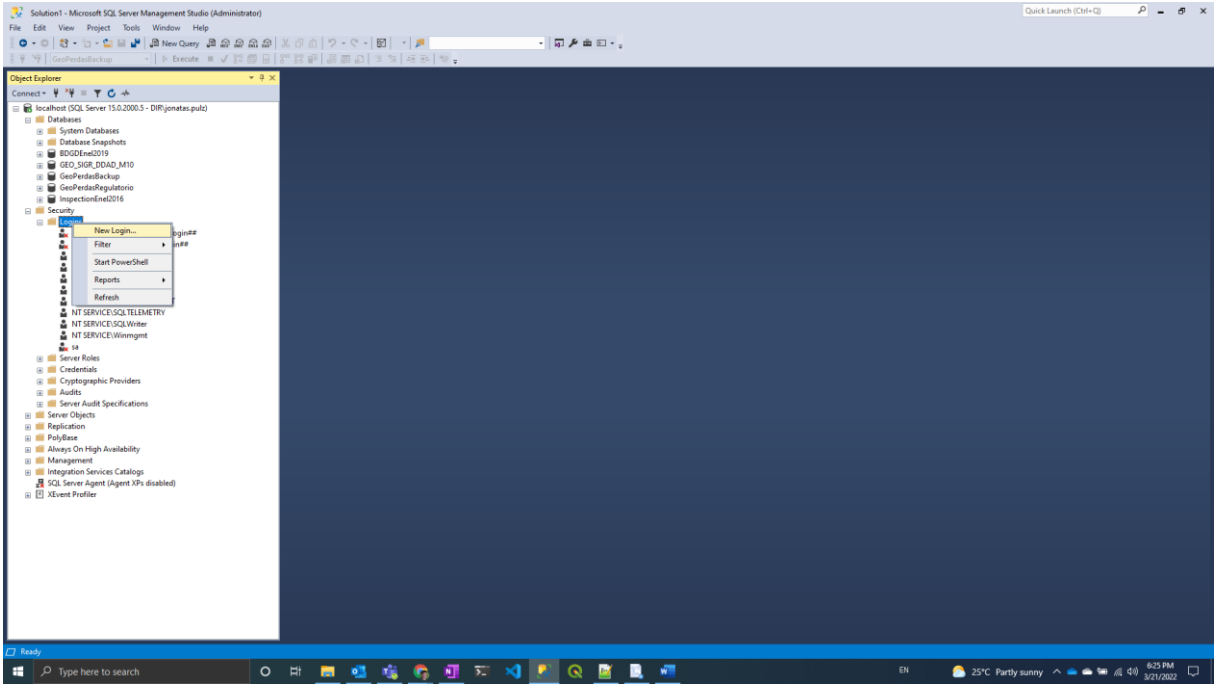
Depois de realizadas essas modificações no script “script_GeoPerdas.sql”, pode-se executá-lo no banco GeoPerdasRegulatorio. Os scripts já modificados, conforme detalhado nesse apêndice, podem ser encontrados em (PULZ, 2022), na pasta “7. GeoPerdas ProgGeoperdas e Curvas\2. GeoPerdas_SIGR_20200811\2. Original - Ajustado”. Após executar o script, deve-se executar a *procedure* “Principal” e a “Recarrega”, que realiza todo o processamento prévio necessário para se executar o ProgGeoPerdas, isso é feito através do comando SQL:

```
EXEC [dbo].[Principal] 390, ‘2019’, 1;
EXEC [dbo].[Recarrega] 390;
```

Esse comando levou 9 horas para ser executado num sistema i7, com 32GB de RAM e SSD, além de serem necessários 100GB de espaço livre em disco.

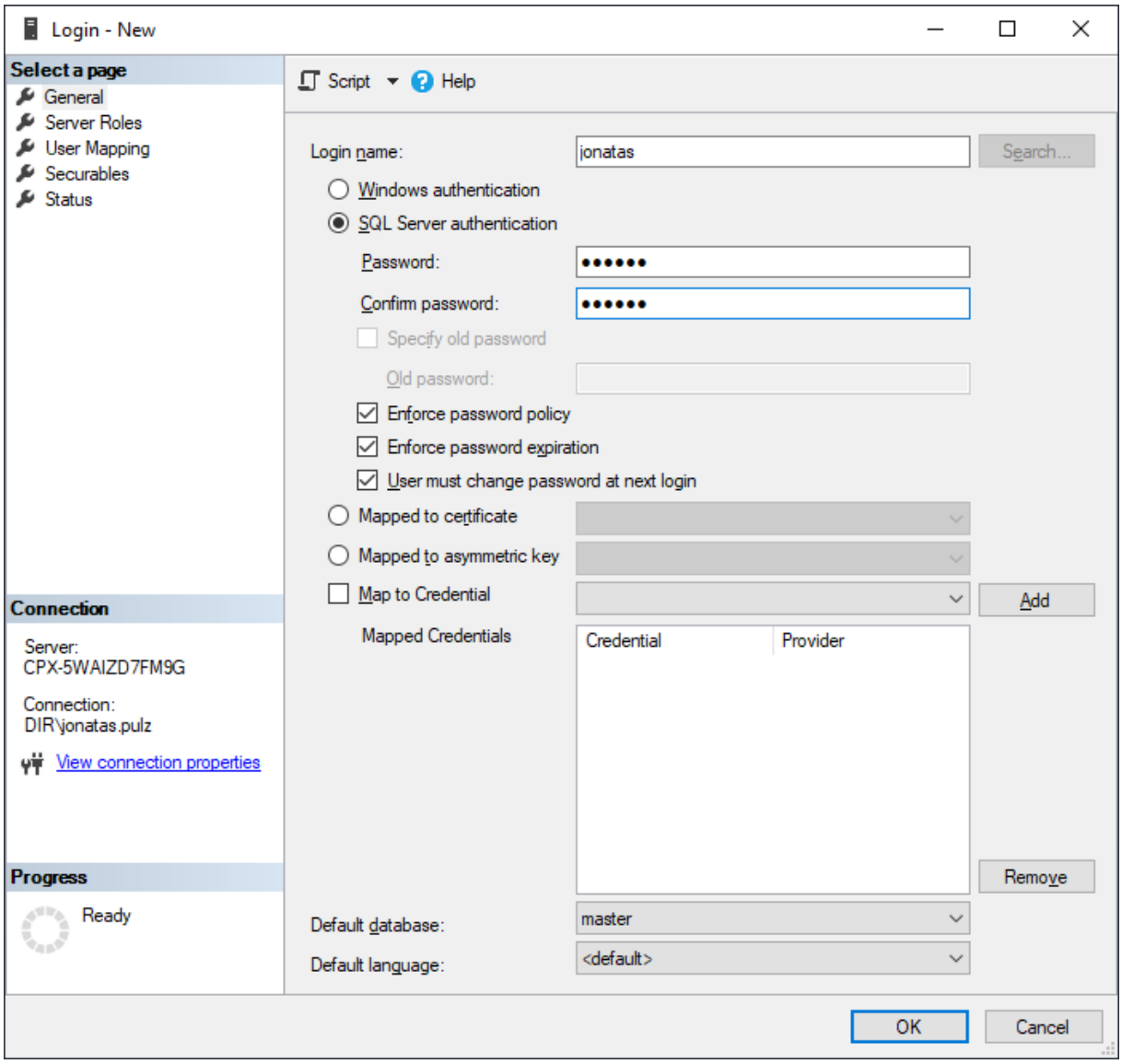
É necessário criar um login para que o ProgGeoPerdas possa se logar no servidor e executar os cálculos de perdas. Isso pode ser feito através do SSMS, como mostram Figura 39, Figura 40 e Figura 41.

Figura 39 - Menu para criar login



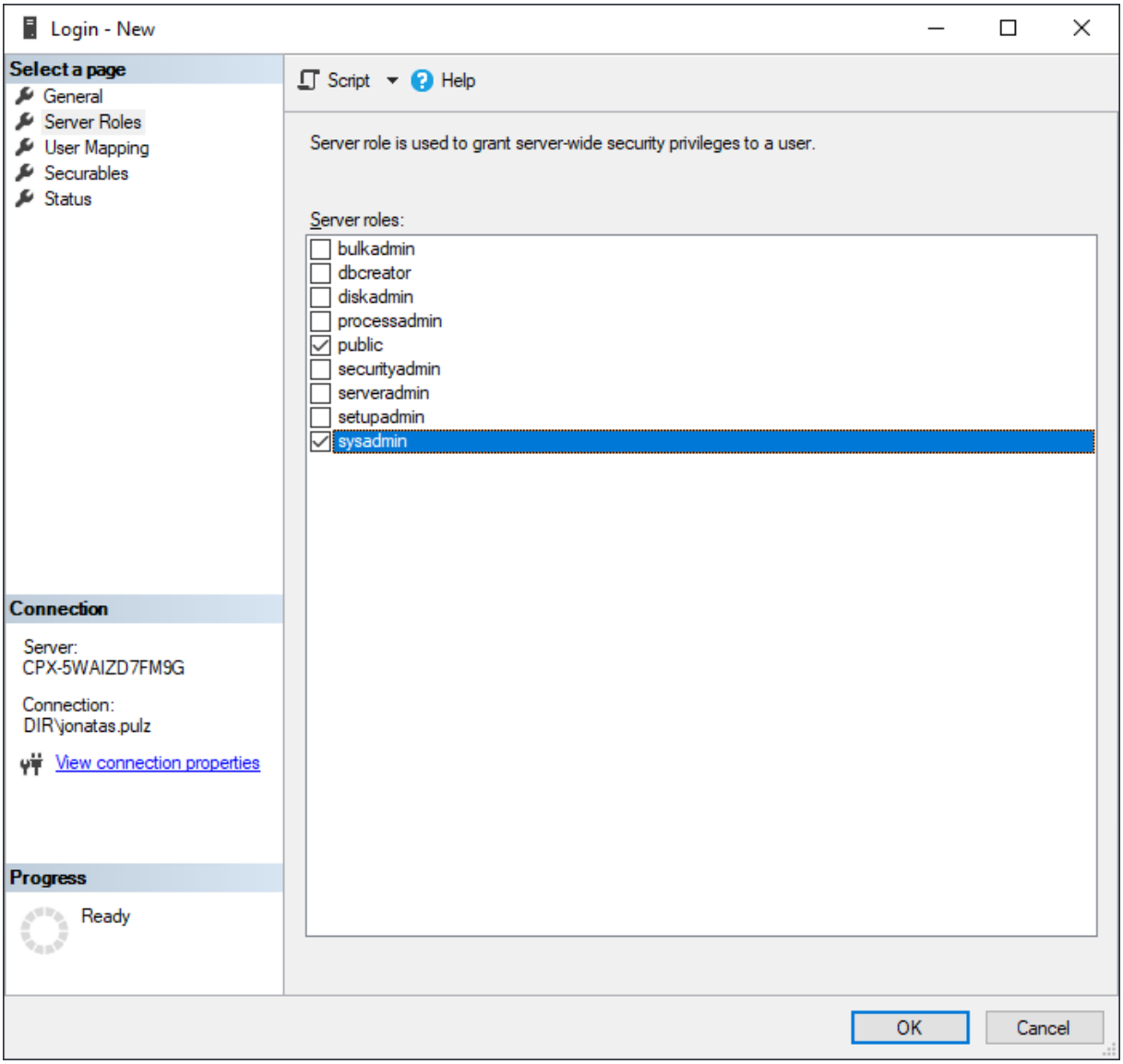
Fonte: Autor

Figura 40 - Como criar login



Fonte: Autor

Figura 41 - Configuração necessária para criar login



Fonte: Autor

Apêndice H - Passo-a-passo para obtenção do banco para cálculo de perdas através da metodologia atual regulatória a partir dos dados da 5ª Revisão Tarifária Periódica

Para a obtenção do banco de dados para utilização do ProgGeoPerdas a partir dos dados é necessário realizar todas as modificações indicadas no 0 para o script "script_GeoPerdas.sql" e mais outras modificações indicadas a seguir.

Primeiramente, deve-se alterar o banco para o qual o script aponta logo no início, mudando o:

USE [GeoPerdasRegulatorio]

Para:
USE [GeoPerdasRTPRegulatorio]

Deve-se alterar a criação da tabela "StoredBase" de:

```
CREATE TABLE [dbo].[StoredBase](
	[CodBase] [bigint] NOT NULL,
	[CodDist] [int] NULL,
	[Ano] [int] NULL,
	[DatBas] [char](10) NULL,
	[Descr] [text] NULL,
	[COD_BDGD] [nvarchar](40) NOT NULL,
	[DataHora] [datetime] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Para:

```
CREATE TABLE [dbo].[StoredBase](
	[CodBase] [bigint] NOT NULL,
	[CodDist] [int] NULL,
	[Ano] [int] NULL,
	[DatBas] [char](10) NULL,
	[Descr] [text] NULL,
	[COD_BDGD] [nvarchar](40) NULL,
	[DataHora] [datetime] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Deve-se alterar a *procedure* "iadTabelaOperacao" em:

```
EXECUTE(
		'CREATE TABLE [dbo].[' + @CodBase + 'Base](
		[CodBase] [bigint] NOT NULL,
		[CodDist] [int] NULL,
		[Ano] [int] NULL,
		[DatBas] [char](10) NULL,
		[Descr] [text] NULL,
```

```
[COD_BDGD] [nvarchar](40) NOT NULL,
[DataHora] [datetime] NULL)'
)
```

Para:

```
EXECUTE(
'CREATE TABLE [dbo].[' + @CodBase + 'Base](
[CodBase] [bigint] NOT NULL,
[CodDist] [int] NULL,
[Ano] [int] NULL,
[DatBas] [char](10) NULL,
[Descr] [text] NULL,
[COD_BDGD] [nvarchar](40) NULL,
[DataHora] [datetime] NULL)'
)
```

Deve-se alterar a criação da tabela “StoredCodCondutor” de:

```
CREATE TABLE [dbo].[StoredCodCondutor](
	[CodBase] [int] NOT NULL,
	[CodCond] [nvarchar](100) NOT NULL,
	[Resis_ohms_km] [decimal](18, 9) NULL,
	[Reat_ohms_km] [decimal](18, 9) NULL,
	[CorrMax_A] [decimal](18, 9) NULL,
	[GMR_km] [decimal](18, 9) NULL,
	[Rac_ohm] [decimal](18, 9) NULL,
	[Diam] [decimal](18, 9) NULL,
	[Cond1_x_m] [decimal](18, 9) NULL,
	[Cond1_y_m] [decimal](18, 9) NULL,
	[Cond2_x_m] [decimal](18, 9) NULL,
	[Cond2_y_m] [decimal](18, 9) NULL,
	[Cond3_x_m] [decimal](18, 9) NULL,
	[Cond3_y_m] [decimal](18, 9) NULL,
	[Cond4_x_m] [decimal](18, 9) NULL,
	[Cond4_y_m] [decimal](18, 9) NULL,
	[Descr] [text] NULL,
 CONSTRAINT [PK_StoredCodCondutor] PRIMARY KEY CLUSTERED
(
	[CodBase] ASC,
	[CodCond] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Para:

```
CREATE TABLE [dbo].[StoredCodCondutor](
	[CodBase] [int] NOT NULL,
	[CodCond] [nvarchar](100) NOT NULL,
	[Resis_ohms_km] [decimal](18, 9) NULL,
	[Reat_ohms_km] [decimal](18, 9) NULL,
```

```
        [CorrMax_A] [decimal](18, 9) NULL,
        [GMR_km] [decimal](18, 9) NULL,
        [Rac_ohm] [decimal](18, 9) NULL,
        [Diam] [decimal](18, 9) NULL,
        [Cond1_x_m] [decimal](18, 9) NULL,
        [Cond1_y_m] [decimal](18, 9) NULL,
        [Cond2_x_m] [decimal](18, 9) NULL,
        [Cond2_y_m] [decimal](18, 9) NULL,
        [Cond3_x_m] [decimal](18, 9) NULL,
        [Cond3_y_m] [decimal](18, 9) NULL,
        [Cond4_x_m] [decimal](18, 9) NULL,
        [Cond4_y_m] [decimal](18, 9) NULL,
        [Descr] nvarchar(255) NULL,
 CONSTRAINT [PK_StoredCodCondutor] PRIMARY KEY CLUSTERED
(
        [CodBase] ASC,
        [CodCond] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Deve-se alterar a *procedure* “iadTabelaOperacao” em:

```
EXECUTE(
                'CREATE TABLE [dbo].[' + @CodBase + 'CodCondutor](
                [CodBase] [int] NOT NULL,
                [CodCond] [nvarchar](100) NOT NULL,
                [Resis_ohms_km] [decimal](18, 9) NULL,
                [Reat_ohms_km] [decimal](18, 9) NULL,
                [CorrMax_A] [decimal](18, 9) NULL,
                [GMR_km] [decimal](18, 9) NULL,
                [Rac_ohm] [decimal](18, 9) NULL,
                [Diam] [decimal](18, 9) NULL,
                [Cond1_x_m] [decimal](18, 9) NULL,
                [Cond1_y_m] [decimal](18, 9) NULL,
                [Cond2_x_m] [decimal](18, 9) NULL,
                [Cond2_y_m] [decimal](18, 9) NULL,
                [Cond3_x_m] [decimal](18, 9) NULL,
                [Cond3_y_m] [decimal](18, 9) NULL,
                [Cond4_x_m] [decimal](18, 9) NULL,
                [Cond4_y_m] [decimal](18, 9) NULL,
                [Descr] [text] NULL,
                CONSTRAINT [PK_CodCondutor' + @CodBase + '] PRIMARY
KEY CLUSTERED
                (
                [CodBase] ASC,
                [CodCond] ASC
                )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]'
)
```

Para:

```
EXECUTE(
'CREATE TABLE [dbo].[' + @CodBase + 'CodCondutor](
[CodBase] [int] NOT NULL,
[CodCond] [nvarchar](100) NOT NULL,
[Resis_ohms_km] [decimal](18, 9) NULL,
[Reat_ohms_km] [decimal](18, 9) NULL,
[CorrMax_A] [decimal](18, 9) NULL,
[GMR_km] [decimal](18, 9) NULL,
[Rac_ohm] [decimal](18, 9) NULL,
[Diam] [decimal](18, 9) NULL,
[Cond1_x_m] [decimal](18, 9) NULL,
[Cond1_y_m] [decimal](18, 9) NULL,
[Cond2_x_m] [decimal](18, 9) NULL,
[Cond2_y_m] [decimal](18, 9) NULL,
[Cond3_x_m] [decimal](18, 9) NULL,
[Cond3_y_m] [decimal](18, 9) NULL,
[Cond4_x_m] [decimal](18, 9) NULL,
[Cond4_y_m] [decimal](18, 9) NULL,
[Descr] nvarchar(255) NULL,
CONSTRAINT [PK_CodCondutor' + @CodBase + '] PRIMARY
KEY CLUSTERED
(
[CodBase] ASC,
[CodCond] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]'
)
```

Deve-se alterar a *procedure* “ExecutaCargaOperacao” em:

```
EXECUTE(
'INSERT INTO [dbo].[' + @CodBase + 'Base]
(
[CodBase]
,[CodDist]
,[Ano]
,[DatBas]
,[Descr]
,[COD_BDGD]
,[DataHora]
)
SELECT [CodBase]
,[CodDist]
,[Ano]
,[DatBas]
```

```
,[Descr]
,[COD_BDGD]
,[DataHora]
FROM [dbo].[StoredBase]
WHERE [CodBase] = ' + @CodBase + ' and datahora = (
        select max(datahora) FROM [dbo].[StoredBase]
        WHERE [CodBase] = ' + @CodBase + ')'
)
```

Para:

```
EXECUTE(
        'INSERT INTO [dbo].[' + @CodBase + 'Base]
        (
        [CodBase]
        ,[CodDist]
        ,[Ano]
        ,[DatBas]
        ,[Descr]
        ,[COD_BDGD]
        ,[DataHora]
        )
        SELECT [CodBase]
        ,[CodDist]
        ,[Ano]
        ,[DatBas]
        ,[Descr]
        ,[COD_BDGD]
        ,[DataHora]
        FROM [dbo].[StoredBase]
        WHERE [CodBase] = ' + @CodBase + ''
        )
```

Por fim, deve-se alterar a *procedure* “Principal”, para que ele não execute mais outra *procedure* chamada “ExecutaCargaSIGR”, que realizava a importação dos dados da BDGD, para isso deve-se comentar a linha (colocando dois traços antes dela):

EXEC @return_value = [dbo].[ExecutaCargaSIGR] @CodBase = @CodBase, @AnoRef = @AnoRef.

A partir desse ponto, cria-se o banco “GeoPerdasRTPRegulatorio” e importa-se os quatro arquivos resultantes da descompactação do arquivo “Dados_Fisico_RTP_v5_PT-TR_ENELSP_v15012019 _cargaBT_p2.zip”, que foi obtido através da Lei de Acesso à Informação e se refere aos dados usados para o cálculo de perdas da Enel São Paulo na 5º Revisão Tarifária Periódica. Esse arquivo também está disponibilizado no link em (PULZ, 2022) - arquivo “12. LAI-48700-

003388-2019-58\Dados_Fisico_RTP_v5_PT-TR_ENELSP_v15012019_cargaBT_p2.zip”

Depois de os dados serem importados para o banco “GeoPerdasRTPRegulatorio”, deve-se executar as *procedures* “Principal” e “Recarrega”, que preparam o banco para a utilização do ProgGeoPerdas, com o seguinte código:

```
EXEC [dbo].[Principal] 390, ‘2017’, 1;
EXEC [dbo].[Recarrega] 390;
```

A execução do código acima, em um sistema i7 com 32GB de RAM e SSD, levou aproximadamente 7 horas.

A partir desse ponto, deve-se criar um banco *backup* com todas as tabelas do banco “GeoPerdasRTPRegulatorio” com o nome “GeoPerdasRTPBackup”, que é usado para a criação do banco de dados que é utilizado para a nova metodologia proposta neste trabalho.

Os scripts já modificados, conforme detalhado nesse apêndice, podem ser encontrados em (PULZ, 2022), na pasta “7. GeoPerdas ProgGeoperdas e Curvas\2. GeoPerdas_SIGR_20200811\2. Original - Ajustado - RTP”.

Apêndice I - Passo-a-passo para obtenção do banco para cálculo de perdas através da metodologia proposta

Primeiramente, os passos do 0 devem ser seguidos até a criação do banco “GeoPerdasRTPBackup”. A partir desse banco é criado um banco de dados, chamado “GeoPerdasRTPNovaMetodologia”.

É necessário realizar todas as modificações indicadas no 0 e 0 para o script “script_GeoPerdas.sql” e mais outras modificações indicadas a seguir.

Deve-se alterar o início do script de:

USE [GeoPerdasRTPRegulatorio]

Para:
USE [GeoPerdasRTPNovaMetodologia]

Deve-se alterar a *procedure* “iadTabelaOperacao” em:

```
EXECUTE(
        'CREATE TABLE [dbo].[' + @CodBase + 'AuxCargaBTDem](
        [CodBase] [int] NOT NULL,
        [CodTrafo] [nvarchar](100) NULL,
        [CodRmlBT] [nvarchar](100) NULL,
        [CodConsBT] [nvarchar](100) NOT NULL,
        [CodFas] [nvarchar](4) NULL,
        [CodPonAcopl] [nvarchar](100) NULL,
        [SemRedAssoc] [int] NULL,
        [TipMedi] [int] NULL,
        [TipCrvaCarga] [nvarchar](100) NULL,
        [EnerMedid01_MWh] [decimal](18, 9) NULL,
        [EnerMedid02_MWh] [decimal](18, 9) NULL,
        [EnerMedid03_MWh] [decimal](18, 9) NULL,
        [EnerMedid04_MWh] [decimal](18, 9) NULL,
        [EnerMedid05_MWh] [decimal](18, 9) NULL,
        [EnerMedid06_MWh] [decimal](18, 9) NULL,
        [EnerMedid07_MWh] [decimal](18, 9) NULL,
        [EnerMedid08_MWh] [decimal](18, 9) NULL,
        [EnerMedid09_MWh] [decimal](18, 9) NULL,
        [EnerMedid10_MWh] [decimal](18, 9) NULL,
        [EnerMedid11_MWh] [decimal](18, 9) NULL,
        [EnerMedid12_MWh] [decimal](18, 9) NULL,
        [DemMaxDU01_kW] [decimal](18, 9) NULL,
        [DemMaxDU02_kW] [decimal](18, 9) NULL,
        [DemMaxDU03_kW] [decimal](18, 9) NULL,
        [DemMaxDU04_kW] [decimal](18, 9) NULL,
        [DemMaxDU05_kW] [decimal](18, 9) NULL,
        [DemMaxDU06_kW] [decimal](18, 9) NULL,
```

```
[DemMaxDU07_kW] [decimal](18, 9) NULL,
[DemMaxDU08_kW] [decimal](18, 9) NULL,
[DemMaxDU09_kW] [decimal](18, 9) NULL,
[DemMaxDU10_kW] [decimal](18, 9) NULL,
[DemMaxDU11_kW] [decimal](18, 9) NULL,
[DemMaxDU12_kW] [decimal](18, 9) NULL,
[DemMaxSA01_kW] [decimal](18, 9) NULL,
[DemMaxSA02_kW] [decimal](18, 9) NULL,
[DemMaxSA03_kW] [decimal](18, 9) NULL,
[DemMaxSA04_kW] [decimal](18, 9) NULL,
[DemMaxSA05_kW] [decimal](18, 9) NULL,
[DemMaxSA06_kW] [decimal](18, 9) NULL,
[DemMaxSA07_kW] [decimal](18, 9) NULL,
[DemMaxSA08_kW] [decimal](18, 9) NULL,
[DemMaxSA09_kW] [decimal](18, 9) NULL,
[DemMaxSA10_kW] [decimal](18, 9) NULL,
[DemMaxSA11_kW] [decimal](18, 9) NULL,
[DemMaxSA12_kW] [decimal](18, 9) NULL,
[DemMaxDO01_kW] [decimal](18, 9) NULL,
[DemMaxDO02_kW] [decimal](18, 9) NULL,
[DemMaxDO03_kW] [decimal](18, 9) NULL,
[DemMaxDO04_kW] [decimal](18, 9) NULL,
[DemMaxDO05_kW] [decimal](18, 9) NULL,
[DemMaxDO06_kW] [decimal](18, 9) NULL,
[DemMaxDO07_kW] [decimal](18, 9) NULL,
[DemMaxDO08_kW] [decimal](18, 9) NULL,
[DemMaxDO09_kW] [decimal](18, 9) NULL,
[DemMaxDO10_kW] [decimal](18, 9) NULL,
[DemMaxDO11_kW] [decimal](18, 9) NULL,
[DemMaxDO12_kW] [decimal](18, 9) NULL,
[Descr] [text] NULL,
[CodAlim] [nvarchar](100) NULL,
[CodAlimAtrib] [nvarchar](100) NULL,
[CodSubAtrib] [nvarchar](100) NULL,
[CodTrafoAtrib] [nvarchar](100) NULL,
[TnsLnh_kV] [decimal](18, 9) NULL,
[TnsFas_kV] [decimal](18, 9) NULL,
CONSTRAINT [PK_AuxCargaBTDem' + @CodBase + ']
PRIMARY KEY CLUSTERED
(
[CodBase] ASC,
[CodConsBT] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]'
)
```

Para:

```
EXECUTE(
```

```
'CREATE TABLE [dbo].[' + @CodBase + 'AuxCargaBTDem](
[CodBase] [int] NOT NULL,
[CodTrafo] [nvarchar](100) NULL,
[CodRmlBT] [nvarchar](100) NULL,
[CodConsBT] [nvarchar](100) NOT NULL,
[CodFas] [nvarchar](4) NULL,
[CodPonAcopl] [nvarchar](100) NULL,
[SemRedAssoc] [int] NULL,
[TipMedi] [int] NULL,
[TipCrvaCarga] [nvarchar](100) NULL,
[EnerMedid01_MWh] [decimal](18, 9) NULL,
[EnerMedid02_MWh] [decimal](18, 9) NULL,
[EnerMedid03_MWh] [decimal](18, 9) NULL,
[EnerMedid04_MWh] [decimal](18, 9) NULL,
[EnerMedid05_MWh] [decimal](18, 9) NULL,
[EnerMedid06_MWh] [decimal](18, 9) NULL,
[EnerMedid07_MWh] [decimal](18, 9) NULL,
[EnerMedid08_MWh] [decimal](18, 9) NULL,
[EnerMedid09_MWh] [decimal](18, 9) NULL,
[EnerMedid10_MWh] [decimal](18, 9) NULL,
[EnerMedid11_MWh] [decimal](18, 9) NULL,
[EnerMedid12_MWh] [decimal](18, 9) NULL,
[DemMaxDU01_kW] [decimal](18, 9) NULL,
[DemMaxDU02_kW] [decimal](18, 9) NULL,
[DemMaxDU03_kW] [decimal](18, 9) NULL,
[DemMaxDU04_kW] [decimal](18, 9) NULL,
[DemMaxDU05_kW] [decimal](18, 9) NULL,
[DemMaxDU06_kW] [decimal](18, 9) NULL,
[DemMaxDU07_kW] [decimal](18, 9) NULL,
[DemMaxDU08_kW] [decimal](18, 9) NULL,
[DemMaxDU09_kW] [decimal](18, 9) NULL,
[DemMaxDU10_kW] [decimal](18, 9) NULL,
[DemMaxDU11_kW] [decimal](18, 9) NULL,
[DemMaxDU12_kW] [decimal](18, 9) NULL,
[DemMaxSA01_kW] [decimal](18, 9) NULL,
[DemMaxSA02_kW] [decimal](18, 9) NULL,
[DemMaxSA03_kW] [decimal](18, 9) NULL,
[DemMaxSA04_kW] [decimal](18, 9) NULL,
[DemMaxSA05_kW] [decimal](18, 9) NULL,
[DemMaxSA06_kW] [decimal](18, 9) NULL,
[DemMaxSA07_kW] [decimal](18, 9) NULL,
[DemMaxSA08_kW] [decimal](18, 9) NULL,
[DemMaxSA09_kW] [decimal](18, 9) NULL,
[DemMaxSA10_kW] [decimal](18, 9) NULL,
[DemMaxSA11_kW] [decimal](18, 9) NULL,
[DemMaxSA12_kW] [decimal](18, 9) NULL,
[DemMaxDO01_kW] [decimal](18, 9) NULL,
[DemMaxDO02_kW] [decimal](18, 9) NULL,
[DemMaxDO03_kW] [decimal](18, 9) NULL,
[DemMaxDO04_kW] [decimal](18, 9) NULL,
```

```
[DemMaxDO05_kW] [decimal](18, 9) NULL,
[DemMaxDO06_kW] [decimal](18, 9) NULL,
[DemMaxDO07_kW] [decimal](18, 9) NULL,
[DemMaxDO08_kW] [decimal](18, 9) NULL,
[DemMaxDO09_kW] [decimal](18, 9) NULL,
[DemMaxDO10_kW] [decimal](18, 9) NULL,
[DemMaxDO11_kW] [decimal](18, 9) NULL,
[DemMaxDO12_kW] [decimal](18, 9) NULL,
[Descr] [text] NULL,
[CodAlim] [nvarchar](100) NULL,
[CodAlimAtrib] [nvarchar](100) NULL,
[CodSubAtrib] [nvarchar](100) NULL,
[CodTrafoAtrib] [nvarchar](100) NULL,
[TnsLnh_kV] [decimal](18, 9) NULL,
[TnsFas_kV] [decimal](18, 9) NULL,
[fraud_idx] float,
CONSTRAINT [PK_AuxCargaBTDem' + @CodBase + ']
PRIMARY KEY CLUSTERED
(
[CodBase] ASC,
[CodConsBT] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE =
OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]'
)
```

Deve-se alterar a *procedure* “AtualizaDemandaCargaBTNT” para:

```
CREATE PROCEDURE [dbo].[AtualizaDemandaCargaBTNT]
    -- Add the parameters for the stored procedure here
    @BaseAnalisada int,
    @AlimAnalisado nvarchar(100),
    @Mes int,
    @DeltaBT_MWh decimal(18,9)
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    DECLARE @DeltaBT float;
    DECLARE @month_formatted nvarchar(2);
    DECLARE @BaseAnalisadaChar nvarchar(100);

    SELECT @month_formatted=FORMAT(@Mes, '00');
    SET @BaseAnalisadaChar = CAST(@BaseAnalisada AS nvarchar(100));

    DECLARE @command nvarchar(MAX);
    DECLARE @params nvarchar(MAX);
```

```
IF @DeltaBT_MWh > 0
BEGIN
	SET @command = '
	SELECT
		@DeltaBT=@DeltaBT_MWh*1.0/(SUM(t1.EnerMedid' + @month_formatted + '_MWh*t2.fraud_idx)
		+ SUM(t2.EnerMedid' + @month_formatted + '_MWh*t2.fraud_idx))
	FROM
		[dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTNTDem' + @AlimAnalisado + '] t1
		,[dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTDem] t2
	WHERE
		t1.[CodBase] = ' + @BaseAnalisadaChar + '
		AND t1.[CodAlimAtrib] = '' + @AlimAnalisado + ''
		AND t1.CodBase = t2.CodBase AND t1.[CodAlimAtrib] = t2.[CodAlimAtrib] AND t1.CodConsBT = t2.CodConsBT
	';
	SET @params = '@DeltaBT_MWh decimal(18,9), @DeltaBT float OUTPUT'
	EXECUTE sp_executesql @command, @params=@params, @DeltaBT_MWh=@DeltaBT_MWh, @DeltaBT=@DeltaBT OUTPUT;

	EXECUTE('
	UPDATE [t1]
		SET [t1].[EnerMedid' + @month_formatted + '_MWh] = [t1].[EnerMedid' + @month_formatted + '_MWh] +
		( [t1].[EnerMedid' + @month_formatted + '_MWh] + [t2].[EnerMedid' + @month_formatted + '_MWh] ) * t2.fraud_idx * ' + @DeltaBT + '
	FROM
		[dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTNTDem' + @AlimAnalisado + '] t1
		,[dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTDem] t2
	WHERE
		t1.[CodBase] = ' + @BaseAnalisadaChar + '
		AND t1.[CodAlimAtrib] = '' + @AlimAnalisado + ''
		AND t1.CodBase = t2.CodBase AND t1.[CodAlimAtrib] = t2.[CodAlimAtrib] AND t1.CodConsBT = t2.CodConsBT
	');
END
ELSE
BEGIN
	SET @command = '
	SELECT
		@DeltaBT=@DeltaBT_MWh*1.0/(SUM(t1.EnerMedid' + @month_formatted + '_MWh)
		+ SUM(t2.EnerMedid' + @month_formatted + '_MWh))
	FROM
```

```
                [dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTNTDem' +
@AlimAnalisado + '] t1
                ,[dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTDem] t2
        WHERE
                t1.[CodBase] = ' + @BaseAnalisadaChar + '
                AND t1.[CodAlimAtrib] = '''' + @AlimAnalisado + ''''
                AND t1.CodBase = t2.CodBase AND t1.[CodAlimAtrib] =
t2.[CodAlimAtrib] AND t1.CodConsBT = t2.CodConsBT
        ';
        SET @params = '@DeltaBT_MWh decimal(18,9), @DeltaBT float
OUTPUT'
        EXECUTE sp_executesql @command, @params=@params,
@DeltaBT_MWh=@DeltaBT_MWh, @DeltaBT=@DeltaBT OUTPUT;

        EXECUTE('
        UPDATE [t1]
                SET [t1].[EnerMedid' + @month_formatted + '_MWh] =
[t1].[EnerMedid' + @month_formatted + '_MWh] +
                ( [t1].[EnerMedid' + @month_formatted + '_MWh] +
[t2].[EnerMedid' + @month_formatted + '_MWh] ) * ' + @DeltaBT + '
        FROM
                [dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTNTDem' +
@AlimAnalisado + '] t1
                ,[dbo].[' + @BaseAnalisadaChar + 'AuxCargaBTDem] t2
        WHERE
                t1.[CodBase] = ' + @BaseAnalisadaChar + '
                AND t1.[CodAlimAtrib] = '''' + @AlimAnalisado + ''''
                AND t1.CodBase = t2.CodBase AND t1.[CodAlimAtrib] =
t2.[CodAlimAtrib] AND t1.CodConsBT = t2.CodConsBT
        ');
    END;

    EXECUTE('UPDATE [dbo].[' + @BaseAnalisadaChar +
'AuxCargaBTNTDem' + @AlimAnalisado + ']
    SET [DemMaxDU01_kW] = ([EnerMedid01_MWh] * [PropEnerMensDU01] *
1000) / ([DiasMesDU01] * 24 * [fcDU]), [DemMaxDU02_kW] =
([EnerMedid02_MWh] * [PropEnerMensDU02] * 1000) / ([DiasMesDU02] * 24 *
[fcDU]), [DemMaxDU03_kW] = ([EnerMedid03_MWh] * [PropEnerMensDU03] *
1000) / ([DiasMesDU03] * 24 * [fcDU]), [DemMaxDU04_kW] =
([EnerMedid04_MWh] * [PropEnerMensDU04] * 1000) / ([DiasMesDU04] * 24 *
[fcDU]), [DemMaxDU05_kW] = ([EnerMedid05_MWh] * [PropEnerMensDU05] *
1000) / ([DiasMesDU05] * 24 * [fcDU]), [DemMaxDU06_kW] =
([EnerMedid06_MWh] * [PropEnerMensDU06] * 1000) / ([DiasMesDU06] * 24 *
[fcDU]), [DemMaxDU07_kW] = ([EnerMedid07_MWh] * [PropEnerMensDU07] *
1000) / ([DiasMesDU07] * 24 * [fcDU]), [DemMaxDU08_kW] =
([EnerMedid08_MWh] * [PropEnerMensDU08] * 1000) / ([DiasMesDU08] * 24 *
[fcDU]), [DemMaxDU09_kW] = ([EnerMedid09_MWh] * [PropEnerMensDU09] *
1000) / ([DiasMesDU09] * 24 * [fcDU]), [DemMaxDU10_kW] =
([EnerMedid10_MWh] * [PropEnerMensDU10] * 1000) / ([DiasMesDU10] * 24 *
[fcDU]), [DemMaxDU11_kW] = ([EnerMedid11_MWh] * [PropEnerMensDU11] *
```

```
1000) / ([DiasMesDU11] * 24 * [fcDU]), [DemMaxDU12_kW] =
([EnerMedid12_MWh] * [PropEnerMensDU12] * 1000) / ([DiasMesDU12] * 24 *
[fcDU]), [DemMaxSA01_kW] = ([EnerMedid01_MWh] * [PropEnerMensSA01] *
1000) / ([DiasMesSA01] * 24 * [fcSA]), [DemMaxSA02_kW] =
([EnerMedid02_MWh] * [PropEnerMensSA02] * 1000) / ([DiasMesSA02] * 24 *
[fcSA]), [DemMaxSA03_kW] = ([EnerMedid03_MWh] * [PropEnerMensSA03] *
1000) / ([DiasMesSA03] * 24 * [fcSA]), [DemMaxSA04_kW] =
([EnerMedid04_MWh] * [PropEnerMensSA04] * 1000) / ([DiasMesSA04] * 24 *
[fcSA]), [DemMaxSA05_kW] = ([EnerMedid05_MWh] * [PropEnerMensSA05] *
1000) / ([DiasMesSA05] * 24 * [fcSA]), [DemMaxSA06_kW] =
([EnerMedid06_MWh] * [PropEnerMensSA06] * 1000) / ([DiasMesSA06] * 24 *
[fcSA]), [DemMaxSA07_kW] = ([EnerMedid07_MWh] * [PropEnerMensSA07] *
1000) / ([DiasMesSA07] * 24 * [fcSA]), [DemMaxSA08_kW] =
([EnerMedid08_MWh] * [PropEnerMensSA08] * 1000) / ([DiasMesSA08] * 24 *
[fcSA]), [DemMaxSA09_kW] = ([EnerMedid09_MWh] * [PropEnerMensSA09] *
1000) / ([DiasMesSA09] * 24 * [fcSA]), [DemMaxSA10_kW] =
([EnerMedid10_MWh] * [PropEnerMensSA10] * 1000) / ([DiasMesSA10] * 24 *
[fcSA]), [DemMaxSA11_kW] = ([EnerMedid11_MWh] * [PropEnerMensSA11] *
1000) / ([DiasMesSA11] * 24 * [fcSA]), [DemMaxSA12_kW] =
([EnerMedid12_MWh] * [PropEnerMensSA12] * 1000) / ([DiasMesSA12] * 24 *
[fcSA]), [DemMaxDO01_kW] = ([EnerMedid01_MWh] * [PropEnerMensDO01] *
1000) / ([DiasMesDO01] * 24 * [fcDO]), [DemMaxDO02_kW] =
([EnerMedid02_MWh] * [PropEnerMensDO02] * 1000) / ([DiasMesDO02] * 24 *
[fcDO]), [DemMaxDO03_kW] = ([EnerMedid03_MWh] * [PropEnerMensDO03] *
1000) / ([DiasMesDO03] * 24 * [fcDO]), [DemMaxDO04_kW] =
([EnerMedid04_MWh] * [PropEnerMensDO04] * 1000) / ([DiasMesDO04] * 24 *
[fcDO]), [DemMaxDO05_kW] = ([EnerMedid05_MWh] * [PropEnerMensDO05] *
1000) / ([DiasMesDO05] * 24 * [fcDO]), [DemMaxDO06_kW] =
([EnerMedid06_MWh] * [PropEnerMensDO06] * 1000) / ([DiasMesDO06] * 24 *
[fcDO]), [DemMaxDO07_kW] = ([EnerMedid07_MWh] * [PropEnerMensDO07] *
1000) / ([DiasMesDO07] * 24 * [fcDO]), [DemMaxDO08_kW] =
([EnerMedid08_MWh] * [PropEnerMensDO08] * 1000) / ([DiasMesDO08] * 24 *
[fcDO]), [DemMaxDO09_kW] = ([EnerMedid09_MWh] * [PropEnerMensDO09] *
1000) / ([DiasMesDO09] * 24 * [fcDO]), [DemMaxDO10_kW] =
([EnerMedid10_MWh] * [PropEnerMensDO10] * 1000) / ([DiasMesDO10] * 24 *
[fcDO]), [DemMaxDO11_kW] = ([EnerMedid11_MWh] * [PropEnerMensDO11] *
1000) / ([DiasMesDO11] * 24 * [fcDO]), [DemMaxDO12_kW] =
([EnerMedid12_MWh] * [PropEnerMensDO12] * 1000) / ([DiasMesDO12] * 24 *
[fcDO])
        WHERE [CodBase] = ' + @BaseAnalisadaChar + ' AND [CodAlimAtrib] =
''' + @AlimAnalisado + ''')

END
```

Essa última modificação faz com que a atualização das cargas virtuais que representam as PNT seja feita levando-se em consideração o Índice Potencial de Fraude da região em que cada carga está.

Depois de executadas todas as modificações no script “script_GeoPerdas.sql” e ele ser executado no novo banco “GeoPerdasRTPNovaMetodologia”, deve-se copiar todas as tabelas do banco “GeoPerdasRTPBackup” para o novo banco de dados.

Para executar o cálculo de perdas através do ProgGeoPerdas, deve-se abrir o arquivo “Código Fonte\ProgGeoPerdas.sln” através do Visual Studio, no código fonte do ProgGeoPerdas disponibilizado pela ANEEL. Depois, deve-se alterar uma linha de código, no local em que é preparado a chamada da *procedure* “AtualizaDemandaCargaBTNT”, é o código dentro do método “bExecuteBD_Click”, na linha 799 do “frmPrincipal.cs” de:

this.sqlCommServer.Parameters.AddWithValue("@DeltaBT", dblDeltaBT)

Para:
this.sqlCommServer.Parameters.AddWithValue("@DeltaBT_MWh", dblDeltaBT_MWh).

Depois de alterado o código fonte do ProgGeoPerdas através do Visual Studio (pode ser baixado do Visual Studio Community do site da Microsoft), deve-se realizar o *build* do novo código alterado, clicando-se em “Build” e depois em “Build ProgGeoPerdas” no Visual Studio, isso cria o executável “Código Fonte\ProgGeoPerdas\bin\Release\ProgGeoPerdas.exe”, que será utilizado para o novo cálculo de perdas.

Antes de executar o ProgGeoPerdas modificado, deve-se atualizar o Índice Potencial de Fraude (IPF) das cargas. O notebook do Apêndice D gera um arquivo chamado “ipf_kriging_500.shp” que deve ser carregado no banco “GeoPerdasRTPNovaMetodologia” através do código abaixo a ser executado através do cmd do Windows (o GDAL deve ser instalado conforme descrito no capítulo 5):

“C:\Program Files\GDAL\ogr2ogr.exe” --config MSSQLSPATIAL_USE_BCP TRUE --config MSSQLSPATIAL_BCP_SIZE 60000 -f MSSQLSpatial "MSSQL:server=localhost;database=GeoPerdasRTPNovaMetodologia;trusted_connection=yes;DRIVER={SQL Server Native Client 11.0};" "ipf_kriging_500.shp" -gt 60000 -progress

Após a execução do comando acima, existirá uma nova tabela no banco chamada “ipf_kriging_500”, então deve-se executar o código SQL abaixo para que os valores de IPF sejam salvos na tabela que é usada pelo ProgGeoPerdas para a alocação de cargas virtuais

```
UPDATE CargaBTDem
SET
	CargaBTDem.[fraud_idx] = (CASE WHEN IPF.value IS NULL THEN 0 ELSE IPF.value END)
FROM
	[GeoPerdasRTPNovaMetodologia].[dbo].[390AuxCargaBTDem] AS CargaBTDem
LEFT JOIN
	[GeoPerdasRTPNovaMetodologia].[dbo].[ipf_kriging_500] AS IPF
ON
	CargaBTDem.CodConsBT = IPF.cod_id
```

Após todos esses passos, o banco de dados “GeoPerdasRTPNovaMetodologia” pode ser utilizado pelo ProgGeoPerdas modificado para o cálculo de perdas através da metodologia proposta neste trabalho.

Os scripts já modificados, conforme detalhado nesse apêndice, podem ser encontrados em (PULZ, 2022), na pasta “7. GeoPerdas ProgGeoperdas e Curvas\2. GeoPerdas_SIGR_20200811\3. Nova Metodologia”.

Apêndice J - Notebook Python para obtenção da análise entre IQR e diferenças para perdas entre metodologias

A versão Python utilizada é 3.10.4. Esse *notebook* também se encontra no link em (PULZ, 2022) – arquivo “3.0-jp-analise_das_perdas.ipynb”. Os seguintes pacotes, com suas respectivas versões, foram utilizados para execução do notebook Python:

argon2-cffi==21.3.0
argon2-cffi-bindings==21.2.0
asttokens==2.0.5
attrs==21.4.0
backcall==0.2.0
beautifulsoup4==4.10.0
bleach==4.1.0
certifi==2021.10.8
cffi==1.15.0
click==8.1.2
click-plugins==1.1.1
cligj==0.7.2
colorama==0.4.4
cycler==0.11.0
debugpy==1.6.0
decorator==5.1.1
defusedxml==0.7.1
entrypoints==0.4
executing==0.8.3
fastjsonschema==2.15.3
Fiona==1.8.20
fonttools==4.31.2
GDAL==3.4.1
geopandas==0.10.2
ipykernel==6.12.1
ipython==8.2.0
ipython-genutils==0.2.0
ipywidgets==7.7.0
jedi==0.18.1
Jinja2==3.1.1
jsonschema==4.4.0
jupyter==1.0.0
jupyter-client==7.2.1
jupyter-console==6.4.3
jupyter-core==4.9.2
jupyterlab-pygments==0.1.2
jupyterlab-widgets==1.1.0
kiwisolver==1.4.2
MarkupSafe==2.1.1
matplotlib==3.5.1
matplotlib-inline==0.1.3

mistune==0.8.4
munch==2.5.0
nbclient==0.5.13
nbconvert==6.4.5
nbformat==5.3.0
nest-asyncio==1.5.5
notebook==6.4.10
numpy==1.22.3
packaging==21.3
pandas==1.4.2
pandocfilters==1.5.0
parso==0.8.3
pickleshare==0.7.5
Pillow==9.1.0
prometheus-client==0.13.1
prompt-toolkit==3.0.29
psutil==5.9.0
pure-eval==0.2.2
pycparser==2.21
Pygments==2.11.2
PyKrige==1.6.1
pyodbc==4.0.32
pyparsing==3.0.7
pyproj==3.3.0
pyrsistent==0.18.1
python-dateutil==2.8.2
pytz==2022.1
pywin32==303
pywinpty==2.0.5
pyzmq==22.3.0
qtconsole==5.3.0
QtPy==2.0.1
Rtree==0.9.7
scipy==1.8.0
seaborn==0.11.2
Send2Trash==1.8.0
Shapely==1.8.0
six==1.16.0
soupsieve==2.3.1
stack-data==0.2.0
terminado==0.13.3
testpath==0.6.0
tornado==6.1
traitlets==5.1.1
wcwidth==0.2.5
webencodings==0.5.1
widgetsnbextension==3.6.0

Segue o *notebook* Python transformado em documento Word utilizando o software Pandoc:

Análise do resultado das simulações de perdas

2021-04-04 Jonatas Pulz

```
import pickle
import os

import pandas as pd
import numpy as np

import pyodbc

import fiona
from shapely.geometry import box, Point
import geopandas
import rtree

import seaborn as sns
import matplotlib
from matplotlib import pyplot as plt
from IPython import display
```

Configurações

```
pd.set_option('display.max_rows', 200)
pd.set_option('display.max_columns', 200)

font = {'family' : 'normal',
        'weight' : 'bold',
        'size'   : 15}

matplotlib.rc('font', **font)
```

Conexão com o banco de dados

```
conn = pyodbc.connect('DRIVER={SQL Server Native Client
11.0};SERVER=localhost;DATABASE=master;Trusted_Connection=yes;')
```

Queries

```
sql_losses_by_feeder = f'\
WITH RTPRegulatorio AS \
( \
      SELECT \
            result.CodAlim \
            ,result.[EnergiaInj_kWh] \
            ,result.[PerdaEnergiaTecnica_kWh] \
            ,nt.EnerNT_MWh*1000 as PerdaEnergiaNT_kWh \
            ,(result.[PerdaEnergiaTecnica_kWh])/result.[EnergiaInj_kWh]*100 AS
PerdaEnergiaTecnica_EnergiaInj_per \
```

```
            ,(nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100 AS
PerdaEnergiaNT_EnergiaInj_per \
            ,(result.[PerdaEnergiaTecnica_kWh] + \
                  nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100 AS
PerdaEnergiaTotal_EnergiaInj_per \
      FROM \
            [GeoPerdasRTPRegulatorio].[dbo].[390AuxResultadoAno] result \
      LEFT JOIN \
            ( \
                  SELECT \
                        CodAlimAtrib \
                        ,sum(EnerNT) EnerNT_MWh \
                  FROM \
                        (SELECT \
                              CodAlimAtrib \
                              ,sum([EnerMedid01_MWh] \
                                    +[EnerMedid02_MWh] \
                                    +[EnerMedid03_MWh] \
                                    +[EnerMedid04_MWh] \
                                    +[EnerMedid05_MWh] \
                                    +[EnerMedid06_MWh] \
                                    +[EnerMedid07_MWh] \
                                    +[EnerMedid08_MWh] \
                                    +[EnerMedid09_MWh] \
                                    +[EnerMedid10_MWh] \
                                    +[EnerMedid11_MWh] \
                                    +[EnerMedid12_MWh] \
                              ) EnerNT \
                        FROM \

      [GeoPerdasRTPRegulatorio].[dbo].[390AuxCargaBTNTDem] \
                        WHERE \
                              CodAlimAtrib IS NOT NULL \
                        GROUP BY \
                              CodAlimAtrib \
                        UNION ALL \
                        SELECT \
                              [CodAlimAtrib] \
                              ,sum([EnerMedid01_MWh] \
                                    +[EnerMedid02_MWh] \
                                    +[EnerMedid03_MWh] \
                                    +[EnerMedid04_MWh] \
                                    +[EnerMedid05_MWh] \
                                    +[EnerMedid06_MWh] \
                                    +[EnerMedid07_MWh] \
                                    +[EnerMedid08_MWh] \
                                    +[EnerMedid09_MWh] \
                                    +[EnerMedid10_MWh] \
                                    +[EnerMedid11_MWh] \
                                    +[EnerMedid12_MWh] \
                              ) EnerNT \
                        FROM \

      [GeoPerdasRTPRegulatorio].[dbo].[390AuxCargaMTNTDem] \
                        WHERE \
                              CodAlimAtrib IS NOT NULL \
                        GROUP BY \
                              CodAlimAtrib \
                        ) nt \
```

```
                    GROUP BY \
                          CodAlimAtrib \
              ) nt \
        ON result.CodAlim = nt.CodAlimAtrib \
        WHERE \
              nt.EnerNT_MWh >= 0 \
) \
,RTPNovaMetodologia AS \
( \
        SELECT \
              result.CodAlim \
              ,result.[EnergiaInj_kWh] \
              ,result.[PerdaEnergiaTecnica_kWh] \
              ,nt.EnerNT_MWh*1000 as PerdaEnergiaNT_kWh \
              ,(result.[PerdaEnergiaTecnica_kWh])/result.[EnergiaInj_kWh]*100 AS
PerdaEnergiaTecnica_EnergiaInj_per \
              ,(nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100 AS
PerdaEnergiaNT_EnergiaInj_per \
              ,(result.[PerdaEnergiaTecnica_kWh] + \
                    nt.EnerNT_MWh*1000)/result.[EnergiaInj_kWh]*100 AS
PerdaEnergiaTotal_EnergiaInj_per \
        FROM \
              [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxResultadoAno] result \
        LEFT JOIN \
              ( \
                    SELECT \
                          CodAlimAtrib \
                          ,sum(EnerNT) EnerNT_MWh \
                    FROM \
                          (SELECT \
                                CodAlimAtrib \
                                ,sum([EnerMedid01_MWh] \
                                      +[EnerMedid02_MWh] \
                                      +[EnerMedid03_MWh] \
                                      +[EnerMedid04_MWh] \
                                      +[EnerMedid05_MWh] \
                                      +[EnerMedid06_MWh] \
                                      +[EnerMedid07_MWh] \
                                      +[EnerMedid08_MWh] \
                                      +[EnerMedid09_MWh] \
                                      +[EnerMedid10_MWh] \
                                      +[EnerMedid11_MWh] \
                                      +[EnerMedid12_MWh] \
                                ) EnerNT \
                          FROM \

        [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxCargaBTNTDem] \
                          WHERE \
                                CodAlimAtrib IS NOT NULL \
                          GROUP BY \
                                CodAlimAtrib \
                          UNION ALL \
                          SELECT \
                                [CodAlimAtrib] \
                                ,sum([EnerMedid01_MWh] \
                                      +[EnerMedid02_MWh] \
                                      +[EnerMedid03_MWh] \
                                      +[EnerMedid04_MWh] \
                                      +[EnerMedid05_MWh] \
```

```
                                        +[EnerMedid06_MWh] \
                                        +[EnerMedid07_MWh] \
                                        +[EnerMedid08_MWh] \
                                        +[EnerMedid09_MWh] \
                                        +[EnerMedid10_MWh] \
                                        +[EnerMedid11_MWh] \
                                        +[EnerMedid12_MWh] \
                                  ) EnerNT \
                            FROM \

      [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxCargaMTNTDem] \
                            WHERE \
                                  CodAlimAtrib IS NOT NULL \
                            GROUP BY \
                                  CodAlimAtrib \
                            ) nt \
                      GROUP BY \
                            CodAlimAtrib \
            ) nt \
      ON result.CodAlim = nt.CodAlimAtrib \
) \
SELECT \
      RTPRegulatorio.* \
      ,RTPNovaMetodologia.EnergiaInj_kWh AS EnergiaInj_kWh_N \
      ,RTPNovaMetodologia.PerdaEnergiaTecnica_kWh AS PerdaEnergiaTecnica_kWh_N \
      ,RTPNovaMetodologia.PerdaEnergiaNT_kWh AS PerdaEnergiaNT_kWh_N \
      ,RTPNovaMetodologia.PerdaEnergiaTecnica_EnergiaInj_per  AS
PerdaEnergiaTecnica_EnergiaInj_per_N \
      ,RTPNovaMetodologia.PerdaEnergiaNT_EnergiaInj_per AS
PerdaEnergiaNT_EnergiaInj_per_N \
      ,RTPNovaMetodologia.PerdaEnergiaTotal_EnergiaInj_per AS
PerdaEnergiaTotal_EnergiaInj_per_N \
FROM \
      RTPRegulatorio \
INNER JOIN \
      RTPNovaMetodologia \
ON \
      RTPNovaMetodologia.CodAlim = RTPRegulatorio.CodAlim \
ORDER BY \
      RTPRegulatorio.CodAlim \
'

sql_lv_loads_rtp = f'\
SELECT \
    CodConsBT \
    ,CodAlimAtrib \
    ,fraud_idx \
FROM \
    [GeoPerdasRTPNovaMetodologia].[dbo].[390AuxCargaBTDem] \
'
```

Carregamento dos dados

Perdas em ambas as metodologias por alimentador

```
df_losses_by_feeder_raw = pd.read_sql(sql_losses_by_feeder, conn)
df_losses_by_feeder = df_losses_by_feeder_raw.copy()
df_losses_by_feeder.head()
```

Consumidores da base de perdas da Revisão Tarifária Periódica (RTP)

Esses dados já contém o IPF para cada consumidor

```
df_lv_loads_rtp_raw = pd.read_sql(sql_lv_loads_rtp, conn)
df_lv_loads_rtp = df_lv_loads_rtp_raw.copy()
df_lv_loads_rtp.head()
```

Prepara dados

```
feeders_simulated = df_losses_by_feeder['CodAlim'].unique()
feeders_simulated.shape[0]
```

```
feeders_simulated = df_losses_by_feeder['CodAlim'].unique()
df_lv_loads_rtp = df_lv_loads_rtp.loc[
    df_lv_loads_rtp['CodAlimAtrib'].isin(feeders_simulated)
].copy()
df_lv_loads_rtp.shape[0]
```

```
for c in df_losses_by_feeder.columns:
    if c[-4:] == '_per':
        print(c)
        df_losses_by_feeder[f'{c}_abs_diff'] = np.abs((df_losses_by_feeder[c]-
df_losses_by_feeder[f'{c}_N'])/df_losses_by_feeder[c])
df_losses_by_feeder['total_losses_diff'] = df_losses_by_feeder.apply(
    #lambda row: np.max([row[c] for c in df_losses_by_feeder.columns if c[-9:] ==
'_abs_diff'])
    lambda row: row['PerdaEnergiaTotal_EnergiaInj_per_abs_diff']
    ,axis=1
)
df_losses_by_feeder.sort_values(by='total_losses_diff', ascending=False,
inplace=True)
df_losses_by_feeder
```

Analisa dados

Plots

```
sub_set = df_losses_by_feeder['CodAlim'].values[:5]
fig, ax = plt.subplots(figsize=(10, 10))
sns.boxplot(
    data=df_lv_loads_rtp.loc[df_lv_loads_rtp['CodAlimAtrib'].isin(sub_set)]
    ,x='fraud_idx'
    ,y='CodAlimAtrib'
    ,order=sub_set
    ,ax=ax
)
```

Correlação entre IQR e diferenças entre metodologias

```
df_lv_loads_rtp_iqr = df_lv_loads_rtp.groupby(
    'CodAlimAtrib'
).agg(
    q1=('fraud_idx', lambda x: np.quantile(x, 0.25))
    ,q3=('fraud_idx', lambda x: np.quantile(x, 0.75))
)
df_lv_loads_rtp_iqr['iqr'] = df_lv_loads_rtp_iqr['q3'] - df_lv_loads_rtp_iqr['q1']
df_lv_loads_rtp_iqr

df_lv_loads_rtp_iqr = df_lv_loads_rtp_iqr.merge(
    df_losses_by_feeder[['CodAlim', 'total_losses_diff']]
    ,how='inner'
    ,left_index=True
    ,right_on='CodAlim'
)

x = df_lv_loads_rtp_iqr['iqr']
y = df_lv_loads_rtp_iqr['total_losses_diff']
a, b = np.polyfit(x, y, 1)
display.display(a)

fig, ax = plt.subplots(figsize=(10, 7))
plt.scatter(
    x
    ,y
    ,label='Pontos obtidos'
)
plt.plot(x, x*a + b, c='r', label='Reta ajustada')
ax.set_xlabel('IQR')
ax.set_ylabel('Variação relativa absoluta nas perdas totais')
ax.legend()
```