DÉBORA RIBEIRO DOIMO DE CAMPOS

ANÁLISE QUANTITATIVA DO IMPACTO DA AUSÊNCIA DE DADOS

São Paulo
2021

DÉBORA RIBEIRO DOIMO DE CAMPOS

ANÁLISE QUANTITATIVA DO IMPACTO DA AUSÊNCIA DE DADOS

Versão Revisada

Dissertação apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção do título
de Mestre em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Jorge Rady de Almeida Júnior

São Paulo

2021

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

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, 19 de julho de 2021.

Assinatura do autor:

Assinatura do orientador:

Catálogo-na-publicação

Campos, Débora R. D.

Análise quantitativa do impacto da ausência de dados / D. R. D.

Campos -- versão corr. -- São Paulo, 2021.

74 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.dados ausentes 2.regressão logística 3.qualidade de dados
I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Dedico este trabalho a Manuela Ribeiro Doimo Farrielo de Campos.

(In Memoriam)

AGRADECIMENTOS

Foram alguns anos de grande provação, de muito aprendizado, de dias de desânimo e de dias de muita fé. Aprendi muito, não só sobre ciência e pesquisa, mas também sobre mim, minhas crenças limitantes e sobretudo sobre perseverança. Tudo isso só foi possível, pois, tive apoio de muita gente e sou muito grata por isso.

Gostaria de agradecer aos meus amigos pela parceria: Latino, Júnior, Mayra, Gustavo, Monique e Carol. Aos meus amigos do GAS (Grupo de Análise de Segurança), que fizeram a minha passagem pela USP muito agradável. Em especial, à minha amiga Lívia, por me ajudar nas revisões.

À minha família, que me dá todo o suporte sempre: minha mãe Francisca, meu pai Pedro, meu irmão Francisco, minha prima-irmã Fernanda e as minhas tias muito queridas (tia Ilsa, tia Maria, tia Jê e tia Leila).

Ao meu marido, Bruno, pela confiança, motivação, apoio e força. Sem esse apoio nada disso seria possível.

À minha bebê Manuzinha que partiu muito precocemente, mas que me deu muita força para poder finalizar esse estudo.

Por fim, mas tão importante quanto, o meu orientador professor Rady pela infinita paciência comigo e com minhas indecisões.

Sem o apoio de todas essas pessoas nada disso seria possível, então, novamente, obrigada a todos! Vocês são incríveis!

Acredito em Deus, todos os outros devem apresentar dados e fatos.

Edward Deming

RESUMO

Hoje o Big Data já faz parte do cotidiano das pessoas e está em itens como: assistentes virtuais, Siri e Alexa; carros autônomos; recomendações de compras; sistemas antifraude; aplicações na área da saúde, como monitoramento cardíaco e de passos; etc. Essas novas tecnologias facilitam muito o dia a dia da população, geram negócios e consequentemente produzem um imenso volume de dados a serem armazenados, analisados e tratados. Com o grande volume de dados, surgem também problemas como *outliers*, atributos vazios e replicados, que caracterizam baixa qualidade da informação. Um dos tipos mais comuns de baixa qualidade da informação é ausência de dados. Essa falta de dados acarreta vários problemas, como a redução do poder estatístico, distorção da estimativa de parâmetros e redução na representatividade das amostras, que impactam a análise do estudo. A maioria das ausências acontece mais comumente em pesquisas respondidas manualmente, transcrições ou anotações, em falha de sensores e em bases de dados de sistemas que possuem algum equívoco de lógica. Por isso, são muito frequentes no dia a dia de profissionais de tecnologia, pesquisadores e cientistas, especialmente para aqueles que trabalham e processam grandes volumes de dados. Boa parte dos trabalhos recentes relacionados ao tema ausência de dados, focam em como tratar ou resolver a falta de dados e não necessariamente, é voltado a compreender o real impacto dessa omissão. Por consequência, o objetivo central desta dissertação é responder a seguinte pergunta: “Como uma análise de dados, feita utilizando um algoritmo preditivo, é impactada pelos dados ausentes?”. Então, foi codificada uma versão do algoritmo *Logistic Regression*, para testar essas situações. Na qual foram implementados alguns cenários, que consistiu em aumentar progressivamente a omissão em dados de treino e de teste. Após codificar toda essa lógica, foram usados os dados da Flor de Íris, que é amplamente reconhecida no meio acadêmico-científico, para testar a solução e servir como demonstrativo da prova de conceito. Depois de executar o algoritmo, foram obtidos os resultados da Acurácia e da Matriz de Confusão da regressão em cada cenário. A partir da análise dos resultados, conclui-se que o aumento das omissões diminui a assertividade do algoritmo *Logistic Regression* utilizando-se o *dataset* Flor de Íris, quando comparada a Acurácia do cenário completo com o cenário de ausências em 60%. Contudo, não é em todas as situações que o acréscimo de incompletudes piorou o resultado. Logo, é difícil de afirmar que em todos os cenários, quanto maior o número de ausências pior é a qualidade da análise.

Palavras-chave: dados ausentes; regressão logística; qualidade de dados.

ABSTRACT

Today Big Data is part of people's daily lives and are present in many items such as: virtual assistants, such as Siri and Alexa; autonomous cars; shopping recommendations; anti-fraud systems; applications in the health area, such as cardiac and step monitoring; etc. These new technologies make the people's daily life much easier, generate business and consequently produce a huge volume of data to be stored, analyzed and treated. Due to the large volume of data, problems also arise such as outliers, empty and replicated attributes, which characterize low quality of information. One of the most common types of low-quality information is missing data. This omission of data causes several problems like the reduction of statistical power, distortion of the estimate of parameters and reduction in the representativeness of the samples, which impact the analysis of the study. Most absences occur most commonly in surveys answered manually, transcriptions or annotations, in failure of sensors and in databases of systems that present some logic mistake. For this reason, they are very common in the daily lives of technology professionals, researchers and scientists, especially for those who work and process large volumes of data. Much of the recent work related to the topic of missing data focuses on how to treat or solve the missing and is not necessarily aimed at understanding the real impact of this omission. Consequently, the main objective of this dissertation is to answer the following question: "How is a data analysis, made by using a predictive algorithm, impacted by the missing data?". Then, a version of the Logistic Regression algorithm was coded to test these situations. In which some scenarios were implemented, which consisted of progressively increasing the omission in training and test data. After coding all this logic, was used Iris Flower dataset, which is widely recognized in the academic-scientific environment, to test the solution and serve as a proof of concept. After executing the algorithm, the results of the Accuracy and Confusion Matrix of the regression were obtained in each scenario. From the analysis of the results, it is concluded that the increase in omissions decreases the assertiveness of the Logistic Regression algorithm using the Iris Flower dataset, when comparing the Accuracy of the complete scenario to the absence scenario by 60%. However, it is not in all situations that the addition of incompleteness worsened the result. Therefore, it is difficult to say that in all scenarios, the greater the number of absences, the worse the quality of the analysis.

Keywords: missing data; missing values; logistic regression; data quality.

LISTA DE FIGURAS

Figura 1 - Tipos de Dados.....	17
Figura 2 - Tipos de ausências.....	21
Figura 3 - Métodos para tratamento de dados ausentes	24
Figura 4 - Flor de Íris.....	35
Figura 5 - Anotações de Flor de Íris.....	64

LISTA DE TABELAS

Tabela 1 - Funções para tratamento de omissões em Pandas/Python.....	30
Tabela 2 - Sequência de passos para o desenvolvimento do projeto	32
Tabela 3 - Resumo dos cenários	34
Tabela 4 - Comparativo entre algoritmos preditivos.....	37
Tabela 5 - Matriz de confusão da flor de Íris	39
Tabela 6 – Matriz de confusão nas três medições para o Cenário 1	51
Tabela 7 – Matriz de confusão nas três medições para o Cenário 2	51
Tabela 8 – Matriz de confusão nas três medições para o Cenário 3	52
Tabela 9 – Matriz de confusão nas três medições para o Cenário 4	53
Tabela 10 – Acurácia nas três medições para cada cenário.....	54

LISTA DE ABREVIATURAS E SIGLAS

Acc	Acurácia
DA	Dados Ausentes
EM	<i>Expectation Maximization</i>
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
KNN	<i>K-Nearest Neighbors</i>
MAR	<i>Missing At Random</i>
MC	Matriz de Confusão
MCAR	<i>Missing Completely at Random</i>
MNAR	<i>Missing Not At Random</i>
NaN	<i>Not a Number</i>

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Contextualização	12
1.2	Trabalhos relacionados	14
1.3	Motivação e Objetivos	15
1.4	Organização	15
2	FUNDAMENTAÇÃO TEÓRICA.....	17
2.1	Tipos de dados.....	17
2.2	Tipos de atributos.....	18
2.3	Limpeza de dados	19
2.4	Dados ausentes	20
2.4.1	<i>Missing Completely At Random</i>	21
2.4.2	<i>Missing At Random</i>	22
2.4.3	<i>Missing Not At Random</i>	22
2.5	Considerações finais do capítulo.....	23
3	TRATAMENTOS	24
3.1	Formas de tratamento	24
3.1.1	Exclusão.....	24
3.1.2	Preenchimento	26
3.2	Comparativo: exclusão <i>versus</i> preenchimento.....	28
3.3	Ferramentas de tratamento	29
3.4	Considerações finais do capítulo.....	31
4	PROPOSTA	32
4.1	Planejamento	32

4.2	Cenários	33
4.3	Base de dados	34
4.4	Técnica preditiva	35
4.4.1	Algoritmo de regressão logística	38
4.4.2	Assertividade do algoritmo	38
4.5	Considerações finais do capítulo.....	39
5	IMPLEMENTAÇÃO	40
5.1	<i>Dataset</i>	40
5.2	Treinamento do modelo.....	43
5.3	Inserção de ausências.....	44
5.4	Implementação de cenários com dados faltantes	45
5.5	Obtenção dos resultados.....	47
5.6	Considerações finais do capítulo.....	47
6	RESULTADOS	49
6.1	Tempo	49
6.2	Medições	50
6.3	Análise dos resultados	54
6.4	Considerações finais do capítulo.....	55
7	CONCLUSÃO.....	56
7.1	Trabalhos futuros	56
7.2	Contribuições desse trabalho	57
	REFERÊNCIAS.....	58
	ANEXOS	63
	ANEXO A – CONJUNTO DE DADOS DA FLOR DE ÍRIS	64
	ANEXO B – IMPLEMENTAÇÃO	65

1 INTRODUÇÃO

Neste capítulo, são apresentadas a contextualização deste estudo, o levantamento de trabalhos relacionados, a motivação e o objetivo deste trabalho, além de sua organização.

1.1 Contextualização

Segundo estudo feito por Lori Lewis e Chadd Callahan, da *Cumulus Media* (a segunda maior operadora de estações de rádio nos EUA), em agosto de 2017 aconteceu na Internet em apenas 60 segundos cerca de 156 milhões de envio de e-mails, 4 milhões de vídeos assistidos no YouTube, 900 mil *logins* no Facebook, 3,5 milhões de buscas no Google, 70 mil horas de vídeos assistidos na Netflix, 40 mil horas de música ouvidas no Spotify, 750 mil dólares gastos em compras on-line e 342 mil aplicativos baixados para smartphones Apple e Android (PERELMUTER, 2018).

Hoje vive-se na era dos dados e da Inteligência Artificial (IA), mas para conseguir algoritmos de IA eficientes são necessárias três coisas: grande volume de dados (*Big Data*), poder computacional e bons profissionais que saibam trabalhar com dados (LEE, 2019). O *Big Data* e a IA estão ganhando cada vez mais espaço no cotidiano das pessoas e é gerado e consumido em coisas como: assistentes virtuais, como Siri e Alexa; carros autônomos; recomendações de compras; sistemas antifraude; aplicações na área da saúde, como monitoramento cardíaco e de passos; etc. Segundo a PricewaterhouseCoopers (Pwc), estima-se que a implementação e a utilização da IA irá adicionar ao PIB global 15,7 trilhões de dólares até 2030 (LEE, 2019).

Essas novas tecnologias facilitam muito o dia a dia da população, geram negócios e conseqüentemente produzem um imenso volume de dados a serem armazenados, analisados e tratados. Contudo, a análise destes grandes volumes de dados exige muito conhecimento técnico, uso ou a criação de sistemas especializados, além da figura do cientista de dados, um especialista em programação computacional, estatística e matemática. Quanto maior o número de sistemas e pessoas envolvidas, maior a chance de erros.

Com o grande volume de dados, surgem também problemas nos dados como *outliers*, atributos vazios e replicados que caracterizam baixa qualidade da

informação. A qualidade da informação impacta diretamente na eficiência e na efetividade de empresas e organizações. Segundo o relatório do *Data Warehousing Institute*, estima-se que a baixa qualidade de dados, custou em 2016, mais de 600 bilhões de dólares às empresas estado-unidenses (BATINI; SCANNAPIECO, 2016).

Um dos tipos mais comuns de baixa qualidade da informação é ausência de dados. A omissão dos dados acarreta vários problemas, como a redução do poder estatístico. Além disso, a ausência pode causar distorção da estimativa de parâmetros, reduzir a representatividade das amostras e impactar a análise do estudo. Essas distorções são preocupantes, pois, podem ameaçar a validade dos ensaios e levar a conclusões erradas (KANG, 2013). Além disso, o tratamento inadequado das incompletudes faz com que se perca muita informação relevante que está contida nesses registros.

Dados ausentes são também conhecidos por incompletos, inexistentes, faltantes, omissos ou perdidos, enquanto em inglês, são chamados de *Missing values* ou *Missing data*. As ausências são caracterizadas pela inexistência da informação em uma determinada posição ou, pensando em bancos de dados, campos que estão preenchidos com caracteres que significam a falta do dado. Quando se trata de identificar omissões em computação, normalmente usa-se *Not a Number* (NaN), esse símbolo é usado em linguagens de programação e em bancos de dados para representar um valor indefinido ou irrepresentável.

A inexistência de dados pode acontecer pelos mais variados motivos: i) os participantes de uma pesquisa podem não responder a algumas perguntas; ii) em uma pesquisa de longo prazo, os participantes podem se retirar do estudo antes que ele seja finalizado; iii) em uma pesquisa preenchida manualmente, podem ocorrer erros no momento da entrada das informações; iv) os equipamentos e mecanismos de coleta ou de gravação de dados podem funcionar mal; etc (OSBORNE, 2013).

O vazio, a omissão, a ausência, o nada, são conceitos explorados desde antes de Cristo. Os antigos gregos viam potencial nesse assunto, exemplo disso é a deusa grega *Khaos*, que representava o espaço vazio. Que, na mitologia da criação, era o espaço não preenchido entre a terra e o céu. Outros interessados pelo assunto foram os matemáticos olmecas, indianos e árabes que identificaram a utilidade da quantificação do nada, que hoje chamamos de zero (OSBORNE, 2013).

O vazio ou nada é um assunto tratado em diferentes áreas do conhecimento, como Filosofia, Física, Cosmologia, Matemática e até mesmo na Gramática. Contudo,

na Computação este tema só veio a ser discutido recentemente por Roderick Joseph A. Little, professor de bioestatística, que a partir da década de 1970 começou a publicar artigos nessa área de pesquisa (LITTLE, 1988).

1.2 Trabalhos relacionados

Atualmente a maioria dos trabalhos relacionados ao tema ausência de dados, podem ser enquadrados como técnicas e metodologias para preenchimento dos dados incompletos ou conceitos básicos sobre DA. Os trabalhos relacionados a imputação, utilizam-se de técnicas matemáticas, estatísticas e de IA com a finalidade de encontrar um padrão para as incompletudes e assim inferir valores para as mesmas. Destacam-se os estudos:

- Imputação múltipla e análise de casos completos em modelos de regressão logística: uma avaliação prática do impacto das perdas em covariáveis (CAMARGOS et al., 2011). Neste trabalho os autores utilizam técnicas de Imputação Múltipla e de Análise de Casos Completos, para comparar os resultados e chegar a conclusão de qual técnica é mais eficiente para o estudo em questão.
- *Missing data* (LITTLE et al., 2016). Este artigo explica técnicas de tratamento de ausências de dados, como Imputação Múltipla e Estimativa de Máxima Verossimilhança e seus impactos.
- *Missing data treatments in intervention studies: What was, what is, and what should be* (RIOUX; LITTLE, 2019). Este artigo discute as características das ausências de dados e também formas de tratamentos possíveis.
- *Random forest missing data algorithms* (TANG; ISHWARAN, 2017). Este estudo é voltado a utilização de tipos diferentes de florestas randômicas para tratamento de ausências.
- *Graphical Models for Inference with Missing Data* (MOHAN, 2017). Nesta tese a autora faz uso de técnicas de visualização gráfica para fazer o preenchimento dos dados faltantes e então compara seus resultados com os resultados de outras técnicas mais tradicionais de tratamento.

1.3 Motivação e Objetivos

Dados ausentes são muito frequentes no dia a dia de profissionais de tecnologia, pesquisadores e cientistas, especialmente para aqueles que trabalham e processam grandes volumes de dados. A maioria das ausências acontece mais comumente em pesquisas respondidas manualmente, transcrições ou anotações, em falha de sensores e em bases de dados de sistemas que possuem algum equívoco de lógica. A falta de conhecimento e tratamento de omissões adequado dificulta a interpretação, a análise e os cálculos da maioria dos estudos.

Muitos pesquisadores e profissionais que trabalham com bases de conhecimento tratam dados ausentes como perdidos, inúteis e, por isso, preenchem essas ausências com informações sem padrão, significado ou excluem o dado. Entretanto, essa falta de dados é, na maioria das vezes, tratável e não deveria ser desconsiderada da amostra. Por isso, da mesma maneira que os antigos viram utilidade e informação na ausência, este trabalho propõe-se a demonstrar que a omissão pode ser informativa e que os dados incompletos não precisam ser excluídos das análises, mas sim, fazer parte de uma compreensão complementar e mais precisa da população estudada.

Como visto anteriormente, boa parte dos trabalhos recentes relacionados a esse tema, focam em como tratar ou resolver a falta de dados e não necessariamente, é voltado a compreender o real impacto dessa omissão. Uma vez que foi difícil encontrar materiais nessa linha de pesquisa, decidiu-se direcionar este estudo para esse aspecto.

Dito isto, o objetivo central desta dissertação é responder a seguinte pergunta: “Como uma análise de dados, feita utilizando um algoritmo preditivo, é impactada pelos dados ausentes?”. A ideia é comparar a utilização de um mesmo algoritmo em uma mesma base de dados, contudo, ir aumentando gradativamente a quantidade de ausências inseridas. Assim, pode-se comparar a assertividade desse algoritmo em diferentes momentos.

1.4 Organização

Para responder à pergunta feita anteriormente, no Capítulo 2 são apresentados os principais conceitos referentes a dados ausentes. Na sequência, no Capítulo 3,

são abordados os tratamentos mais utilizados, o comparativo entre eles em diferentes situações, algumas funções de tratamentos previamente programadas e a técnica preditiva utilizada nesta pesquisa. Tanto no segundo quanto no terceiro capítulo a ideia é embasar o leitor para a discussão da pesquisa.

O Capítulo 4, contém o detalhamento de como aconteceu o desenvolvimento e organização da proposta, os cenários trabalhados, a escolha da base de dados, do algoritmo, etc. Já o Capítulo 5, trata da discussão de toda implementação feita. No capítulo 6, são mostrados os resultados obtidos e a análise técnica dos mesmos. Por fim são feitas as conclusões do projeto e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são providos os subsídios necessários para a compreensão dos aspectos básicos relevantes ao tema dados ausentes: i) os tipos básicos de dados encontrados em populações de dados; ii) os tipos de atributos; iii) o que é limpeza de dados; iv) o conceito teórico de dados ausentes e v) a categorização de dados ausentes em grupos.

2.1 Tipos de dados

Nesta seção descrevem-se os tipos básicos de dados existentes, uma vez que a utilização de determinado algoritmo pode ser inviabilizada pela incompatibilidade de formato esperada. Por exemplo, o algoritmo de regressão linear só faz previsões de atributos quantitativos contínuos, pois, tenta encontrar os valores que melhor se enquadrem em uma função linear (AWAD; KHANNA, 2015). Logo, se houver outro formato de dado na posição que deveria ser usada nos cálculos, o algoritmo não conseguirá fazer os cálculos necessários e dará erro.

Na literatura, há dois grandes grupos de dados, o Quantitativo, que é dividido em Discreto e Contínuo, e o Qualitativo, que é dividido em Nominal e Ordinal (SHIMAKURA, 2012), como pode ser visto na Figura 1.

Figura 1 - Tipos de Dados

Fonte: Próprio autor

Dados do tipo Quantitativo são variáveis que possuem medidas escalares, ou seja, valores numéricos. A sua subdivisão chamada Discreta, suporta variáveis que contêm números inteiros, geralmente resultado de contagens, por isso, podem assumir um número finito ou mesmo infinito de valores. Por exemplo: número de canetas, número de filhos, etc. Enquanto a subdivisão chamada Contínua, contempla variáveis que assumem valores contínuos em uma reta real, assim sendo, valores fracionais. Por exemplo: altura, peso, tempo, etc.

Já dados do tipo Qualitativo (também conhecido por Categórico), são as variáveis que contêm categorias, características, classificações e não quantidades. Uma das subdivisões dos dados Qualitativos é chamada de Nominal e nesse tipo, a variável se caracteriza pela falta de ordenação possível entre as categorias. Por exemplo: sexo, doente / sadio, fumante / não fumante, etc. Adicionalmente, na subdivisão chamada de Ordinal, o dado se caracteriza pela ordenação evidente entre as categorias. Por exemplo: meses do ano, dias da semana, escolaridade, etc.

2.2 Tipos de atributos

Além dos tipos de dados descritos anteriormente, há também o conceito de tipos de atributos, diretamente relacionados a bancos de dados (SANCHES, 2005):

- composto: é um atributo composto por outros atributos e que pode ser dividido. Exemplo: um campo chamado endereço, pode ser subdividido em rua, número, apartamento, etc.
- simples: é o atributo não divisível, também chamado de atômico.
- monovalorado: é o campo que só tem um valor possível. Exemplo: idade, só é possível ter uma idade por vez.
- multivalorado: que pode ter várias respostas. Exemplo: línguas que uma pessoa fala, pode haver só uma resposta, como podem aparecer muitas, inglês, francês, espanhol, etc.
- derivado: são os atributos que podem ser derivados de outros atributos. Exemplo: data de nascimento e idade, tendo a data de nascimento em mãos, é possível saber a idade do indivíduo. Nesse exemplo, pode-se dizer que o

atributo idade é derivado da data de nascimento. Outro exemplo um pouco diferente, seria pensar em quantidade de funcionários de um departamento e departamento de cada funcionário.

- nulo: existem situações nas quais um campo não necessita, obrigatoriamente, um valor, como em número do apartamento, já que esse campo só deve ser preenchido se a pessoa morar em um apartamento. Além disso, o nulo, ou null, também é usado quando não se sabe o valor do atributo.

Com base nessas classificações, pode-se enquadrar praticamente qualquer forma de dado e com isso, é possível avaliar se determinada técnica é adequada ou não para tratar uma ausência em específico. A partir dessa compreensão, são introduzidos os conceitos para a análise e tratamento das ausências.

2.3 Limpeza de dados

Em situações reais, frequentemente, os conjuntos de dados que estão sendo analisados para extração de conhecimento podem estar incompletos, ruidosos e/ou inconsistentes. Mais especificamente, dados incompletos são aqueles que possuem informações ausentes para algum atributo. Os ruidosos são os que contêm valores considerados divergentes do esperado ou em inglês *outliers*. Por fim, os inconsistentes são aqueles que possuem discrepância semântica entre si, ou seja, em um campo em que se espera texto, aparece um número ou vice-versa (GOLDSCHMIDT, 2005). Pelo fato de dados ruidosos serem tão comuns, a limpeza se torna algo importante.

Segundo especialistas, uma das partes de fundamental importância em uma boa análise de dados é a preparação dos mesmos (LAURETTO, 2011) (CAETANO, 2018), uma vez que essa parte consome de 70% a 80% do tempo e do esforço do projeto como um todo (PRATI, 2019) (MARREIROS; OLIVEIRA, 2000). A preparação é caracterizada pela coleta, limpeza, normalização e combinação de dados para análise, estando o processo de trabalho com dados ausentes contido dentro da limpeza (FERREIRA, 2017).

A fase de limpeza de dados, do inglês *Data Cleaning* ou *Data Cleasing*, é caracterizada pelo entendimento e correção dos dados, verificação da consistência das informações, preenchimento ou exclusão de valores ausentes, e eliminação de valores fora do domínio. Essa fase tem por objetivo corrigir os dados, prevenindo

consultas desnecessárias ou problemas, para que quando algoritmos de predição, consulta e/ou classificação vierem a ser usados, tais inconsistências impactem menos o desempenho (GOLDSCHMIDT, 2005).

O processo de análise dos dados pode continuar sem, necessariamente, atuar sobre as ausências, ou seja, apagar os registros que contenham uma ou mais omissões e trabalhar somente com os dados completos. Apesar de ser prudente fazê-lo em alguns casos, há uma redução da amostra, podendo inviabilizar a análise pela pouca quantidade de informação. Foi demonstrado que se 10% dos dados estão aleatoriamente ausentes em um conjunto de cinco variáveis, em média, quase 60% dos registros (ou linhas) terão pelo menos um valor omissos. Assim sendo, se somente os dados completos forem considerados, a amostra será reduzida para 40% do tamanho original (HAIR JR. et al., 2014).

2.4 Dados ausentes

Conforme descrito na Introdução, dados ausentes são caracterizados pela falta de informação em um determinado campo.

Segundo (OSBORNE, 2013), dados ausentes podem acontecer por diversos motivos, tais como: (i) falha dos participantes no preenchimento de questionários (conscientemente ou não), exemplo: o participante pode não ter visto um campo a ser preenchido ou simplesmente ignorado; (ii) descontinuidade de resposta, exemplo: o participante só preenche a primeira parte do estudo; (iii) problemas em equipamentos de coleta de dados ou ainda nos mecanismos de gravação, ou seja, falha em *hardware* e/ou *software*. Por exemplo, em uma determinada pesquisa há o campo “Última remuneração” (para que a pessoa informe o valor do último salário recebido), mas por qualquer que seja o motivo, um dos respondentes deixou este item em branco, logo, essa informação ao entrar no banco de dados se caracteriza como uma ausência.

Mesmo sabendo que essa falta de dado pode ser tratável, investigar sua causa geradora e corrigi-la é importante. Pois, o tratamento mesmo quando feito por um profissional experiente, provoca um consumo de tempo que não seria necessário se o dado estivesse presente ou completo. Logo, quanto maior o número de incompletudes, maior o tempo gasto em tratamento. Por isso, se a correção da causa raiz é importante.

Segundo (GOLDSCHMIDT, 2005), a melhor maneira de evitar a poluição da base, ou seja, de encontrar dados incompletos, ruidosos ou inconsistentes em uma

população ou amostra, é monitorar e organizar a entrada dos dados, especialmente em rotinas de interfaces críticas. No entanto, no dia a dia de grandes instituições essas rotinas podem não estar disponíveis para alterações ou, ainda, a causa do problema é alguma falha humana.

Na literatura encontram-se diferentes categorias de omissões, contudo, uma das divisões mais importantes aceitas foi feita por Donald B. Rubin e Roderick Joseph A. Little e classifica a omissão em três diferentes categorias, Figura 2: i) valores ausentes gerados completamente ao acaso (*Missing Completely At Random*); ii) valores ausentes gerados parcialmente ao acaso (*Missing At Random*) e iii) valores ausentes gerados não ao acaso (*Missing Not At Random*) (VINHA; LAROS, 2018).

Figura 2 - Tipos de ausências

Fonte: Próprio autor

2.4.1 *Missing Completely At Random*

Ausências completamente randômicas (MCAR) são caracterizadas quando não há nenhum padrão na omissão desses dados. Adicionalmente, pode-se dizer que os faltantes não estão relacionados a nenhum valor específico ou um conjunto de respostas observadas (KANG, 2013). É difícil encontrar ausências realmente randômicas, pois, somente fenômenos verdadeiramente aleatórios podem produzi-las (BOCK, 2019), como jogar um dado não viciado ou algum fenômeno da natureza. No entanto, se houver ausências por causa de uma falha de equipamento, porque as

amostras foram perdidas em trânsito ou tecnicamente insatisfatórias, tais dados são considerados MCAR (KANG, 2013).

Para garantir que as ausências de uma amostra são do tipo MCAR, existe uma técnica formal denominada *Little's MCAR test*. Quando o valor p calculado no teste for insignificante ($p < 0,05$) (LITTLE, 1988), pode-se assumir que a ausência é do tipo MCAR e, com isso, assumir que aquela ausência não é importante na análise final (GARSON, 2015). Um método informal de avaliação é predizer a omissão usando qualquer outra técnica que não o teste de Little (exemplo: senso comum; regressão; etc.), se isso for possível de se fazer, significa que a ausência não é do tipo MCAR (BOCK, 2019).

2.4.2 *Missing At Random*

Já as ausências parcialmente randômicas (MAR), também conhecidas por ausência ignorável (OSBORNE, 2013), embora o nome randômico passe uma ideia de aleatoriedade, não é isso o que ocorre, pois, a resposta para aquele valor ausente pode ser determinada com base nos valores contidos na população de dados (SILVA, 2010). Por exemplo, em uma pesquisa qualquer os homens são mais propensos a deixar um item sem resposta, e o pesquisador coletou dados sobre o gênero dos entrevistados, então os valores faltantes seriam considerados MAR (OSBORNE, 2013). As omissões randômicas são mais comuns que as totalmente randômicas, pois, a segunda depende de situações aleatórias, o que é mais difícil de acontecer.

Pode-se afirmar que uma omissão é do tipo MAR quando: i) o valor p é significativo no teste MCAR de Little ou ainda, ii) quando a imputação dessa ausência pode ser prevista utilizando outras variáveis observadas na base e não depende de variáveis não observadas (GARSON, 2015).

2.4.3 *Missing Not At Random*

Por fim, os dados ausentes do tipo não randômicos (MNAR), também conhecidos como ausência não-ignorável, é a omissão mais problemática, uma vez que esta forma aparece quando os dados ausentes não são nem MCAR nem MAR.

Este tipo acontece quando a omissão depende, pelo menos em parte, de características não observadas na atual população de dados (GARSON, 2015).

Para tratar esse tipo de omissão de dados, pode-se obter informações por meio da conferência do histórico da previsão do tempo, ou mesmo da análise das informações do Instituto Brasileiro de Geografia e Estatística (IBGE), entre outras que podem nos ajudar com ausências MNAR. Exemplo: renda média por bairro, temperatura em determinada cidade, chuva em certa região, etc.

A título de curiosidade, tratando-se de frequência, é possível dizer que a ausência mais comum é a MNAR, seguida de MAR e a menos comum é a MCAR (OSBORNE, 2013).

2.5 Considerações finais do capítulo

Neste capítulo foram apresentados os conceitos de tipos de dados que podem ser quantitativos e qualitativos, sendo o primeiro dividido em dados discretos e contínuos, e o segundo, em nominais e ordinais. Adicionalmente, apresentaram-se os tipos de atributos existentes, que são mais relacionados a bancos de dados, são eles: composto; simples; monovalorado; multivalorado; derivado e nulo. Além disso, foi apresentado os conceitos básicos sobre limpeza de dados, no qual a intenção é corrigir as inconsistências para que haja o menor impacto possível nas análises a serem feitas. Por fim, foi discutido os tipos de ausência de dados MCAR, MAR e MNAR e suas características.

No próximo capítulo, são apresentadas as técnicas mais utilizadas para se tratar ausências em bases de dados e também são feitos comparativos entre estes métodos, a fim de exemplificar quando é recomendado ou não utilizar um método.

3 TRATAMENTOS

3.1 Formas de tratamento

A melhor forma possível para lidar com ausência de dados é prevenir o problema planejando bem o estudo e coletando os dados cuidadosamente (KANG, 2013). Contudo, talvez não seja possível fazer essa prevenção, por vários motivos, como: falta de equipe especializada para encontrar e corrigir o erro, ou sistemas “caixa preta”. Por isso, existem algumas formas de tratamento. O tipo do dado juntamente com o tipo da ausência, fornecem indicativos de qual será o tratamento mais adequado, com o intuito de obter os melhores resultados. Segundo a literatura, existem dois grandes grupos de tratamento de ausências: Exclusão ou Preenchimento, que podem ser subdivididos em mais algumas subcategorias como mostrado na Figura 3.

Figura 3 - Métodos para tratamento de dados ausentes

Fonte: Próprio autor

3.1.1 Exclusão

A exclusão, ou deleção, é um procedimento usado quando o número de casos de dados inexistentes é pequeno (arbitrariamente definido como menos de 5% da amostra), sendo comum simplesmente descartar esses registros da análise

(GARSON, 2015). Na literatura, pode-se encontrar dois métodos amplamente conhecidos que tratam a exclusão de forma diferente: Exclusão de Lista e Exclusão de Pares.

A abordagem mais comum para tratamento de faltantes é a Exclusão de Lista, em inglês *Listwise*, na qual se omite os casos com omissões e se analisa o restante (KANG, 2013). Esta forma de tratamento é muito simples, pois, todos os casos que contenham um ou mais valores ausentes são excluídos, ou seja, apenas observações completas são consideradas na análise (VINHA; LAROS, 2018). Esta abordagem é conhecida como caso completo (ou caso disponível) e é o método mais usado para lidar com dados ausentes e, portanto, tornou-se a opção padrão análise na maioria dos pacotes de software estatístico (KANG, 2013). Normalmente é usado quando se quer comparar a diferença do resultado final da análise utilizando dados preenchidos completos e dados sem tratamento completos (STATISTICS SOLUTIONS, 2019).

Exclusão de Pares, em inglês *Pairwise*, visa atenuar a perda que ocorre na exclusão de listas (STATISTICS SOLUTIONS, 2019). Nesta abordagem, elimina-se informações apenas quando os dados específicos necessários para testar uma suposição específica estão ausentes. Se houver dados faltando em outro lugar no conjunto de dados, os valores existentes são usados no teste (KANG, 2013). Na exclusão de pares, a quantidade de dados utilizada na análise é maior, pois, consideram-se as observações que contenham pares de variáveis completas, ao invés do conjunto inteiro de informação do indivíduo. Tal método é comumente usado quando se empregam técnicas baseadas em correlações, pelo fato de que a estimação da correlação depende de um par de valores. Contudo, nessa abordagem as amostras utilizadas nos diferentes cálculos não são as mesmas, gerando assim críticas de alguns pesquisadores (VINHA; LAROS, 2018). O procedimento não pode utilizar uma variável específica quando ela possui uma incompletude, mas ainda assim pode usar o caso ao analisar outras variáveis com valores não ausentes. Por exemplo: um caso contém três variáveis: VAR1, VAR2 e VAR3. A variável VAR1 é nula, mas isso não impede que alguns procedimentos estatísticos usem o mesmo caso para analisar as variáveis VAR2 e VAR3. Logo, a exclusão de pares permite a maior utilização de dados (IBM, 2019b).

3.1.2 Preenchimento

O preenchimento é utilizado quando a quantidade de ausências não é muito alta (alguns autores dizem não mais que 20% enquanto outros pesquisadores dizem não mais que 50%) e também, quando as outras variáveis da amostra têm a capacidade de prever as faltantes (GARSON, 2015). Na literatura pode-se encontrar várias técnicas de preenchimento, a seguir lista-se algumas delas:

- Preenchimento manual de valores: essa técnica pode ser aplicada em qualquer tipo de dado e de ausência, uma vez que depende do conhecimento prévio do especialista do domínio e/ou responsável pela base, e talvez até de uma fonte adicional de informação para a captação desses dados. Contudo, essa é uma abordagem que demanda grande quantidade de tempo e esforço, portanto, quando se tem alto volume de ausências, ela se torna impraticável. Nesse método, o conhecedor dos dados deve completar via digitação, caso a caso, as omissões existentes (GOLDSCHMIDT, 2005).
- Preenchimento utilizando valores globais e constantes: Esse método é caracterizado pela substituição dos valores ausentes por um valor padrão tal como “desconhecido” ou “null” ou qualquer outra coisa nesse sentido. Esse valor substituto deve ser escolhido pelo especialista do domínio da aplicação, para que faça sentido no contexto dos dados. Vale ressaltar que, em determinados algoritmos de mineração de dados, a constância de certos valores pode significar algo importante na análise, ou seja, pode enviesar a análise. Então, dependendo do algoritmo a ser usado para obtenção da análise final, esse não é o método mais indicado (GOLDSCHMIDT, 2005).
- Preenchimento utilizando medidas estatísticas: Uma alternativa viável ao preenchimento de ausências por utilização de constantes padrões é a utilização de medidas estatísticas, como: média para dados do tipo quantitativo e moda para dados qualitativos. Uma variante desse método pode ser aplicada em problemas de classificação. Nessa situação, as medidas estatísticas são inseridas de acordo com cada classe especificamente, assim, o cálculo da medida estatística que será imputado no lugar da ausência fica restrito ao conjunto de dados pertencente à classe em que a omissão se encontra (GOLDSCHMIDT, 2005). O preenchimento utilizando a média ainda é muito

utilizado, todavia, gera distorção mesmo quando os dados são do tipo MCAR. Por isso, esse método é possivelmente um dos piores tratamentos para dados ausentes (VINHA; LAROS, 2018).

- Preenchimento utilizando métodos de aprendizado de máquina: Esse tipo de técnica se caracteriza pela criação de modelos preditivos que sugerem os valores mais prováveis para o preenchimento dos dados ausentes. Para a construção dos modelos existem alguns algoritmos de aprendizado de máquina amplamente conhecidos, como redes neurais, modelos bayesianos, k vizinhos mais próximos (*K-Nearest Neighbors* - KNN), maximização da esperança (*Expectation Maximization* - EM) e árvores de decisão. Fazendo um comparativo com os métodos anteriormente discutidos, esta técnica, juntamente com a técnica de imputação por medidas estatísticas, utiliza informações existentes para prever as omissões. Enquanto os outros métodos são baseados em conhecimento prévio dos dados em si, ou seja, de onde vêm, como foram coletados e qual foi o público respondente e não necessariamente está relacionado com as informações encontradas na população. Há um ponto positivo nessa técnica, pois, quando atributos já contidos na base são utilizados para a predição de outros atributos, as chances de preservação de possíveis relacionamentos entre eles são maiores (GOLDSCHMIDT, 2005).

Vale salientar que o preenchimento das lacunas visa preservar as características da base de dados e, conseqüentemente, dar suporte aos posteriores estudos e não prejudicar as análises subsequentes. Outro ponto importante é que, somente quando o dado faltante está contido em um atributo utilizado no estudo é necessária atuação/tratamento. Dessa forma “economiza-se” tempo de entendimento, de processamento e, quando se trata de deleção, poupam-se linhas. Além disso, os quatro primeiros métodos (deleção, preenchimento manual, preenchimento de constantes e preenchimento de estatísticas) devem ser utilizados com parcimônia, uma vez que podem tornar determinados atributos tendenciosos (GOLDSCHMIDT, 2005).

3.2 Comparativo: exclusão *versus* preenchimento

Após conhecer os tipos de tratamento existentes, o próximo passo é entender quais das metodologias apresentadas se adequam melhor a cada tipo de omissão, pois, dependendo do tipo da omissão e da sua frequência na base dados, um ou outro tratamento é mais efetivo.

Na situação em que há uma omissão do tipo MCAR, a exclusão é um dos métodos mais indicados, pois, seu preenchimento se torna muito complexo, pelo fato de que este dado representa um fenômeno aleatório (GARSON, 2015). Em situações nas quais o tamanho da amostra é grande e o número de faltantes é relativamente pequeno (menos de 5% do total de casos) o método de exclusão de listas se mostra tão eficiente quanto técnicas mais sofisticadas para omissões do tipo MCAR (IBM, 2019a) (VINHA; LAROS, 2018). Contudo, a exclusão reduz os casos de MCAR consideravelmente e conseqüentemente o tamanho da amostra, ao mesmo tempo que aumenta a quantidade de erros padrão, aumentando assim a chance de falsos negativos.

Enquanto isso, em uma ausência do tipo MAR a exclusão não é muito recomendada, pois, pode enviesar a base de dados (GARSON, 2015). Por outro lado, o preenchimento é geralmente uma metodologia bastante indicada para esse tipo de ausência. Adicionalmente, se a omissão puder ser bem prevista a partir das variáveis observadas dentro da base de dados, então a imputação múltipla é o método mais indicado.

Finalmente, ao tratar uma omissão categorizada como MNAR, as variáveis da população contidas no banco de dados são preditoras inadequadas, pois, a omissão é correlacionada com variáveis que não foram anotadas, o que prejudica a eficácia dos métodos de imputação usuais, incluindo a imputação múltipla. Por isso, uma possível solução é inserir valores baseados em dados que seriam externos, como estimar faixa salarial tomando como base dados do IBGE associados ao endereço do respondente (GARSON, 2015).

3.3 Ferramentas de tratamento

Existem muitas ferramentas e linguagens de programação que podem ser usadas em projetos de ciência de dados para tratar dados perdidos, como: *Python*, *R*, *Matlab*, *Scala*, *SAS*, *Julia*, etc. Atualmente, as mais famosas e difundidas para este tipo de projeto são *R* e *Python*, pois, ambas são de código aberto e gratuitas. Contudo, nesta seção é tratado especificamente de *Python*, pois, é uma linguagem que possui várias bibliotecas já implementadas e com ampla utilização quando se trata de dados ausentes. Além disso, essa foi a linguagem escolhida para o desenvolvimento da prova de conceito.

Python é uma linguagem de programação, lançada em 1991 por Guido van Rossum, de propósito geral, multiparadigma, interpretada, *open source* e de alto nível. Com ela é possível criar programas complexos ou simples e escrever códigos mais rapidamente devido à sua semelhança com a linguagem humana (NEEDHAM, 2007). Sendo até comparada, por algumas pessoas, como um pseudocódigo executável (OLIPHANT, 2007). Atualmente uma das linguagens mais utilizadas por cientistas de dados em suas análises devido à sua facilidade de uso e da quantidade de pessoas que colabora desenvolvendo funções para ela.

Em *Python*, existem algumas bibliotecas e funções as quais auxiliam o tratamento de ausências, um exemplo desse tipo de biblioteca é a Pandas, um pacote voltado para análise de dados, que fornece estruturas de dados rápidas e flexíveis para facilitar o trabalho com dados estruturados (PANDAS...,). Na Tabela 1, seguem algumas funções contidas dentro dessa biblioteca que são utilizadas no tratamento de omissões.

Apesar de existirem formas de se encontrar e tratar omissões, em nenhuma destas há uma automatização total do processo, o que existe são os tratamentos pré-implementados que facilitam no quesito codificação da solução. Em cada situação o profissional escolhe o método que será aplicado dependendo do seu problema, ou seja, apesar de existirem maneiras de tratar a incompletude de forma mais simples, o especialista/conhecedor do dado é quem precisa saber identificar o tipo da ausência e os tratamentos possíveis e, então, utilizar a funcionalidade que mais se adequa à resolução do problema em questão. Portanto, novamente é reforçada aqui, a ideia de que conhecer bem a ausência em seus mais variados aspectos é o que de fato faz com que a análise seja efetiva.

Tabela 1- Funções para tratamento de omissões em Pandas/Python

NOME DA FUNÇÃO	O QUE FAZ	O QUE RETORNA
isna ()	Identifica se a célula está nula ou não.	Retorna <i>true</i> para nulo e <i>false</i> para não nulo.
notna ()	Identifica se a célula está preenchida ou não.	Retorna <i>true</i> para preenchido e <i>false</i> para não preenchido.
fillna ()	Preenche uma ou mais células nulas com um valor especificado, que pode ser um inteiro, uma <i>string</i> , a média, etc.	Retorna o <i>dataset</i> com a(s) célula(s) preenchida(s) com o valor especificado.
dropna ()	Apaga linha(s) ou coluna(s) que contenham ausência de dados.	Retorna o <i>dataset</i> já sem essa linha ou coluna.
interpolate ()	Preenche a célula nula através de um cálculo matemático tomando como base a célula anterior mais próxima preenchida e a célula posterior mais próxima preenchida.	Retorna o <i>dataset</i> com todas as células preenchidas.
replace ()	Preenche uma ou mais células com um valor especificado.	Retorna o <i>dataset</i> com a(s) célula(s) preenchida(s) com o valor especificado.

Fonte: Próprio autor

3.4 Considerações finais do capítulo

Na literatura, foi possível encontrar duas grandes classificações de formas de tratar omissões em populações de dados, que são: Deleção (ou Exclusão) e Imputação (ou Preenchimento). Essas divisões ainda podem ser quebradas em subcategorias, para Exclusão tem-se: Exclusão de Lista e Exclusão de Pares. E para Imputação tem-se: Preenchimento Manual, Preenchimento com valores globais e constantes, Preenchimento utilizando Estatísticas (exemplo: média e moda) e por último Preenchimento utilizando técnicas de Aprendizado de Máquina (exemplo: Redes Neurais, KNN, Modelos Bayesianos). Adicionalmente, foi feito um comparativo entre os tratamentos listados anteriormente para cada forma de ausência descrita no Capítulo 2 (MCAR, MAR e MNAR). Por fim, foram mostradas algumas técnicas baseadas em códigos (funções Python) que auxiliam no tratamento do dado faltante.

4 PROPOSTA

A partir da leitura dos capítulos anteriores, foi possível compreender os conceitos básicos sobre dados faltantes. Como visto na Introdução, mais especificamente na sessão de Objetivos, a ideia desta dissertação de mestrado é verificar como o aumento das ausências em uma base de dados impacta na eficiência de uma análise, ou seja, mostrar se há uma correlação entre a ausência do dado e a qualidade final do processo de análise. Por isso, neste capítulo descreve-se, detalhadamente como foi o processo de prova de conceito para a pergunta de pesquisa, desde a organização das atividades, a escolha da base de dados, do algoritmo, etc.

4.1 Planejamento

Uma parte importante para nortear o desenvolvimento desta proposta foi estruturar o que deveria ser pesquisado ou executado. Na Tabela 2 seguem os passos desse levantamento:

Tabela 2 - Sequência de passos para o desenvolvimento do projeto

PASSO	DESCRIÇÃO
1	Levantar os tipos de ausências e as causas.
2	Entender as formas de tratamento dessas ausências.
3	Contextualizar a importância desse estudo.
4	Definir o escopo e os cenários a serem testados.
5	Estudar possíveis algoritmos de predição.
6	Escolher uma base de dados para testar a análise.
7	Escolher o algoritmo que melhor se encaixa no desafio.
8	Escolher as ferramentas a serem utilizadas.
9	Implementar o algoritmo de predição testando os cenários.
10	Analisar o resultado do teste.

Fonte: Próprio autor

4.2 Cenários

O objetivo da criação de cenários foi mostrar as possibilidades de inserção de ausências e como isso impactaria o resultado da análise preditiva para, a partir da análise desses cenários, provar a correlação entre falta de informação e diminuição de acertos do algoritmo. Desta forma, os cenários considerados foram os seguintes:

Cenário 1: o modelo é treinado com dados completos e o algoritmo é submetido ao teste também com dados completos.

No cenário 1, o algoritmo foi treinado e testado com todos os dados, sem nenhuma omissão. Esse é o melhor cenário da análise, o ideal, quando se trabalha com processamento e análise de dados. Por isso, espera-se que a assertividade seja a maior entre todas as situações levantadas e constitua a base de comparação com os demais cenários.

Cenário 2: o modelo é treinado com dados completos, mas o algoritmo é depois submetido a ausências.

No cenário 2, treinou-se o modelo com dados completos e testou-se com falta de informações. Esse cenário é o que normalmente acontece no dia a dia de pesquisadores, cientistas de dados e outros profissionais que trabalham com dados. O treino aqui representa o cenário controlado do ambiente de desenvolvimento, enquanto, o teste representa a situação real da informação, em produção, onde o modelo é validado de verdade.

Cenário 3: o modelo é treinado com falta de dados e testado com dados completos.

No cenário 3, o modelo foi treinado com dados faltantes, contudo, no teste, os dados estavam completos. Essa é possivelmente a situação mais improvável de ocorrer no cotidiano de um profissional. Pois, significa que o cenário previsto para testes foi mal preparado, que não condiz com a realidade dos dados.

Cenário 4: o modelo é treinado e submetido a dados ausentes.

No cenário 4, o algoritmo foi treinado e testado utilizando-se dados contendo ausências. Essa situação é incomum no dia a dia, pois, significa que os dados reais

são tão ruins que a melhor alternativa é treinar o modelo pensando nessa falha e inserindo as ausências no treino. Na teoria, esse é um dos piores cenários para a acurácia do algoritmo, por isso, os resultados dessa avaliação devem ser ruins, comparado aos demais. Assim como o cenário anterior, esse também é um cenário com possível baixa acurácia pela falta de dados na hora do treinamento.

A definição dos cenários constitui a parte central do desenvolvimento do algoritmo de correlação, logo, também são parte importante no processo de validação da hipótese. Por isso, segue um resumo os cenários, Tabela 3:

Tabela 3 - Resumo dos cenários

CENÁRIO	DADOS EM TREINAMENTO	DADOS EM TESTE
1	Completo	Completo
2	Completo	Parcialmente ausente
3	Parcialmente ausente	Completo
4	Parcialmente ausente	Parcialmente ausente

Fonte: Próprio autor

Além disso, nos cenários 2, 3 e 4 há subcenários que são relacionados à porcentagem de ausências inseridas, que variam de 10% a 60%, ou seja, ao invés de quatro cenários, têm-se ao todo dezenove cenários, um para o cenário completo e seis para cada cenário de omissão. Essa estratégia permite analisar se a qualidade do dado piora ou não com o aumento da falta da informação em cada situação descrita.

4.3 Base de dados

Para garantir a replicabilidade deste estudo, além de disponibilizar o código fonte desenvolvido, decidiu-se também utilizar uma base de dados mundialmente famosa no universo da ciência de dados, o *dataset* da Flor de Íris (SCIKIT LEARN, b). O conjunto de dados Flor de Íris também é conhecido como conjunto de dados Íris de Fisher. Esse conjunto foi apresentado à comunidade científica em 1936 pelo estatístico e biólogo Ronald Fisher no artigo “O uso de múltiplas medições em

problemas taxonômicos” (SC.D.; F.R.S., 1936), como um exemplo de análise discriminante linear.

Esse *dataset* é composto por 150 amostras, cinquenta de cada uma das três espécies de Íris (Setosa, Versicolor e Virgínica), como se pode ver no Anexo A, Figura 5. De cada espécie, foram medidas quatro variáveis: o comprimento e a largura das sépalas (*Sepal*) e das pétalas (*Petal*), em centímetros (Figura 4). Baseado nessas quatro variáveis, Fisher criou um modelo para distinguir as espécies umas das outras.

Figura 4 - Flor de Íris

Fonte: Retirado de <https://rpubs.com/vidhividhi/irisdataeda>

Existem diferentes formas de obter essa base de dados, contudo, para agilizar e facilitar o processo de desenvolvimento, a cópia utilizada neste projeto foi obtida por meio do *dataset* disponível na biblioteca *Python* chamada *Scikit Learn* (SCIKIT LEARN, b). Sua utilização na implementação, pode ser vista no Anexo B.

4.4 Técnica preditiva

Para verificar o impacto entre falta de dados e qualidade da análise, foi preciso implementar uma solução na qual fosse possível fazer um comparativo com o resultado esperado e o resultado real. Por isso, escolheu-se o estudo e a utilização de técnicas preditivas de aprendizado de máquina supervisionado, que é um subtipo de IA. Esse tipo de técnica propicia resultados que facilitam a comparação tanto entre os vários cenários, quanto com o valor esperado de resposta.

Mas o que é aprendizado de máquina supervisionado? O aprendizado de máquina supervisionado é qualquer processo de aprendizagem, na qual a máquina

aprende uma função baseada no entendimento dos dados de entrada e de saída. Classificação e regressão são os dois típicos exemplos de aprendizagem supervisionada. Nesses dois casos, a saída para cada um deles são respectivamente valores categóricos (classes) e numéricos. A grande diferença entre o aprendizado supervisionado para o não supervisionado é que, na segunda opção, o algoritmo busca aprender os padrões por meio de recompensa, tentativa e erro, sem os exemplos do que é certo ou não (SUPERVISED..., 2011).

Quando se fala de aprendizado de máquina supervisionado existem muitos algoritmos disponíveis, então, encontrar aquele que serviria de base para esse estudo foi um pequeno desafio. A escolha da linguagem de programação na qual a implementação seria feita, também foi determinante no levantamento dos algoritmos candidatos. Por haver vasta documentação, grande quantidade de algoritmos e, ainda, devido ao conhecimento que a mestrandia já possui nessa linguagem, decidiu-se por programar a execução do projeto em *Python*. O próximo passo foi pesquisar com mais profundidade sobre o tema aprendizado de máquina e conectar a proposta desse projeto de pesquisa com especificidade do algoritmo.

A partir disso, selecionaram-se três opções de algoritmos para uma análise mais aprofundada, que são: *Cat Boost*, *Logistic Regression* e *Sagemaker DeepAR*. Na Tabela 4, são mostradas as características mais relevantes de cada um deles.

Tendo visto as características de cada algoritmo, escolheu-se implementar o *Logistic Regression*, pelos seguintes motivos: i) simples implementação; ii) permite a utilização da base de dados de Íris, que está disponível na internet; iii) não demanda grandes gastos para sua implementação; iv) não requer infraestrutura específica e v) tem vasta documentação disponível na internet.

A escolha de algoritmo apresentou algumas dificuldades pelo fato de a maior parte dos algoritmos de predição não aceitarem atributos vazios como entrada (NaN). Essa foi parte a mais desafiadora do processo de desenvolvimento, pois, não havia algoritmos que não precisassem de alterações. Então, decidiu-se substituir as ausências por zeros para que a implementação pudesse acontecer. A decisão pela inserção de zeros se deu, pois, a base de dados de Íris contém medições de comprimento/largura e nenhuma dessas variáveis pode ser zero na prática, então inserir zero significa inserir algo irreal. Logo, na parte do código onde as omissões são inseridas é possível notar que foram inseridos zeros e ao invés de 'NaN'.

Tabela 4 - Comparativo entre algoritmos preditivos

ALGORITMO	CARACTERÍSTICAS	ACEITA AUSÊNCIAS
Cat Boost	Este algoritmo criado pelos engenheiros e pesquisadores da Yandex (RESEARCHERS,) e é um projeto open source, do tipo gradient boosting com árvores de decisão. Isso significa que ele faz várias árvores de decisão diferentes para fazer a predição e, então, faz uma “média” para poder encontrar a previsão do que se espera.	Não
Logistic Regression	Esse algoritmo se baseia no conceito de regressão logística, que é muito utilizado para fazer categorizações ou agrupamentos. Este modelo é útil para se expressar a relação entre uma variável dependente categórica e uma ou mais variáveis independentes (LOPES, 2004). Além disso, está disponível na biblioteca Scikit Learn (SCIKIT LEARN, a).	Não
DeepAR	Esta solução foi desenvolvida pela AWS e está disponível somente dentro de um dos serviços do Sagemaker, que é pago. Além disso, é um algoritmo de <i>forecasting</i> (que é a previsão de algo baseado em séries temporais), ou seja, dada a somatória de algo em determinado período de tempo, acontece a predição dessa somatória para datas futuras na mesma frequência do período dado (AWS,).	Não

Fonte: Próprio autor

4.4.1 Algoritmo de regressão logística

Em modelos de regressão linear ($ax+b=y$) a variável dependente, Y , é uma variável aleatória de natureza contínua. Contudo, há situações em que a variável dependente é qualitativa e expressa por duas ou mais categorias, por exemplo maçã e banana. Nesses casos, a regressão linear não funciona tão bem e uma solução viável é a utilização da regressão logística, que permite o uso de um modelo de regressão para calcular ou prever a probabilidade de um evento específico (FIGUEIRA, 2006).

No caso da base de dados de Íris, no qual se tem três classificações possíveis (Setosa, Versicolor e Virgínica) o modelo de regressão logística utilizado é o chamado multinomial, pois, prediz mais de duas categorias.

4.4.2 Assertividade do algoritmo

Um aspecto diferencial quando se fala de algoritmo de aprendizado de máquina supervisionado é poder avaliar a assertividade do mesmo, ou seja, quanto o resultado predito se aproxima do real. Por isso, existem algumas técnicas que dão essa dimensão e que foram utilizadas para posterior avaliação dos resultados, são elas: matriz de confusão e acurácia.

A matriz de confusão é uma maneira visual de representar os acertos e os erros de uma classificação. Na maior parte das vezes, esta é uma matriz bidimensional indexada em uma dimensão pela verdadeira classe do objeto e na outra dimensão, pela classe atribuída pelo algoritmo (TING, 2017). Contudo, para o caso tratado nesta dissertação, foi necessária a utilização de uma matriz 3x3, pois, temos 3 três espécies diferentes de Íris (Íris Setosa, Íris Versicolor, Íris Virgínica). A matriz de confusão deste estudo tem a estrutura mostrada na Tabela 5. Verdadeiro Positivo significa a quantidade de acertos que o teste do modelo teve, enquanto o Falso Negativo significa a quantidade de erros. O cenário ideal para matriz de confusão é a matriz diagonal.

Tabela 5 - Matriz de confusão da flor de Íris

		Predito		
		Íris Setosa	Íris Versicolor	Íris Virgínica
Real	Íris Setosa	Verdadeiro Positivo	Falso Negativo	Falso Negativo
	Íris Versicolor	Falso Negativo	Verdadeiro Positivo	Falso Negativo
	Íris Virgínica	Falso Negativo	Falso Negativo	Verdadeiro Positivo

Fonte: Próprio autor

Já a segunda medida de assertividade de um algoritmo é a acurácia. Tal termo é comumente usado no contexto de classificação, para se referir a uma medida do grau em que as previsões de um modelo correspondem à realidade. O cenário ideal para a acurácia é 1. A precisão de um algoritmo pode ser calculada como (ACCURACY..., 2017):

$$\text{acurácia} = (\sum \text{acertos}) / (\sum \text{registros})$$

4.5 Considerações finais do capítulo

Nesse capítulo foi descrita a proposta deste trabalho de pesquisa, como aconteceu o desenvolvimento do projeto. Contou-se do planejamento, qual foi o passo a passo para chegar até a conclusão do projeto. Adicionalmente, foram listados os cenários a serem testados para validação da proposta, que são: 1) quando o modelo é treinado com dados completos e o algoritmo é submetido ao teste também com dados completos; 2) quando o modelo é treinado com dados completos, mas o algoritmo é depois submetido a ausências; 3) quando o modelo é treinado com falta de dados e testado com completos e 4) quando o modelo é treinado e submetido a dados ausentes. Além disso, para os cenários 2, 3 e 4 descreve-se a progressão na quantidade de faltantes, que vai de 10% a 60%. Outro ponto importante descrito, foi a base de dados utilizada para testar o modelo, que é a base de Íris. Também se discutiu sobre o porquê da utilização e características de uma técnica preditiva neste estudo, como aconteceu a escolha do algoritmo, o que é o algoritmo de regressão logística e como medir sua assertividade (matriz de confusão de acurácia).

O capítulo que se segue, trata do detalhamento da implementação do algoritmo.

5 IMPLEMENTAÇÃO

Nesse capítulo, é explicado o passo-a-passo da implementação e da execução do algoritmo que serve de base para este estudo. Para ver o código na íntegra, siga para o Anexo B.

A implementação foi feita pela própria mestrandia dentro da plataforma de programação colaborativa do Google, chamada de Google Colab (GOOGLE, 2020). Essa plataforma foi escolhida pelos seguintes motivos: facilidade de uso; estar na nuvem, ou seja, pode ser acessado de qualquer lugar; ter salvamento automático e ter uma versão gratuita.

Após a definição da ferramenta de codificação, dos cenários, da base de dados e da linguagem de programação, iniciou-se a implementação da solução, que foi pensada e construída em quatro grandes fases: i) entendimento do *dataset* e separação das variáveis; ii) treinamento e teste do modelo; iii) inserção das omissões, que são caracterizadas pelos zeros; iv) implementações dos cenários descritos na seção 4.2; v) obtenção dos resultados.

5.1 Dataset

Na seção 1, relacionada ao entendimento do *dataset* e a separação das variáveis, a primeira coisa feita foi a importação do *dataset* e, em seguida, a separação em duas variáveis, X e y .

```
#import da biblioteca que contém os dados de Íris  
from sklearn.datasets import load_iris  
  
#carregamento dos dados de Íris em X e y  
X, y = load_iris(return_X_y=True)
```

A variável X é preditora e a y a variável alvo. No mundo de ciência de dados, variável preditora é o conjunto de informações que serão usadas como base para a predição. Enquanto, a variável alvo é aquela que queremos prever e será utilizada como base de comparação para saber se o algoritmo desenvolvido está acertando ou

não. Especificamente para o *dataset* de Íris, as colunas *sepal length*, *sepal width*, *petal length* e *petal width* fazem parte da variável preditora (X). Pois, a partir da análise dessas colunas, que se chegará na conclusão da variável alvo (y) que é o tipo de Flor de Íris.

Após obter o *dataset* de Íris e dividi-lo em X e y , o passo seguinte é separar em dados de treino e teste. Esse método de separação se chama validação cruzada (em inglês *cross-validation*) e consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos de um determinado tamanho e, a partir daí, $k-x$ subconjuntos são utilizados para teste e os restantes são utilizados para estimação dos parâmetros, fazendo-se o cálculo da acurácia do modelo, o que é chamado nesse estudo de treino (BROWNE, 2000).

#separação da base em dados de treino e de teste

```
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y)
```

Isso significa que o modelo será treinado e validado com 75% dos dados e os 25% restantes serão utilizados para avaliar a efetividade da predição. Esses valores, 75% e 25%, são valores comumente utilizados no treinamento de modelos de aprendizado supervisionado.

Para facilitar a codificação do processo, os dados que estavam em um tipo de *array*, foram convertidos para o formato *dataframe* da biblioteca Pandas do *Python* (PANDAS...,), o que facilita o manuseio dos dados.

#criação de dataframe de treino com os dados de X

```
dfX=pd.DataFrame(X_treino, columns= ['sepal length', 'sepal width', 'petal length', 'petal width'])
```

#criação de dataframe de treino com os dados de y

```
dfy=pd.DataFrame(y_treino, columns=['resultados'])
```

A partir desse ponto, a ideia é entender um pouco melhor o dado, olhando para as suas características como total, somatório, média, mediana, desvio padrão, etc. Para tal, foi utilizada a função *Describe* da biblioteca Pandas, analisando primeiramente a variável X .

```
#função que mostra informações de somatório, média, etc, do dataframe de X
print (dfX.describe())
```

Com a execução desse código, pode-se notar que a nossa variável preditora (dfX) possui 112 linhas e não há ausências em nenhuma coluna. Além disso, é possível observar as que a coluna *sepal length* tem a maior média. Ademais, é importante notar os itens de mínimo e máximo, pois, esses itens estarão refletidos no gráfico *boxplot*.

Em seguida, verifica-se a variável *y*, pois, ela precisa ter a mesma quantidade de linhas que *X*. Também, confere-se se está tudo certo com os mínimos e máximos, 0 e 2 respectivamente, conforme esperado.

```
#função que mostra informações de somatório, média, etc, do dataframe de y
print (dfy.describe())
```

A última parte desta fase é a visualização do dado como histograma e *boxplot*. Ambos os métodos de visualização servem para analisar a distribuição dos dados, contudo, o histograma é mais voltado para a quantidade de itens por centímetros (nesse caso) e o *boxplot* para observação de *outliers*.

Primeiro visualiza-se o histograma da variável *X*, no qual é possível notar uma distribuição de quantidades por centímetros quase que aleatória em *sepal length* e em *petal width*. Enquanto em *sepal width* e em *petal length* é possível ver algo parecido com uma distribuição normal (Anexo B).

```
#criação e visualização do histograma de dfX
dfX.hist()
pl.show()
```

Já no histograma da variável *y*, conforme esperado as quantidades estão contidas em 0, 1 e 2. Observa-se, que o total de itens em cada torre é praticamente igual, o que é muito importante para garantir que não há uma tendência para determinado tipo de Íris (Anexo B).

#criação e visualização do histograma de dfy

```
dfy.hist()
pl.show()
```

Agora falando do gráfico *boxplot* para *X*, nota-se que para a coluna *sepal width* tem um valor fora do esperado, um *outlier*. Tal observação é relevante, pois, valores fora do esperado costumam impactar no treinamento do modelo, gerando resultados errados.

#visualização do gráfico de boxplot para dfX

```
dfX.boxplot()
```

Enquanto no *dataframe* de *y*, nota-se que não há *outliers*, conforme esperado, pois, os resultados esperados são: 0, 1 ou 2.

#visualização do gráfico de boxplot para dfy

```
dfy.boxplot()
```

5.2 Treinamento do modelo

Já na segunda parte da implementação, há primeiramente o treinamento do modelo com dados completos, que nada mais é que chamar a função *Logistic Regression* (SCIKIT LEARN, a) parametrizada e passar as variáveis *X* e *y* de treino. Esses parâmetros foram escolhidos de forma a facilitar o processo de predição.

#treinamento do modelo de regressão logística usando X_treino e y_treino

```
clf=LogisticRegression(random_state=0,solver='newton-cg',
    multi_class='multinomial').fit(X_treino, y_treino)
```

Nesta fase também foi feito o teste do modelo, também sem ausências. Que nada mais é do que utilizar as informações da coluna preditora como entrada da função e assim gerar uma predição.

```
#teste do modelo de regressão logística usando a variável X_ teste
pred=clf.predict(X_teste)
```

5.3 Inserção de ausências

Na fase 3, criou-se uma função para inserir as ausências, que conforme falado anteriormente, são os zeros. Para essa função, decidiu-se utilizar os faltantes do tipo MCAR, que são os totalmente randômicos, sendo que em situações cotidianas nesse tipo de dado faltante geralmente é aplicado é exclusão. Escolheu-se o MCAR a fim de mostrar que dados incompletos ainda podem ser úteis.

Nessa função, a primeira coisa a ser feita é saber a quantidade de registros totais do *dataset*, pois, este valor pode variar dependendo da divisão feita entre treino e teste. Depois, calcula-se o número de zeros que deverão ser inseridos baseado na porcentagem de ausências que se deseja ter, utilizando-se a seguinte função matemática:

$$Qtde_zeros = (numero_linhas \times numero_colunas) \times porcentagem / 100$$

Por fim, insere-se aleatoriamente N zeros, sendo N baseado no resultado da função matemática anterior. Utilizando-se a função *Random*, da biblioteca de mesmo nome, para que a inserção de zeros seja aleatória e assim não enviesar o modelo preditivo.

```
#laço que repete tantas vezes quanto foi calculado pela função anterior chamada
calcula
for i in range(int(calcula)):

    #escolhe aleatoriamente uma coluna dentre o número total de colunas da base
    de dados
    rand_coluna= random.randrange(num_colunas)

    #escolhe aleatoriamente uma linha dentre o número total de colunas da base
    de dados
    rand_linha= random.randrange(num_linhas)
```

```
#insere o zero em determinada linha e coluna escolhidas acima
resultado[rand_linha, rand_coluna]= 0.0
```

É importante ressaltar que a inserção de omissões ou zeros, só acontecerá nas colunas da variável preditora (X), podendo ser na X_teste ou, X_treino ou ambas, dependendo do caso.

A variável alvo (y) precisa estar intacta, tanto em treino quanto em teste, para verificar se o algoritmo está acertando ou errando nas predições.

5.4 Implementação de cenários com dados faltantes

Até este momento os códigos mostrados nas seções anteriores são referentes ao Cenário 1, no qual todos os dados estão completos, tanto de treino quanto de teste. De modo a verificar se quanto mais ausências pior a qualidade da análise, foi feita uma gradação da inserção de vazios de 10% a 60% nos Cenários 2, 3 e 4. Por isso, na fase 4 da implementação, os códigos mostrados serão referentes aos Cenários 2, 3 e 4. A grande diferença entre cada cenário é o local onde serão inseridos os zeros (em X_treino, X_teste, ou ambos).

Na implementação do Cenário 2, que é focado em treinar o modelo com dados completos e testar com falta de dados, não foi necessário treinar o modelo novamente, foi utilizado o mesmo do Cenário 1. Então, o primeiro passo foi inserir as ausências nos dados de teste, que variaram de 10% a 60%, depois, têm-se as predições e a captura dos resultados a serem avaliados. Lembrando que esse mesmo processo foi repetido seis vezes, devido a variação da quantidade de omissões.

```
#laço que repete 6 vezes o código que se segue, para testar o modelo com
ausências de 10% a 60%
```

```
for i in range (1,7):
```

```
#criação da variável X_teste_C2 já inserindo as ausências
```

```
X_teste_C2=insercao_ausencias (4, X_teste, i*10)
```

```
#predição / teste do modelo usando a variável X_teste_C2
```

```
pred_C2=clf.predict(X_teste_C2)
```

Na implementação do Cenário 3, que é aquele no qual se treinou com parte da informação e testou-se com todos os dados, o primeiro passo foi inserir as ausências nos dados de treino. Em seguida, treinar o modelo, depois, testar o modelo usando todos os dados e, por fim, obter os dados para posterior comparação do resultado.

```
#laço que repete 6 vezes o código que se segue, para treinar o modelo com
    ausências de 10% a 60%
```

```
for i in range (1,7):
```

```
    #criação da variável X_treino_C3 já inserindo as ausências
```

```
    X_treino_C3= insercao_ausencias (4, X_treino, i*10)
```

```
    #treinamento do modelo de regressão logística utilização as variáveis
```

```
        X_treino_C3 e y_treino
```

```
    clf=LogisticRegression(random_state=0,solver='newton-cg',
```

```
        multi_class='multinomial').fit(X_treino_C3, y_treino)
```

```
    #predição / teste do modelo usando X_teste (que é o dado completo)
```

```
    pred_C3=clf.predict(X_teste)
```

Por fim, na implementação do último cenário, que é aquele treinado e testado com omissão de informação, primeiro inseriram-se os zeros nos dados de treino, então, treinou-se o modelo. Depois, foi a vez de inserir as ausências nos dados de teste, testar o modelo e guardar os resultados para análise.

```
#laço que repete 6 vezes o código que se segue, para treinar e testar o modelo com
    ausências de 10% a 60%
```

```
for i in range (1,7):
```

```
    #criação da variável X_treino_C4 já inserindo as ausências
```

```
    X_treino_C4= insercao_ausencias (4, X_treino, i*10)
```

```
    #treino do modelo de regressão logística usando as variáveis X_treino_C4 e
```

```
        y_treino
```



```

clf=LogisticRegression(random_state=0,solver='newton-cg',
    multi_class='multinomial').fit(X_treino_C4, y_treino)

#criação da variável X_teste_C4 já inserindo as ausências
X_teste_C4=insercao_ausencias (4, X_teste, i*10)

#predição / teste do modelo utilizando a variável X_teste_
C4 pred_C4=clf.predict(X_teste_C4)

```

5.5 Obtenção dos resultados

Por fim, na última seção, tratou-se da obtenção dos resultados da execução do algoritmo para poder avaliar todo o processo. Foi criado um *dataframe* (df_Comparativo) para armazenar as seguintes informações: Cenário (C1_xx, ..., CN_xx, sendo N o cenário e xx a porcentagem de omissões); Ausências Treino (porcentagem de zeros inseridos nos dados de treino); Ausências Teste (porcentagem de zeros inseridos nos dados de testes); Acurácia (cálculo entre de acertos e erros) e Matriz de Confusão (mostra exatamente onde houveram os erros e os acertos).

```

#criação do dataframe que vai receber os resultados da assertividade do modelo de
regressão logística em cada cenário e subcenário
df_Comparativo=pd.DataFrame([[ 'C1_00', 0, 0, acc, cm]], columns=[ 'Cenário',
    'Ausências Treino (%)', 'Ausências Teste (%)', 'Acurácia', 'Mat Confusão'])

```

Depois da criação do *dataframe*, para cada cenário foi feita a coleta dos resultados.

5.6 Considerações finais do capítulo

Esse capítulo foi voltado à explanação da implementação do algoritmo. Primeiramente, descreveu-se como foi feita a importação do *dataset* de Íris. Em seguida, como se treinou o algoritmo de regressão logística. Depois, como foi pensada e construída a função de inserção de ausências. Então, como se implementou os

cenários de inserção de faltantes. Por fim, como foram obtidos os resultados da análise para posterior verificação.

No próximo capítulo serão comentados os resultados da execução do código implementado.

6 RESULTADOS

Neste capítulo serão tratados os assuntos: i) base de dados utilizada; ii) tempo de resposta do algoritmo como um todo; iii) tempo de resposta de cada algoritmo de predição; iv) resultados relacionados a acurácia e v) comparativos.

6.1 Tempo

Para medir a duração da execução do algoritmo, usou-se a biblioteca *Time* (PYTHON ORG,) para capturar o horário de início e fim do processo. Assim, a diferença entre o final e o começo é o tempo, em milissegundos, que se levou para executar o código em sua completude.

```
#importação da biblioteca que conta o tempo
import time

#início da medição do tempo
inicio=time.time()

#final da medição
fim=time.time()

#resultado da diferença de fim e início em milissegundos
print(fim-inicio)
```

A execução do código na sua totalidade, em todas as vezes que foi executado, levou cerca de três segundos. Esse tempo é pouco expressivo, se comparado a quantidade de atividades executadas. Logo, esse não foi um problema, nem um ponto de grande relevância nesse projeto. Talvez se o *dataset* fosse maior, essa questão tomasse maiores proporções e importância no desenvolvimento da solução.

6.2 Medições

Durante o processo de desenvolvimento, foram colocados alguns gatilhos em cada etapa para que ao final fosse possível avaliar se existe correlação entre ausência e qualidade da análise. Para que essa avaliação fosse efetiva, escolheu-se duas métricas bastante populares no meio da ciência de dados, que são a Acurácia e a Matriz de Confusão, conforme discutido na seção 4.4.2. Para a Matriz de Confusão, quanto maior o valor de verdadeiros positivos, maior o número de acertos, ou seja, deseja-se que os maiores valores estejam concentrados na diagonal da matriz. Já para a Acurácia, o desejado é que seja o mais próximo possível de 1.

Para analisar a correlação ou não entre dados incompletos e a qualidade de análise, o código foi executado algumas dezenas de vezes. Contudo, selecionou-se três amostras, nas quais é possível tirar as conclusões necessárias para este trabalho. Mas, antes de mostrar os resultados, é importante detalhar o que são cada coluna das tabelas.

Cenário: os cenários estão descritos nas tabelas que se seguem no formato $C(i)_j$, sendo i o cenário, que vai de 1 a 4, e j a quantidade de zeros inseridos no *dataset*, que vai de 0 a 60. Lembrando que a explicação dos cenários pode ser encontrada na seção 4.2 e o código implementado no Anexo A.

% DA Treino: é a porcentagem de dados ausentes inseridos no treino do modelo.

% DA Teste: é a porcentagem de dados ausentes inseridos no teste do modelo.

MC: significa matriz de confusão.

Para facilitar a visualização da matriz de confusão em cada medição, dividiu-se cada cenário em uma tabela. O Cenário 1 na Tabela 6, o Cenário 2 na Tabela 7, o Cenário 3 na Tabela 8 e o Cenário 4 na Tabela 9. Já os resultados da acurácia estão todos contidos na Tabela 10.

Tabela 6 – Matriz de confusão nas três medições para o Cenário 1

Cenário	% DA Treino	% DA Teste	MC - Medição 1	MC - Medição 2	MC - Medição 3
C1_00	0	0	[14 0 0 0 11 1 0 0 12]	[13 0 0 0 11 0 0 0 14]	[10 0 0 0 16 0 0 0 12]

Fonte: Próprio autor

Tabela 7 – Matriz de confusão nas três medições para o Cenário 2

Cenário	% DA Treino	% DA Teste	MC - Medição 1	MC - Medição 2	MC - Medição 3
C2_10	0	10	[13 1 0 4 4 4 1 0 11]	[11 2 0 2 8 1 2 0 12]	[10 0 0 2 14 0 0 3 9]
C2_20	0	20	[10 4 0 5 5 2 0 0 12]	[12 1 0 3 8 0 1 0 13]	[9 1 0 2 12 2 3 3 6]
C2_30	0	30	[12 2 0 4 5 3 3 2 7]	[11 2 0 6 3 2 5 2 7]	[9 1 0 6 9 1 4 0 8]
C2_40	0	40	[13 1 0 5 4 3 6 0 6]	[12 1 0 5 4 2 3 2 9]	[10 0 0 7 9 0 3 5 4]
C2_50	0	50	[12 2 0 5 7 0 5 3 4]	[12 1 0 4 7 0 7 1 6]	[9 1 0 10 5 1 3 2 7]
C2_60	0	60	[14 0 0 6 6 0 4 2 6]	[11 2 0 7 3 1 8 2 4]	[10 0 0 8 7 1 6 1 5]

Fonte: Próprio autor

Tabela 8 – Matriz de confusão nas três medições para o Cenário 3

Cenário	% DA Treino	% DA Teste	MC - Medição 1	MC - Medição 2	MC - Medição 3
C3_10	10	0	[14 0 0 0 11 1 0 0 12]	[13 0 0 0 8 3 0 1 13]	[10 0 0 0 15 1 0 0 12]
C3_20	20	0	[14 0 0 0 3 9 0 0 12]	[13 0 0 0 1 10 0 0 14]	[10 0 0 0 5 11 0 0 12]
C3_30	30	0	[14 0 0 0 6 6 0 0 12]	[13 0 0 0 1 10 0 0 14]	[10 0 0 0 5 11 0 0 12]
C3_40	40	0	[14 0 0 0 0 12 0 0 12]	[13 0 0 0 1 10 0 0 14]	[10 0 0 0 0 16 0 0 12]
C3_50	50	0	[14 0 0 0 0 12 0 0 12]	[12 0 1 0 0 11 0 0 14]	[9 0 1 0 0 16 0 0 12]
C3_60	60	0	[14 0 0 0 0 12 0 0 12]	[8 5 0 0 1 10 0 0 14]	[10 0 0 0 0 16 0 0 12]

Fonte: Próprio autor

Tabela 9 – Matriz de confusão nas três medições para o Cenário 4

Cenário	% DA Treino	% DA Teste	MC - Medição 1	MC - Medição 2	MC - Medição 3
C4_10	10	10	[14 0 0 1 9 2 0 3 9]	[13 0 0 0 7 4 0 8 6]	[10 0 0 2 12 2 1 1 10]
C4_20	20	20	[14 0 0 2 7 3 0 6 6]	[13 0 0 1 4 6 1 4 9]	[10 0 0 1 15 0 0 2 10]
C4_30	30	30	[14 0 0 3 4 5 1 4 7]	[11 2 0 3 4 4 1 5 8]	[10 0 0 3 9 4 2 2 8]
C4_40	40	40	[13 1 0 0 8 4 1 4 7]	[10 3 0 0 8 3 3 1 10]	[8 2 0 3 5 8 0 2 10]
C4_50	50	50	[13 1 0 1 7 4 1 3 8]	[12 1 0 1 5 5 2 8 4]	[10 0 0 3 9 4 2 0 10]
C4_60	60	60	[14 0 0 2 6 4 2 1 9]	[9 4 0 3 4 4 3 6 5]	[10 0 0 2 8 6 3 2 7]

Fonte: Próprio autor

Tabela 10 – Acurácia nas três medições para cada cenário

Cenário	% DA	% DA	Acurácia -	Acurácia -	Acurácia -
C1_00	0	0	0,97368421	1	1
C2_10	0	10	0,73684211	0,81578947	0,86842105
C2_20	0	20	0,71052632	0,86842105	0,71052632
C2_30	0	30	0,63157895	0,55263158	0,68421053
C2_40	0	40	0,60526316	0,65789474	0,60526316
C2_50	0	50	0,60526316	0,65789474	0,55263158
C2_60	0	60	0,68421053	0,47368421	0,57894737
C3_10	10	0	0,97368421	0,89473684	0,97368421
C3_20	20	0	0,76315789	0,73684211	0,71052632
C3_30	30	0	0,84210526	0,73684211	0,71052632
C3_40	40	0	0,68421053	0,73684211	0,57894737
C3_50	50	0	0,68421053	0,68421053	0,55263158
C3_60	60	0	0,68421053	0,60526316	0,57894737
C4_10	10	10	0,84210526	0,68421053	0,84210526
C4_20	20	20	0,71052632	0,68421053	0,92105263
C4_30	30	30	0,65789474	0,60526316	0,71052632
C4_40	40	40	0,73684211	0,73684211	0,60526316
C4_50	50	50	0,73684211	0,55263158	0,76315789
C4_60	60	60	0,76315789	0,47368421	0,65789474

Fonte: Próprio autor

6.3 Análise dos resultados

A partir da análise das medições apresentadas anteriormente é possível notar detalhes interessantes que, em seguida, ajudarão a responder o objetivo deste estudo. Lembrando que o cenário ideal para a Matriz de Confusão é quando ela se torna uma matriz diagonal e para a Acurácia, é quanto mais próximo de 1 a acurácia, mais assertiva a análise.

Em teoria, o cenário que teria a maior acurácia em todas as medições, deveria ser o C1_00, pois, é o único completo para treino e teste. Todavia, é possível observar na Medição 1, comparando os resultados de C1_00 com C3_10 são similares, tanto na

tabela de Acurácia quanto na de Matriz de Confusão, ou seja, o cenário completo não é o melhor, contudo, está entre os melhores.

Verificando-se a Matriz de Confusão, pode-se observar situações na qual um cenário com maior quantidade de faltantes tem maior ou igual número de acertos do que um cenário com menos ausências. Exemplos: Medição 1 - C3_40, C3_50 e C3_60 tem a mesma assertividade; Medição 3 - C3_60 tem mais acertos comparado com C3_50; entre outros.

Assim como no item anterior, pode-se observar situações na qual um cenário pode ter melhor Acurácia quando existem mais ausências do que o contrário. Exemplo disso, são: Medição 1 - caso C2_60 comparado com C2_50, C2_40 e C2_30; Medição 2 - C2_20 comparado com C2_10; Medição 3 - C4_50 comparado com C4_40; entre outros.

Agora, comparando-se o cenário completo, C1_00, Acurácia - Medição 2, resultado 1, com um cenário de muitas ausências como C4_60, Acurácia - Medição 2, resultado 0,4736842105, é possível notar uma piora de mais de 50% na qualidade.

6.4 Considerações finais do capítulo

Neste capítulo, tratou-se de mostrar os resultados e analisar os mesmos. O primeiro item avaliado foi o tempo, então, mostrou-se a implementação do código que contabiliza o tempo de execução do algoritmo e então, avaliou-se esse item. Em seguida, foram apresentados os resultados da medição da matriz de confusão e a acurácia em cada cenário/subcenário. E por fim, foi feita análise desses resultados.

No capítulo que se segue, será discutido um pouco mais sobre os resultados obtidos, se eles estão aderentes ao que foi proposto.

7 CONCLUSÃO

Nos capítulos iniciais deste trabalho, apresentou-se o que são dados ausentes, o porquê acontece, o que são e quais as melhores maneiras de tratá-los, para que o leitor compreenda os aspectos básicos e relevantes do tema. Em seguida, foi levantada a pergunta que este projeto de mestrado quer responder que é: “Como uma análise de dados, feita utilizando um algoritmo preditivo, é impactada pelos dados ausentes?”. Então, foi codificada uma versão do algoritmo *Logistic Regression*, para essa situação. Na qual foram implementados alguns cenários, que consistiu em aumentar progressivamente a omissão em dados de treino e de teste. Após codificar toda essa lógica, foram usados os dados da Flor de Íris, que é amplamente reconhecida no meio acadêmico-científico, para testar a solução e servir como demonstrativo da prova de conceito. Depois de executar o algoritmo, foram obtidos os resultados da Acurácia e da Matriz de Confusão da regressão em cada cenário.

A partir da análise dos resultados feitos na seção 6.3, conclui-se que o aumento das omissões diminui a qualidade do resultado de uma análise preditiva de uma maneira geral, comparando-se por exemplo a Acurácia. Contudo, não é em todas as situações que o acréscimo de incompletudes piorou o resultado. Logo, é difícil de afirmar que em todos os cenários, quanto maior o número de ausências pior é a qualidade da análise. É importante salientar que essa conclusão é baseada na utilização de um algoritmo, base de dados e cenários específicos.

7.1 Trabalhos futuros

Uma vez que este estudo teve um escopo bem definido, como possibilidade de trabalhos futuros possíveis destacam-se possibilidades de ampliação do escopo como:

- Foram testados quatro cenários, que variaram de 10% a 60% de ausência, então, uma hipótese é fazer uma alteração desses cenários e subcenários. Podendo-se ir alternando ausência em treino e teste, não só em um ou em outro ou nos dois ao mesmo tempo, como foi feito. Pode-se ainda, aumentar a quantidade de faltantes chegando a 100% e ver o que acontece.
- O algoritmo implementado foi o *Logistic Regression*, pode-se testar outro algoritmo de predição.

- Nesse trabalho a implementação do algoritmo foi focada somente em ausências randômicas, outra possibilidade é testar com outros tipos.
- A base de dados utilizada é do tipo quantitativa, por isso, uma sugestão seria utilizar uma base de dados qualitativa.
- A lógica criada na função de inserção de ausências, pode ter impactado os resultados, então mudar a lógica para inserção de ausências pode mudar o resultado.
- Neste trabalho, a base de dados utilizada continha somente 150 registros, logo, seria interessante testar se com uma base de dados maior ainda seriam registrados resultado e comportamento similares aos vistos aqui.
- Apesar de notar que o aumento de incompletudes, em algumas situações, melhora o resultado da análise, não se sabe o motivo. Por isso, um trabalho interessante seria investigar o porquê deste fenômeno.

7.2 Contribuições desse trabalho

Neste trabalho, as incompletudes foram estudadas em seus mais variados aspectos, como se caracterizam; por que acontecem; quais são as possibilidades de tratamento e as melhores práticas para cada tipo de situação. Com isso, mostrou-se que a omissão pode ser informativa e que a exclusão é uma das alternativas, mas não a única. Dessa maneira, popularizando as melhores práticas para o tratamento de dados e facilitando o dia a dia de profissionais que trabalham com grande variedade de dados.

Adicionalmente, verificou-se que a afirmação “Quanto maior a quantidade de ausências, pior a qualidade da análise final” é uma inverdade. O que é importante para desmistificar esse pré-conceito, sobre a ausência de dados.

Com todas essas informações em mãos, é possível dizer que este trabalho correspondeu ao seu objetivo inicial e assim, pode-se afirmar que nem sempre o aumento do número de dados inexistente impacta negativamente no resultado final da análise.

REFERÊNCIAS

- ACCURACY. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning and Data Mining. Boston, MA: Springer, 2017. Disponível em: https://doi.org/10.1007/978-1-48997687-1_3. Acesso em: 08/11/2020.
- AWAD, M.; KHANNA, R. Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers. [S.l.]: Apress, 2015. ISBN 978-1-4302-5989-3e-ISBN 978-1-4302-5990-9.
- AWS. DeepAR Forecasting Algorithm. Disponível em: <https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html>. Acesso em: 01/03/2021.
- BATINI, C.; SCANNAPIECO, M. Data and Information Quality: Dimensions, Principles and Techniques. Switzerland: Springer International Publishing, 2016.
- BOCK, T. What are the Different Types of Missing Data? 2019. Disponível em: <https://www.displayr.com/different-types-of-missing-data/>. Acesso em: 22/02/2019.
- BROWNE, Michael W. Cross-Validation Methods. Journal of Mathematical Psychology, [s. l.], v. 44, ed. 1, p. 108-132, 2000. DOI <https://doi.org/10.1006/jmps.1999.1279>. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0022249699912798>. Acesso em: 14/07/2021.
- CAETANO, J. A. Por que estudar preparação de dados? 2018. Disponível em: <http://igti.com.br/blog/por-que-estudar-preparacao-de-dados/>. Acesso em: 22/02/2019.
- CAMARGOS, V. P. et al. Imputação múltipla e análise de casos completos em modelos de regressão logística: uma avaliação prática do impacto das perdas em covariáveis. Cadernos de Saúde Pública, v. 27, n. 12, p. 2299 – 2313, Dezembro 2011.
- FERREIRA, J. Data preparation: alicerce dos projetos de Big Data. 2017. Disponível em: <http://www.bigdatabusiness.com.br/data-preparation/>. Acesso em: 22/02/2019.
- FIGUEIRA, C. V. Modelos de Regressão Logística. 2006. 149 p. Dissertação (Mestrado em Matemática) — Universidade Federal do Rio Grande do Sul.
- GARSON, G. D. Missing Values Analysis and Data Imputation. Asheboro, NC: Statistical Associates Publishers, 2015. Disponível em: <http://www.statisticalassociates.com/missingvaluesanalysis.htm>.

GOLDSCHMIDT, R. Data mining: um guia Prático. [S.l.]: Gulf Professional, 2005. ISBN 8535218777, 9788535218770.

GOOGLE. Python Data Science Handbook. 2020. Disponível em: <https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb>. Acesso em: 03/08/2020.

HAIR JR., J. F. et al. Multivariate Data Analysis. 7. ed. [S.l.]: Pearson Education Limited, 2014.

IBM. IBM SPSS Missing Values 24. 2019a. Disponível em: http://public.dhe.ibm.com/software/analytics/spss/documentation/statistics/24.0/ptBR/client/Manuals/IBM_SPSS_Missing_Values.pdf. Acesso em: 22/02/2019.

IBM. Pairwise vs. Listwise deletion: What are they and when should I use them? 2019b. Disponível em: <http://www-01.ibm.com/support/docview.wss?uid=swg21475199>. Acesso em: 22 fev. 2019.

KANG, H. The prevention and handling of the missing data. Korean Journal of Anesthesiology, v. 64, n. 5, p. 402 – 404, May 2013. Disponível em: <https://doi.org/10.4097/kjae.2013.64.5.402>.

LAURETTO, M. de S. Análise Exploratória de Dados. 2011. Disponível em: http://www.each.usp.br/lauretto/SIN5008_2011/aula01/aula1.pdf. Acesso em: 22/02/2019.

LEE, K. Inteligência artificial: Como os robôs estão mudando o mundo, a forma como amamos, nos relacionamos, trabalhamos e vivemos. 1. ed. [S.l.]: Globo Livros, 2019. 292 p.

LITTLE, R. J. A. A test of missing completely at random for multivariate data with missing values. Journal of the American Statistical Association, v. 83, n. 44, p. 1198 – 1202, 1988. Acesso em: 22/02/2019.

LITTLE, T. D. et al. Missing Data. Developmental psychopathology, v. 1, p. 1 – 37, 2016.

LOPES, L. dos S. Definição de um modelo de cobrança (Collection Score) utilizando Regressão Logística Multinomial. 2004. 58 p. Monografia (Bacharelado em Estatística) — Universidade Federal do Rio Grande do Sul.

MARREIROS, G.; OLIVEIRA, P. Preparação de dados. 2000. Disponível em: <https://paginas.fe.up.pt/~mgi99021/it/pdados.htm>. Acesso em: 22/02/2019.

MOHAN, K. Graphical Models for Inference with Missing Data. 2017. Tese (Doutorado) — University of California. Disponível em: <https://escholarship.org/uc/item/6mk2b174>.

NEEDHAM, T. C. Python: For Beginners A Crash Course Guide To Learn Python in 1 Week. [S.l.]: Independently published, 2007. ISBN 1549776673.

OLIPHANT, T. E. Python for Scientific Computing. Computing in Science & Engineering, v. 9, n. 3, p. 10 – 20, MAY/JUNE 2007.

OSBORNE, J. W. Dealing with missing or incomplete data: debunking the myth of emptiness. In: OSBORNE, J. W. (ed.). Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data. Thousand Oaks, CA: SAGE Publications Ltd., 2013. cap. 6, p. 105 – 138.

PANDAS: Python Data Analysis Library. Disponível em: <https://pandas.pydata.org/>. Acesso em: 24/10/2019.

PERELMUTER, G. Um minuto na Internet. 2018. Disponível em: <https://economia.estadao.com.br/noticias/geral,um-minuto-na-internet,70002283969>. Acesso em: 22/02/2019.

PRATI, R. C. Preparação dos Dados. 2019. Disponível em: <http://professor.ufabc.edu.br/~ronaldo.prati/DataMining/Preprocessamento.pdf>. Acesso em: 22/02/2019.

PYTHON ORG. time — Time access and conversions. Disponível em: <https://docs.python.org/3/library/time.html>. Acesso em: 30/09/2020.

RESEARCHERS, Y. Yandex. Disponível em: <https://research.yandex.com/>. Acesso em: 24/10/2020.

RIOUX, C.; LITTLE, T. D. Missing data treatments in intervention studies: What was, what is, and what should be. International Journal of Behavioral Development, v. 45, n. 1, p. 51 – 58, October 2019.

SANCHES, A. R. Disciplina: Fundamentos de Armazenamento e Manipulação de Dados. 2005. Disponível em: <https://www.ime.usp.br/~andrers/aulas/bd2005-1/aula7.html>. Acesso em: 08/11/2020.

SC.D., R. A. F.; F.R.S. THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. *Annals of Eugenics*, v. 7, n. 2, p. 179 – 188, September 1936. Disponível em: <https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1469-1809.1936.tb02137.x>. Acesso em: 02/11/2020.

SCIKIT LEARN. `sklearn.linear_model.LogisticRegression`. Disponível em: https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Acesso em: 04/08/2020.

SCIKIT LEARN. The Iris Dataset. Disponível em: https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html. Acesso em: 11/08/2020.

SHIMAKURA, S. E. Tipos de variáveis. 2012. Disponível em: <http://leg.ufpr.br/~silvia/CE055/node8.html>. Acesso em: 22/02/2019.

SILVA, J. de A. Substituição de valores ausentes: uma abordagem baseada em um algoritmo evolutivo para agrupamento de dados. 2010. 150 p. Dissertação (Mestrado) — Universidade de São Paulo.

STATISTICS SOLUTIONS. Missing Data: Listwise vs. Pairwise. 2019. Disponível em: <https://www.statisticssolutions.com/missing-data-listwise-vs-pairwise/>. Acesso em: 22/02/2019.

SUPERVISED Learning. In: Sammut C., Webb G.I. (eds) *Encyclopedia of Machine Learning*. Boston, MA: Springer, 2011. Disponível em: https://doi.org/10.1007/978-0-387-30164-8_803. Acesso em: 08/11/2020.

TANG, F.; ISHWARAN, H. Random forest missing data algorithms. *Statistical Analysis and Data Mining*, v. 10, n. 6, p. 363 – 377, June 2017.

TING, K. M. Confusion Matrix. In: Sammut C., Webb G.I. (eds) *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer, 2017. Disponível em: https://doi.org/10.1007/978-1-4899-7687-1_50. Acesso em: 08/11/2020.

VINHA, L. G. do A.; LAROS, J. A. Dados ausentes em avaliações educacionais: comparação de métodos de tratamento. Estudos em Avaliação Educacional, São Paulo, v. 29, n. 70, p. 156 – 187, Janeiro 2018. ISSN 0103-6831. Disponível em: <http://publicacoes.fcc.org.br/ojs/index.php/eae/article/view/3916>. Acesso em: 01/04/2019.

ANEXOS

ANEXO A – CONJUNTO DE DADOS DA FLOR DE ÍRIS

Figura 5 - Anotações de Flor de Íris

<i>Iris setosa</i>				<i>Iris versicolor</i>				<i>Iris virginica</i>			
Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8
4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8
4.9	3.1	1.5	0.1	5.2	2.7	3.9	1.4	7.2	3.6	6.1	2.5
5.4	3.7	1.5	0.2	5.0	2.0	3.5	1.0	6.5	3.2	5.1	2.0
4.8	3.4	1.6	0.2	5.9	3.0	4.2	1.5	6.4	2.7	5.3	1.9
4.8	3.0	1.4	0.1	6.0	2.2	4.0	1.0	6.8	3.0	5.5	2.1
4.3	3.0	1.1	0.1	6.1	2.9	4.7	1.4	5.7	2.5	5.0	2.0
5.8	4.0	1.2	0.2	5.6	2.9	3.6	1.3	5.8	2.8	5.1	2.4
5.7	4.4	1.5	0.4	6.7	3.1	4.4	1.4	6.4	3.2	5.3	2.3
5.4	3.9	1.3	0.4	5.6	3.0	4.5	1.5	6.5	3.0	5.5	1.8
5.1	3.5	1.4	0.3	5.8	2.7	4.1	1.0	7.7	3.8	6.7	2.2
5.7	3.8	1.7	0.3	6.2	2.2	4.5	1.5	7.7	2.6	6.9	2.3
5.1	3.8	1.5	0.3	5.6	2.5	3.9	1.1	6.0	2.2	5.0	1.5
5.4	3.4	1.7	0.2	5.9	3.2	4.8	1.8	6.9	3.2	5.7	2.3
5.1	3.7	1.5	0.4	6.1	2.8	4.0	1.3	5.6	2.8	4.9	2.0
4.6	3.6	1.0	0.2	6.3	2.5	4.9	1.5	7.7	2.8	6.7	2.0
5.1	3.3	1.7	0.5	6.1	2.8	4.7	1.2	6.3	2.7	4.9	1.8
4.8	3.4	1.9	0.2	6.4	2.9	4.3	1.3	6.7	3.3	5.7	2.1
5.0	3.0	1.6	0.2	6.6	3.0	4.4	1.4	7.2	3.2	6.0	1.8
5.0	3.4	1.6	0.4	6.8	2.8	4.8	1.4	6.2	2.8	4.8	1.8
5.2	3.5	1.5	0.2	6.7	3.0	5.0	1.7	6.1	3.0	4.9	1.8
5.2	3.4	1.4	0.2	6.0	2.9	4.5	1.5	6.4	2.8	5.6	2.1
4.7	3.2	1.6	0.2	5.7	2.6	3.5	1.0	7.2	3.0	5.8	1.6
4.8	3.1	1.6	0.2	5.5	2.4	3.8	1.1	7.4	2.8	6.1	1.9
5.4	3.4	1.5	0.4	5.5	2.4	3.7	1.0	7.9	3.8	6.4	2.0
5.2	4.1	1.5	0.1	5.8	2.7	3.9	1.2	6.4	2.8	5.6	2.2
5.5	4.2	1.4	0.2	6.0	2.7	5.1	1.6	6.3	2.8	5.1	1.5
4.9	3.1	1.5	0.2	5.4	3.0	4.5	1.5	6.1	2.6	5.6	1.4
5.0	3.2	1.2	0.2	6.0	3.4	4.5	1.6	7.7	3.0	6.1	2.3
5.5	3.5	1.3	0.2	6.7	3.1	4.7	1.5	6.3	3.4	5.6	2.4
4.9	3.6	1.4	0.1	6.3	2.3	4.4	1.3	6.4	3.1	5.5	1.8
4.4	3.0	1.3	0.2	5.6	3.0	4.1	1.3	6.0	3.0	4.8	1.8
5.1	3.4	1.5	0.2	5.5	2.5	4.0	1.3	6.9	3.1	5.4	2.1
5.0	3.5	1.3	0.3	5.5	2.6	4.4	1.2	6.7	3.1	5.6	2.4
4.5	2.3	1.3	0.3	6.1	3.0	4.6	1.4	6.9	3.1	5.1	2.3
4.4	3.2	1.3	0.2	5.8	2.6	4.0	1.2	5.8	2.7	5.1	1.9
5.0	3.5	1.6	0.6	5.0	2.3	3.3	1.0	6.8	3.2	5.9	2.3
5.1	3.8	1.9	0.4	5.6	2.7	4.2	1.3	6.7	3.3	5.7	2.5
4.8	3.0	1.4	0.3	5.7	3.0	4.2	1.2	6.7	3.0	5.2	2.3
5.1	3.8	1.6	0.2	5.7	2.9	4.2	1.3	6.3	2.5	5.0	1.9
4.6	3.2	1.4	0.2	6.2	2.9	4.3	1.3	6.5	3.0	5.2	2.0
5.3	3.7	1.5	0.2	5.1	2.5	3.0	1.1	6.2	3.4	5.4	2.3
5.0	3.3	1.4	0.2	5.7	2.8	4.1	1.3	5.9	3.0	5.1	1.8

Fonte: Retirada do artigo THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

ANEXO B – IMPLEMENTAÇÃO

Observação: A implementação foi feita no Google Colab e a exportação dos códigos e resultados seguem a dinâmica dessa ferramenta. Por isso, a formatação está fora do padrão esperado.

▼ Teste de correlação entre ausência e qualidade do dado

A ideia desse teste é mostrar como um algoritmo de predição se comporta quando submetido à omissões de formas diversas. São 4 cenários diferentes entre si, desde treino e teste utilizando dados completos à treino e teste com dado omissos. É importante ressaltar que os dados `y_treino` e `y_teste` não sofrerão alteração, pois são importantes para calcular a assertividade do modelo.

```
import time ### início da medição do tempo
inicio=time.time()
```

▼ Cenário 1: modelo é treinado e submetido somente a dados completos.

Esse é a nossa base de comparação para provar que quanto mais omissões, pior é a qualidade do resultado final da análise.

```
### Import da base de IRIS
from sklearn.datasets import load_iris

##carrega a base já separada em:
## X (colunas 'sepal length', 'sepal width', 'petal length', 'petal width')
## y (resultados 0-Setosa, 1-Versicolour ou 2-Virginica)
X, y = load_iris(return_X_y=True)
print(y)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

```
### separando a base
from sklearn.model_selection import train_test_split

#separa dados em 75% para treino e 25% para teste
X_treino, X_teste, y_treino, y_teste = train_test_split(X, y)
```

```
##coloca em data um dataset X e y para facilitar o processamento
import pandas as pd

dfX=pd.DataFrame(X_treino, columns= ['sepal length', 'sepal width',
                                     'petal length', 'petal width'])
dfy=pd.DataFrame(y_treino, columns=['resultados'])
```

```
print(dfX.head())
```

```
      sepal length  sepal width  petal length  petal width
0              5.5           4.2           1.4           0.2
```

1	5.0	3.4	1.5	0.2
2	5.1	3.7	1.5	0.4
3	6.0	2.2	5.0	1.5
4	5.8	4.0	1.2	0.2

```
### analisando a base antes de qualquer alteração
print ('Resumo dos dados - X')
print (dfX.describe() )
```

```
Resumo dos dados - X
```

	sepal length	sepal width	petal length	petal width
count	112.000000	112.000000	112.000000	112.000000
mean	5.854464	3.099107	3.718750	1.188393
std	0.853070	0.437745	1.823744	0.775613
min	4.300000	2.200000	1.000000	0.100000
25%	5.100000	2.800000	1.500000	0.300000
50%	5.800000	3.000000	4.400000	1.350000
75%	6.425000	3.400000	5.100000	1.800000
max	7.900000	4.200000	6.900000	2.500000

```
print ('Resumo dos dados - y')
print (dfy.describe())
```

```
Resumo dos dados - y
```

	resultados
count	112.000000
mean	0.982143
std	0.838082
min	0.000000
25%	0.000000
50%	1.000000
75%	2.000000
max	2.000000

```
import pylab as pl

#visualizando as colunas
dfX.hist()
pl.show()
```

```
dfy.hist()  
pl.show()
```

```
dfX.boxplot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff30b58aac8>

```
dfy.boxplot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff30b4cc358>

```

### treinando o modelo
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0, solver='newton-cg',
                        multi_class='multinomial').fit(X_treino, y_treino)

### testando o modelo
pred=clf.predict(X_teste)

#### matriz de confusão
from sklearn.metrics import confusion_matrix

cm=confusion_matrix( y_teste, pred)
print(cm)

### acurácia
from sklearn.metrics import accuracy_score
acc=accuracy_score(y_teste, pred)

```

```

[[10  0  0]
 [ 0 16  0]
 [ 0  0 12]]

```

▼ *Dataset* para armazenar os resultados de cada cenário

```

df_Comparativo=pd.DataFrame(['C1_00', 0, 0, acc, cm],
                             columns=['Cenário', 'Ausências Treino (%)',
                                       'Ausências Teste (%)', 'Acurácia', 'Mat Confusão'])
print(df_Comparativo)

```

	Cenário	Ausências Treino (%)	...	Acurácia	Mat Confusão
0	C1_00	0	...	1.0	[[10, 0, 0], [0, 16, 0], [0, 0, 12]]

[1 rows x 5 columns]

▼ Função para inserir ausências aleatoriamente em um *dataset*

```

def insercao_ausencias (num_colunas, serie, porcentagem):
    import random

    num_linhas= len(serie) ## para saber qual o total de linhas do dataset

    ## calculando o número de ausências que serão inseridas
    calcula= ((num_linhas*num_colunas)*porcentagem)/100
    resultado= serie.copy()

    for i in range(int(calcula)):
        rand_coluna= random.randrange(num_colunas)
        rand_linha= random.randrange(num_linhas)
        resultado[rand_linha, rand_coluna]= 0.0 ## para marcar a ausência

    return resultado

```

▼ Cenário 2: modelo é treinado com dados completos, mas o algoritmo é depois submetido a ausências.

O que significa que somente alteraremos os dados de teste (X_{teste}). Se alterarmos y_{teste} depois será impossível comparar a taxa de acerto, por isso, continuará como está.

Primeiro serão inseridas 10%, depois 20%, até 60% de ausências e analisando a acurácia.

```
for i in range(1,7):

    X_teste_C2=insercao_ausencias ( 4, X_teste, i*10) ## inserindo ausências
    pred_C2=clf.predict(X_teste_C2) ## fazendo a predição
    acc_C2=accuracy_score(y_teste, pred_C2) ## acurácia
    cm_C2=confusion_matrix( y_teste, pred_C2) ## matriz de confusão
    df_aux= pd.DataFrame([[ 'C2_'+str(i*10), 0, i*10, acc_C2, cm_C2]],
                          columns=[ 'Cenário', 'Ausências Treino (%)',
                                    'Ausências Teste (%)', 'Acurácia', 'Mat Confusão'])
    df_Comparativo=df_Comparativo.append(df_aux)

    print('C2_'+str(i*10))

C2_10
C2_20
C2_30
C2_40
C2_50
C2_60
```

▼ Cenário 3: modelo é treinado com ausentes e submetidos a completos.

O que significa que somente alteraremos os dados de treino (X_{treino}) e então precisaremos retreinar o modelo.

```
for i in range(1,7):

    X_treino_C3= insercao_ausencias ( 4, X_treino, i*10) ## inserindo ausências
    clf = LogisticRegression(random_state=0, solver='newton-cg',
                             multi_class='multinomial').fit(X_treino_C3, y_treino)

    pred_C3=clf.predict(X_teste) ## fazendo a predição
    acc_C3=accuracy_score(y_teste, pred_C3) ## acurácia
    cm_C3=confusion_matrix( y_teste, pred_C3) ## matriz de confusão
    df_aux= pd.DataFrame([[ 'C3_'+str(i*10), i*10, 0, acc_C3, cm_C3]],
                          columns=[ 'Cenário', 'Ausências Treino (%)',
                                    'Ausências Teste (%)', 'Acurácia', 'Mat Confusão'])
    df_Comparativo=df_Comparativo.append(df_aux)

    print('C3_'+str(i*10))
```


C3_10
C3_20
C3_30
C3_40
C3_50
C3_60

▼ Cenário 4: modelo é treinado e submetido a dados ausentes.

O que significa que alteraremos os dados de treino e de teste (X_treino e X_teste).

```
for i in range(1,7):

    X_treino_C4= insercao_ausencias ( 4, X_treino, i*10) ## inserindo ausências em treino
    clf = LogisticRegression(random_state=0, solver='newton-cg',
                             multi_class='multinomial').fit(X_treino_C4, y_treino)

    X_teste_C4=insercao_ausencias ( 4, X_teste, i*10) ## inserindo ausências em teste
    pred_C4=clf.predict(X_teste_C4)

    acc_C4=accuracy_score(y_teste, pred_C4) ## acurácia
    cm_C4=confusion_matrix( y_teste, pred_C4) ## matriz de confusão
    df_aux= pd.DataFrame(['C4_'+str(i*10), i*10, i*10, acc_C4, cm_C4],
                          columns=['Cenário', 'Ausências Treino (%)',
                                   'Ausências Teste (%)', 'Acurácia', 'Mat Confusão'])
    df_Comparativo=df_Comparativo.append(df_aux)

    print('C4_'+str(i*10))
```

C4_10
C4_20
C4_30
C4_40
C4_50
C4_60

```
print(df_Comparativo)
```

	Cenário	Ausências Treino (%)	...	Acurácia	Mat Confusão
0	C1_00	0	...	1.000000	[[10, 0, 0], [0, 16, 0], [0, 0, 12]]
0	C2_10	0	...	0.868421	[[10, 0, 0], [2, 14, 0], [0, 3, 9]]
0	C2_20	0	...	0.710526	[[9, 1, 0], [2, 12, 2], [3, 3, 6]]
0	C2_30	0	...	0.684211	[[9, 1, 0], [6, 9, 1], [4, 0, 8]]
0	C2_40	0	...	0.605263	[[10, 0, 0], [7, 9, 0], [3, 5, 4]]
0	C2_50	0	...	0.552632	[[9, 1, 0], [10, 5, 1], [3, 2, 7]]
0	C2_60	0	...	0.578947	[[10, 0, 0], [8, 7, 1], [6, 1, 5]]
0	C3_10	10	...	0.973684	[[10, 0, 0], [0, 15, 1], [0, 0, 12]]
0	C3_20	20	...	0.710526	[[10, 0, 0], [0, 5, 11], [0, 0, 12]]
0	C3_30	30	...	0.710526	[[10, 0, 0], [0, 5, 11], [0, 0, 12]]
0	C3_40	40	...	0.578947	[[10, 0, 0], [0, 0, 16], [0, 0, 12]]
0	C3_50	50	...	0.552632	[[9, 0, 1], [0, 0, 16], [0, 0, 12]]
0	C3_60	60	...	0.578947	[[10, 0, 0], [0, 0, 16], [0, 0, 12]]
0	C4_10	10	...	0.842105	[[10, 0, 0], [2, 12, 2], [1, 1, 10]]
0	C4_20	20	...	0.921053	[[10, 0, 0], [1, 15, 0], [0, 2, 10]]

0	C4_30	30	...	0.710526	[[10, 0, 0], [3, 9, 4], [2, 2, 8]]
0	C4_40	40	...	0.605263	[[8, 2, 0], [3, 5, 8], [0, 2, 10]]
0	C4_50	50	...	0.763158	[[10, 0, 0], [3, 9, 4], [2, 0, 10]]
0	C4_60	60	...	0.657895	[[10, 0, 0], [2, 8, 6], [3, 2, 7]]

[19 rows x 5 columns]

```
fim=time.time() ## final da análise
print(fim-inicio) ### resultado em milisegundos
```

3.586163282394409

```
## enviando os resultados para um excel
from google.colab import drive
drive.mount('/content/drive')
root = "/content/drive/My Drive/Colab Notebooks/Mestrado/"

df_Comparativo.to_excel(root+'resultado.xlsx')
```

Mounted at /content/drive