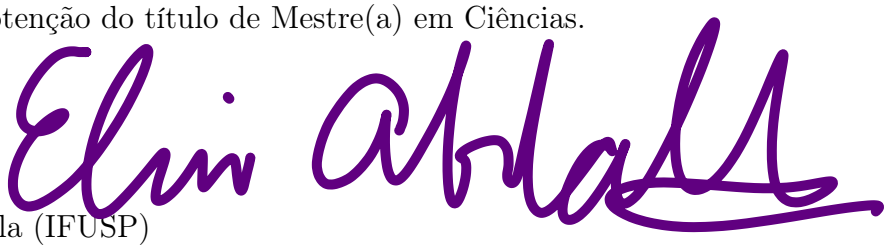
Universidade de São Paulo
Instituto de Física

Análise de dados fotométricos obtidos através do aprendizado de máquina e da K-d Tree

Amanda Farias dos Santos

Orientador: Prof. Dr. Elcio Abdalla

Dissertação de mestrado apresentada ao Instituto de Física da Universidade de São Paulo, como requisito parcial para a obtenção do título de Mestre(a) em Ciências.



Banca Examinadora:
Prof. Dr. Élcio Abdalla (IFUSP)
Profa. Dra. Claudia Lucia Mendes de Oliveira (IAG - USP)
Prof. Dr. Carlos Alexandre Wuensche de Souza (INPE)

São Paulo
2023

FICHA CATALOGRÁFICA
Preparada pelo Serviço de Biblioteca e Informação
do Instituto de Física da Universidade de São Paulo

Santos, Amanda Farias dos

Análise de dados fotométricos obtidos através do aprendizado de máquina e da K-D TREE. São Paulo, 2023.

Dissertação (Mestrado) - Universidade de São Paulo. Instituto de Física. Depto. de Física Geral

Orientador: Prof. Dr. Élcio Abdalla

Área de Concentração: Física Geral

Unitermos: 1. Análise de dados; 2. Astronomia; 3. Aprendizado de máquina; 4. Galáxias; 5. Fotometria.

USP/IF/SBI-016/2023

University of São Paulo
Physics Institute

Analysis of photometric data obtained through machine learning and K-d Tree

Amanda Farias dos Santos

Supervisor: Prof. Dr. Élcio Abdalla

Dissertation submitted to the Physics Institute of the University of São Paulo in partial fulfillment of the requirements for the degree of Master of Science.

Examining Committee:
Prof. Dr. Élcio Abdalla (IFUSP)
Prof. Dr. Claudia Lucia Mendes de Oliveira (IAG - USP)
Prof. Dr. Carlos Alexandre Wuensche de Souza (INPE)

São Paulo
2023

Agradecimentos

Ao meu orientador Élcio Abdalla pelos ensinamentos e pela orientação. Obrigada por propiciar um ambiente de aprendizado e pesquisa científica de alto nível, além dos conselhos para que eu pudesse entregar o melhor trabalho possível.

Ao Filipe Abdalla por coordenar o meu projeto, me ajudando a buscar conhecimento e me ensinando sobre astrofísica. Quando eu comecei o mestrado não sabia muito desta área, obrigada pela paciência de me ensinar de modo que eu pudesse escrever este trabalho.

À Karin Fornazier por ser uma luz para clarear a minha mente. Obrigada por todas as conversas e por todo o ensinamento.

Ao grupo de dados óticos pelas trocas no grupo de pesquisa.

À minha família por ter me encorajado a sempre buscar conhecimento. Obrigada por todo o apoio emocional, financeiro e logístico que permitiu que eu concluísse o mestrado. Obrigada ao meu padrasto, Marcus Pacce, pela torcida e pelo apoio logístico. E obrigada em especial aos meus pais, Claudia Valéria e Luiz Almério, vocês são responsáveis por todas as minhas conquistas e eu sou eternamente grata por tudo que vocês me proporcionaram.

Ao Jefferson Ferraz, meu orientador da iniciação científica, por colocar uma pulga de curiosidade para eu tentar compreender o mundo à nossa volta. Obrigada pelas palavras e incentivos para continuar no caminho da pesquisa científica.

Aos meus amigos, por estarem sempre ao meu lado me incentivando e apoiando durante todo o meu mestrado. Obrigada ao meu namorado, Daniel Rabaça, por todo suporte emocional para que eu pudesse terminar o mestrado, além do apoio logístico. Às minhas amigas de colégio, Ana Carolina Lopes, Giulia Nocito, Júlia Izquierdo e Paola Andrade, por me alegrarem quando eu precisava e tornar o mestrado algo mais fácil de completar.

Aos meus colegas do grupo BINGO, obrigada por todas as conversas e trocas científicas que tivemos na USP, um agradecimento especial aos pós graduandos Gabriel Hoerning, Eveling Costa, Alessandro Marins, Lucas Formigari, Jordany Vieira, Pablo Motta e João Alberto.

Obrigada.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Nesta dissertação utilizamos três modelos de aprendizados de máquina (*machine learning*) para o cálculo do redshift fotométrico de galáxias do catálogo astronômico do Dark Energy Survey (`DES`). Dois destes modelos são códigos públicos e nós construímos o terceiro modelo a partir de uma interface de programação de *deep learning* chamada `keras`. Para o treinamento dos modelos de *machine learning*, foram utilizadas as informações espectroscópicas das galáxias do catálogo astronômico do VIMOS Public Extragalactic Redshift Survey (`VIPERS`).

Com o intuito de verificar a acurácia dos redshifts fotométricos, foi desenvolvida uma estrutura de dados chamada `K-d Tree` que separa as galáxias em subconjuntos de acordo com os seus dados fotométricos. Para cada subconjunto foi criado um outro aprendizado de máquina que calcula o quão preciso é o valor do redshift fotométrico calculado pelos três modelos para cada galáxia. Através deste resultado, foi possível excluir galáxias cujo redshift fotométrico está longe do valor do redshift espectroscópico.

Palavras-chave: Redshift fotométrico, aprendizado de máquina, K-d Tree.

Abstract

In this thesis, we use three machine learning models to evaluate the photometric redshift of galaxies from the Dark Energy Survey (`DES`). Two of these models are open source and we built the third model using a deep learning programming interface called `keras`. In order to train the machine learning models, we used the spectroscopic information of galaxies from the astronomical catalog of the VIMOS Public Extragalactic Redshift Survey (`VIPERS`).

In order to verify the accuracy of photometric redshifts, we developed a data structure called `K-d Tree` that separates galaxies into subsets according to their photometric data. In each subset, another *machine learning* was created. It evaluates how accurately is the photometric value of the redshift calculated by the three models for each galaxy is. Through this result, it was possible to exclude galaxies whose photometric redshift is far from the spectroscopic redshift value.

Keywords: Photometric redshift, machine learning, K-d Tree.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

AGN	Active Galactic Nucleus (núcleo galático ativo)
BAO	Baryon Acoustic Oscillations
BINGO	Baryon Acoustic Oscillations from Integrated Neutral Gas Observations
BOSS	Baryon Oscillation Spectroscopic Survey
CFHTLS	Canada-France-Hawaii Telescope Legacy Survey
DEC	Declination (Declinação)
DEEP2	Deep Extragalactic Evolutionary Probe 2
DES	Dark Energy Survey
EQM	Erro quadrático médio
ESO	European Southern Observatory
FRA	Fast Radio Bursts
HI	Hidrogênio atômico
IA	Inteligência artificial
K-d Tree	K-dimensional Tree (árvore de K dimensões)
K-NN	K-Nearest Neighbors (K-ésimo vizinho mais próximo)
PDF	Probability density function
RA	Right Ascension (Ascensão reta)
SDSS	Sloan Digital Sky Survey
SED	Spectral Energy Distributions (distribuição de energia espectral)
SQL	Structured Query Language
t-SNE	t-Distributed Stochastic Neighbour Embedding (Implante estocástico disperso vizinho)
VIPERS	VIMOS Public Extragalactic Redshift Survey
VVDS	VIMOS VLT Deep Survey

Conteúdo

1. Introdução

BINGO (**B**aryon Acoustic Oscillations from **I**ntegrated **N**eutral **G**as **O**bservations) é um projeto que tem como objetivo a construção de um Radio Telescópio na Paraíba, no município de Aguiar. O projeto utiliza o mapeamento da intensidade da radiação de comprimento de onda de 21 cm para observar a distribuição de matéria através do Hidrogênio atômico (HI) como marcador principal. Este comprimento de onda equivale a uma frequência de 1420 MHz. Porém, por causa da expansão do universo, esta frequência é transladada para uma região mais avermelhada do espectro eletromagnético. Isto significa que o BINGO observará efetivamente as frequências entre 980 MHz e 1260 Mhz. O BINGO poderá mapear, através da citada linha do Hidrogênio, o deslocamento para o vermelho - redshift z - no intervalo $0.127 \leq z \leq 0.449$.

BINGO é um telescópio projetado para ser um dos primeiros telescópios a sondar oscilações acústicas de bárions (*Baryon Acoustic Oscillations* - BAO) em frequências de rádio. O BINGO tem dois objetivos, sendo cada um destes objetivos relacionado a um campo da física, da cosmologia e da astrofísica. Na área de cosmologia, o BINGO pretende ser um dos primeiros telescópios a detectar o BAO em baixo redshift para impor restrições aos modelos do setor escuro. No campo da astrofísica, o segundo objetivo é ajudar a descobrir e estudar rajadas rápidas de rádio (*Fast Radio Bursts* - FRB) no hemisfério sul [1, 2, 3, 4, 5, 6, 7, 8].

As observações de rádio têm excelente resolução de frequência, portanto, do *redshift*. Com esta informação conseguimos determinar a distância de um dado observado melhor do que usando dado ótico. Por outro lado, apesar de observações de rádio poderem definir uma posição angular, a configuração do BINGO não permite fazer essa definição com acurácia. Por conta disso, são usados dados óticos para fazer uma localização angular muito precisa, geralmente restrita ao ângulo de visão de 40 minutos de arco.

Há ainda uma importante questão no que concerne à divisão entre dados de rádio e dados óticos. Os redshifts espectroscópicos são obtidos através do espectro eletromagnético de elementos dos objetos astronômicos quando comparados com o espectro eletromagnético obtido desses mesmos elementos em laboratório terrestre. Este processo é demorado, caro e difícil de se fazer para um grande número de galáxias devido às limitações de tempo do telescópio. Por conta disto, olhamos então para os redshifts fotométricos a fim de determinar esta propriedade. Estes são obtidos através da fotometria. Usando as precisas observações obtidas por espectroscopia, podemos usá-las como dados de treino para aprender como os dados fotométricos se comportam. Esta aprendizagem consiste em criar, a partir de um cálculo matemático, um modelo para calcular o redshift fotométrico através dos dados fornecidos pelos catálogos astronômicos. Esse processo é impraticável

no caso de cálculos explícitos por conter muitas variáveis e parâmetros (coeficientes), por isso partimos para um processo computacional denominado *aprendizado de máquina*.

O *aprendizado de máquina* é uma área da inteligência artificial que tem como objetivo construir um modelo que coleta dados e, a partir dessas informações, consegue analisar e fazer inferências para a criação de padrões com estes dados. No caso deste trabalho, as informações utilizadas foram os dados espectroscópicos de um catálogo de dados de galáxias que contém redshifts espectroscópicos precisos e cujos dados fotométricos são também conhecidos. Posteriormente, através do aprendizado de máquina, calculamos o redshift fotométrico de outras galáxias das quais temos apenas os dados fotométricos.

O presente projeto de dissertação consiste em construir e aperfeiçoar métodos de se calcular o redshift fotométrico de galáxias, utilizando dados fotométricos destes objetos. Neste trabalho, utilizamos as magnitudes das galáxias nos filtros m_g, m_r, m_i, m_z e m_Y. Estas magnitudes são decorrentes das limitações de valores dos filtros, cada filtro permite a passagem de frequências entre um intervalo de comprimento de onda e atenua as frequências fora dessa faixa. Os valores permitidos em cada filtro podem ser vistos na figura 1, retirada do artigo de um dos catálogos utilizado nesse trabalho, o Dark Energy Surveys (`DES`) [9]. Uma descrição deste catálogo será feita na seção 2.3.2. Na figura podemos ver os valores da restrição de cada filtro.

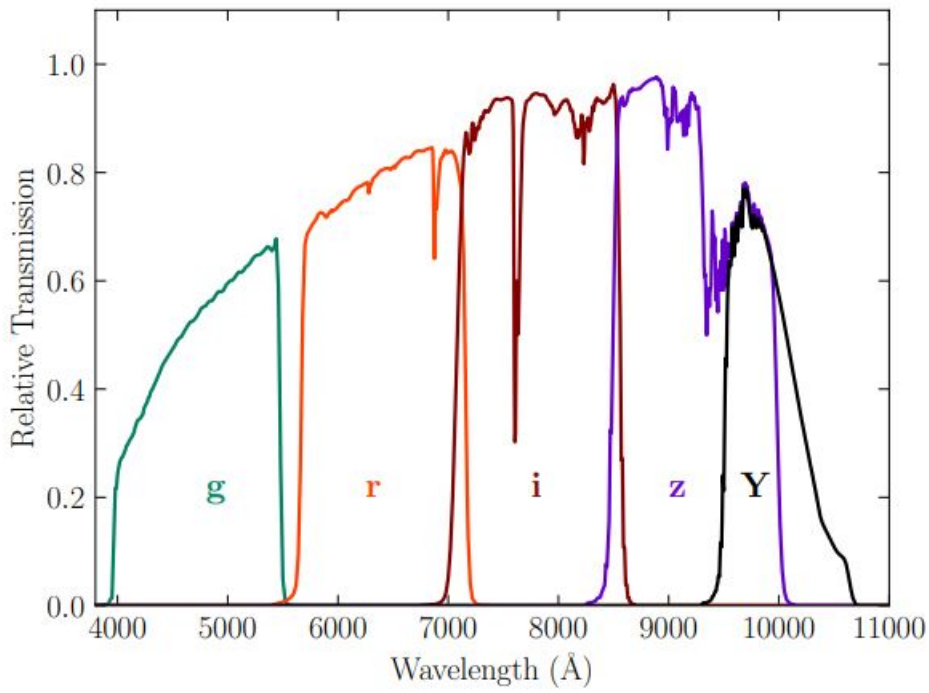


Figura 1: Figura retirada de Abbott et al. [9]. Gráfico da relação dos espectros observados por cada filtro em função do comprimento de onda. As limitações de cada filtro são aproximadamente: 4000Å $< m_g <$ 5500Å, 5500Å $< m_r <$ 7200Å, 7000Å $< m_i <$ 8600Å, 8300Å $< m_z <$ 10000Å, 9400Å $< m_Y <$ 10700Å. O eixo y representa a comparação entre quantidade de fluxo no espectro observado por cada filtro.

Cada filtro isola partes do espectro e mede um fluxo cuja amplitude do sinal depende da frequência que está sendo analisada. Por isso, podemos ver na figura 1 a transmissão relativa para diferentes frequências dentro de um mesmo filtro. Com esses valores, são calculadas as magnitudes. Neste trabalho, utilizamos catálogos que usam o sistema de magnitude AB, que consiste em usar o valor do fluxo de densidade do espectro eletromagnético em cada intervalo de comprimento de onda (filtro). A magnitude é uma propriedade sem unidade já que o seu cálculo (equação 1) é a razão entre a densidade de fluxo do espectro (fração da radiação eletromagnética sobre a área e sobre o comprimento de onda) por uma constante na unidade de Jankys, que é uma unidade para fluxo de densidade dada por $1J_y = 10^{-23} ergs^{-1} Hz^{-1} cm^{-2}$ que elimina as unidades. Estes filtros estão situados no campo de 4000 Å até 11000 Å com cada uma das bandas tendo em média 1500 Å de intervalo. A magnitude é definida através da equação

$$m_t = -2,5 log_{10}\left(\frac{f_\lambda}{3631 J_y}\right) \quad . \tag{1}$$

Na equação (1) é mostrado o cálculo da magnitude do filtro t, em que t pode ser qualquer um dos filtros m_g, m_r, m_i, m_z e m_Y disponíveis nos catálogos. Podemos ver que as unidades dos valores dentro do logaritmo se cancelam, dessa forma a magnitude vira um valor sem dimensão.

Nos catálogos utilizados neste trabalho, já são fornecidos os valores da fotometria das galáxias na forma das magnitudes m_t como mostrado na equação (1). Para fixar apenas um valor da magnitude para determinado filtro, é preciso calcular o valor do fluxo durante o intervalo de comprimento de onda permitido pelo filtro. Por exemplo, na figura 2, a área verde corresponde ao intervalo de comprimento de onda do filtro m_r mostrado na figura 1. O valor de f_λ que irá na equação do cálculo da magnitude (equação 1) é a integral dos fluxos Δf_λ para cada comprimento de onda entre 5500 Å e 7200 Å.

Em nosso trabalho, utilizamos tanto o conceito de magnitude quanto o de cores. A cor é a diferença entre duas magnitudes de filtros consecutivos, como $m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$. Por exemplo, para sabermos o valor da cor $m_g - m_r$ temos que fazer a subtração entre as duas magnitudes como mostrado na equação

$$cor_{m_g - m_r} = m_g - m_r \quad . \tag{2}$$

Hoje, há diversos tipos de aprendizado de máquina capazes de predizer o redshift fotométrico, porém poucos consideram a similaridade fotométrica entre os objetos astronômicos. Nosso intuito é calcular o redshift fotométrico e dividir as galáxias de acordo com as suas características fotométricas, pois são estes os dados utilizados para a predição.

Nesta dissertação, usamos dois conjuntos de dados astronômicos, o Dark Energy Survey (DES) [12] e o VIMOS Public Extragalactic Redshift Survey (VIPERS) [13]. Para baixar

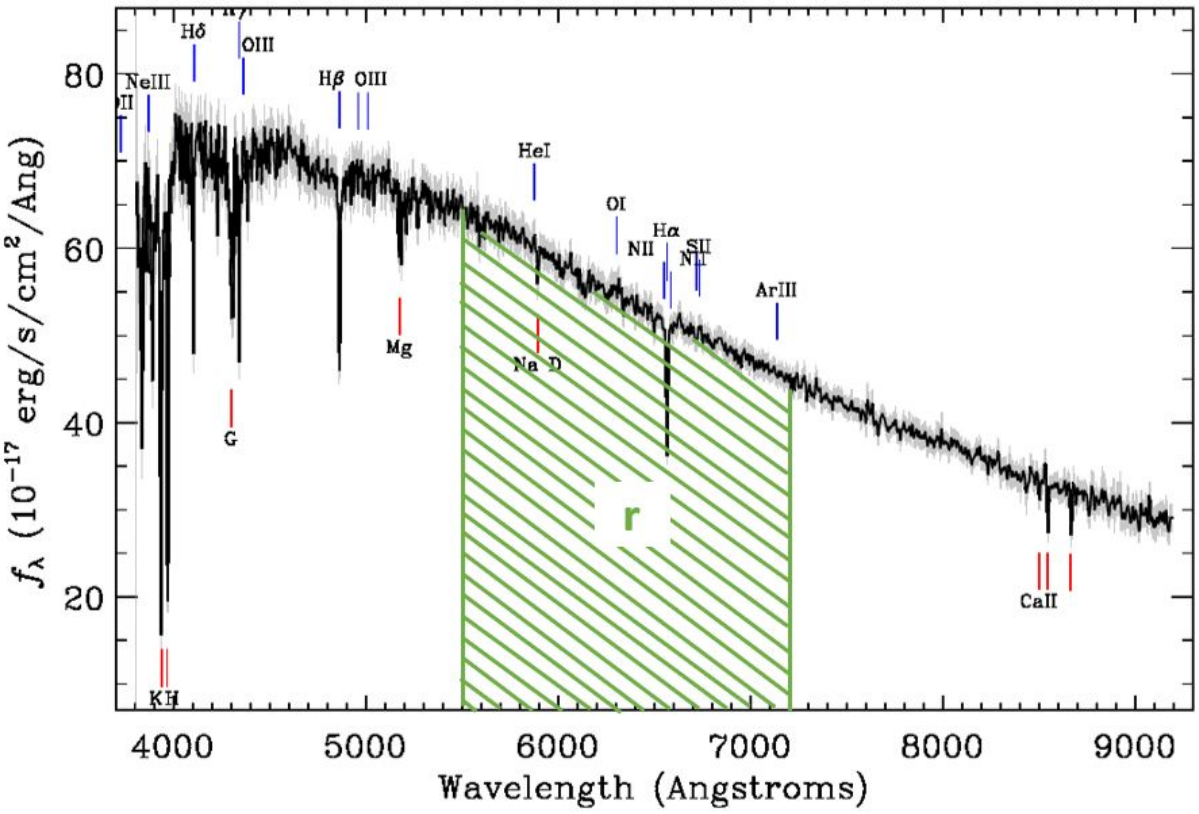


Figura 2: Figura modificada do catálogo do Sloan Digital Sky Survey (SDSS) [10] do artigo Ahn1 et al. [11] de uma estrela (identificador: J112149.55+264535.3). Gráfico do fluxo f_λ em unidades de $10^{-23} ergs^{-1} Hz^{-1} cm^{-2}$ Å^{-1} em função do comprimento de onda em Å. A área hachurada verde corresponde ao intervalo de comprimento de onda do filtro m_r (5500Å a 7200Å). O valor do fluxo do filtro m_r que irá ser usado na equação 1 é a integral de todos os fluxos para cada comprimento de onda dentro do intervalo do filtro m_r, ou seja, a área hachurada verde.

os dados foi escrita uma *query*, que é uma busca específica de informações em um conjunto de dados através de um código. Essa busca tem como resposta uma combinação de dados desse conjunto. O programa, para conseguir essa resposta, foi escrito na linguagem de programação chamada *Structured Query Language* (SQL), de modo a criar um conjunto de dados para ser utilizado em nosso projeto.

Ambos os catálogos (DES e VIPERS) contêm propriedades de galáxias que foram obtidas a partir de diversas manipulações de filtros e tratamentos de dados. Por esta razão, não podemos dizer que galáxias que estão espacialmente no mesmo lugar ou a uma pequena distância entre si - de acordo com as coordenadas Ascensão Reta (RA) e Declinação (DEC) - compartilham das mesmas propriedades fotométricas. Há também o fato de que o DES tem quase 8 mil vezes mais objetos que o VIPERS. Como o aprendizado de máquina usa as informações fotométricas para calcular o redshift fotométrico, a probabilidade de o redshift das galáxias do DES na região, que não é representativa do VIPERS, estar longe do seu valor real é grande. Isso acontece pois o aprendizado de máquina não aprendeu com essa informação fotométrica para calcular o redshift. Por conta disso, foi criada

uma estrutura de dados para dividir as galáxias em grupos de acordo com os valores fotométricos de cada objeto.

Para o treinamento da máquina, é preciso que os dados de entrada sejam parecidos com os dados que a máquina usará para calcular o redshift fotométrico. Se, após a máquina ser treinada, utilizarmos galáxias que não são parecidas fotometricamente com as galáxias que o mecanismo usou para treinar, é possível que a máquina dê um resultado longe do real. Para evitar isso, será feita uma divisão de galáxias em relação a sua fotometria de modo a usarmos em nossos cálculos apenas os redshifts que têm maior probabilidade de ter um erro pequeno. Essas galáxias serão as que mais se parecem, fotometricamente, com as galáxias do `VIPERS`.

Esta divisão será feita pela estrutura de organização de dados chamada árvore `K` dimensional (`k`-d Tree), ou árvore de dados, em que `k` é o número de dimensões usadas na estrutura. Em nosso caso, utilizamos duas árvores de dados com `k` = 4 e `k` = 5. Isto significa que cada estrutura tinha quatro e cinco dimensões respectivamente. Na primeira foram utilizadas as intensidades das 4 cores ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$) e, na segunda, foram usados 5 dados de entrada, as 4 intensidades de cores e a intensidade da magnitude m_i ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$ e m_i). Desta forma, nós calculamos duas estruturas de dados separadas e independentes. Esta estruturação será feita de acordo com a fotometria das galáxias. Esta ordenação separou as galáxias em diversos grupos. Cada conjunto reúne as galáxias que são fotometricamente similares. Ser parecida fotometricamente significa que os valores da fotometria (cor ou magnitude) de duas galáxias são iguais ou têm uma pequena diferença. Por exemplo, uma galáxia `a` do `VIPERS` tem os valores da fotometria próximos (com erro pequeno entre os valores) de uma galáxia `b` do `DES`. Os grupos são definidos pela posição das galáxias em um espaço cujos valores em cada dimensão são as intensidades das cores e das magnitudes. Isto significa que as galáxias com valores parecidos em relação a sua fotometria estarão agrupados no mesmo conjunto. Todo este processo é descrito na seção 4.1.

Dentro desta estrutura, também foi criado um modelo de aprendizado de máquina que nos dá a probabilidade de galáxias serem parecidas fotometricamente. Para a criação desse método, primeiro foi criado um catálogo com as galáxias do `DES` com uma classificação binária indicando quais destas galáxias estão em ambos os catálogos e quais estão só no catálogo do `DES` (seção 3.2). As galáxias que estão em ambos os catálogos foram consideradas representativas dos dois catálogos (classificadas com o número 1) e, as que não estão, são classificadas como não representativas (classificadas com o número 0). A partir da análise da fotometria de todas essas galáxias mais a classificação binária desse objetos, foi treinado um aprendizado de máquina com o algoritmo do K-ésimo vizinho mais próximo para dizer a probabilidade das galáxias do `DES` serem parecidas fotometricamente com as galáxias que estão no `VIPERS`. Deste modelo cada galáxia do `DES` teve um valor atribuído de probabilidade chamada lambda (λ), que vai de 0 a 1. As galáxias do `DES` que

Aprendizados de máquina			
	Métodos de aprendizados de máquina	Dados de entrada	Detalhamento
GPz - Magnitude	Processos Gaussianos	5 magnitudes: m_g, m_r, m_i, m_z, m_Y	seção 5.1
GPz - Color	Processos Gaussianos	4 cores e 1 magnitude: $m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$, m_i	seção 5.1
ANNz	Combinação de diferentes métodos	4 cores e 1 magnitude: $m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$, m_i	seção 5.2
Keras	Redes neurais	4 cores e 1 magnitude: $m_g - m_r$, $m_r - m_i$, $m_i - m_z$d, $m_z - m_Y$, m_i	seção 5.3

Tabela 1: Tabelas do resumo dos aprendizados de máquina utilizados neste trabalho.

não têm informação fotométrica similar às galáxias do `VIPERS` têm um valor de λ mais próximo do 0 e as galáxias que têm informação fotométrica parecidas têm um valor de λ mais próximo de 1. O cálculo da probabilidade é detalhado na seção 4.2.

O cálculo do redshift fotométrico neste trabalho é feito através de três aprendizados de máquina diferentes, duas públicas e uma desenvolvida nesta dissertação. As duas públicas são o `GPz` [14] (seção 5.1), que utiliza processos gaussianos no cálculo do redshift, e o `ANNz` [15] (seção 5.2), que utiliza a combinação de diversos algoritmos de aprendizados de máquina. O aprendizado de máquina desenvolvido neste trabalho utiliza uma interface de programação em Python chamada `Keras` [16] (seção 5.3). Para cada um destes aprendizados de máquina foi utilizada como dados de entrada a fotometria das galáxias.

Com o aprendizado de máquina e a seleção fotométrica das galáxias, conseguimos criar um catálogo com as galáxias do`DES`com valores de redshifts fotométricos. Os detalhes desses modelos serão descritos ao longo desta dissertação. Todos estes métodos foram escritos na linguagem de programação Python.

A partir dos dados fotométricos podemos construir um mapa da distribuição de redshifts fotométricos na área do céu em que está localizado o nosso catálogo fotométrico. Com este mapa, seria possível calcular o espectro de potência da distribuição de galáxias do `DES`. Porém, para isto, os valores dos redshifts têm que estar próximos do valor real.

Na figura 3 temos um diagrama das etapas deste trabalho. Cada fase será detalhada nas próximas seções e capítulos desta dissertação.

Assim, esta dissertação visa aprimorar as técnicas de predição de propriedades astrofísicas ao fazer uma seleção dos melhores dados a serem utilizados para se obterem bons resultados. Outra finalidade é construir um modelo de aprendizado de máquina que possa predizer os dados fotométricos a partir da utilização e combinação de diversos modelos.

Além disto, com o mapa de distribuição dos redshifts, podemos calcular o espectro de potência angular de cada secção desse mapa e obter parâmetros cosmológicos. A partir disto, podemos dar continuidade a uma ciência mais abrangente, com dados fotométricos obtidos, dados espectroscópicos existentes e observações de rádio que teremos do `BINGO` em breve. Por fim, podemos capacitar os alunos, tanto da graduação quanto da pós-graduação, a estudar este tipo de informação e, ao mesmo tempo, preparar o projeto para fazer a correlação dos diferentes tipos de dados astronômicos.

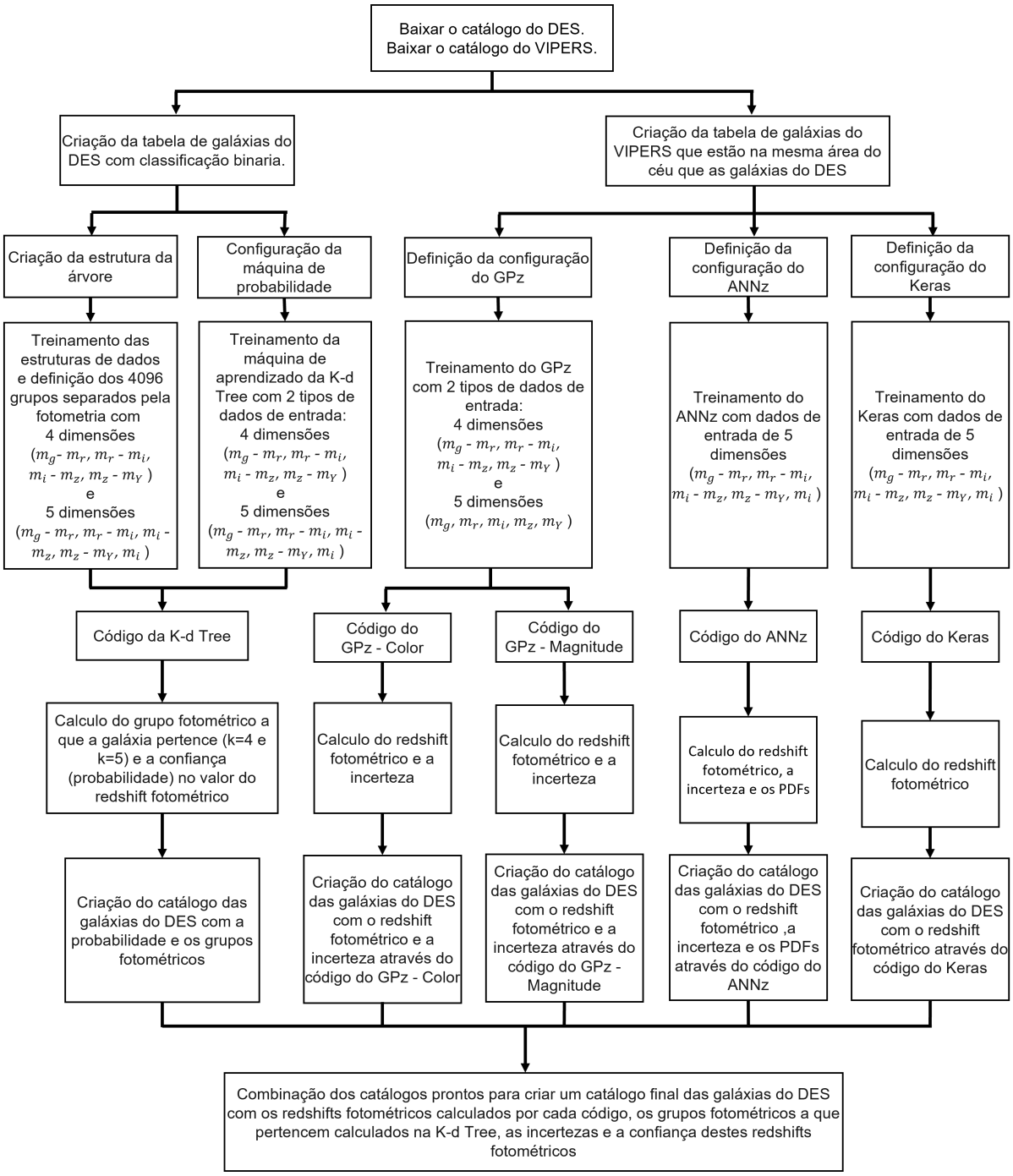


Figura 3: Diagrama das etapas deste trabalho. Cada estágio da dissertação será detalhado nos próximos capítulos. A primeira etapa é responsável por baixar os catálogos astronômicos utilizados neste trabalho, o DES e o VIPERS, este processo é explicado na seção 2.3. A próxima etapa é a do pré processamento dos dados, deixando-os aptos a serem utilizados neste estudo, os detalhes deste processo estão no capítulo 3. A etapa seguinte é a criação das estruturas e aprendizados de máquina a serem utilizadas neste trabalho, para a parte de estrutura de dados desde a sua configuração até a criação do código, que pode ser vista no capítulo 4. A parte de configurar os aprendizados de máquina para o cálculo do redshift fotométrico foi detalhada no capítulo 5. Os resultados do trabalho podem ser vistos no capítulo 6. Todos os códigos utilizados neste estudo podem ser vistos no apêndice 2.3.

2. Aprendizado de máquina e cosmologia

Neste capítulo, descrevemos o conceitos dos recursos utilizados neste trabalho. Na primeira seção, apresentamos o aprendizado de máquina junto com o significado de algumas definições deste mecanismo. Ainda na seção 2.1, vemos um exemplo do funcionamento de uma rede neural simples em 2.1.3. Na seção 2.2, temos a contextualização do redshift, assim como a diferença entre redshift espectroscópico e redshift fotométrico em 2.2.3. Por fim, na seção 2.3, descrevemos os catálogos astronômicos utilizados neste trabalho e o porquê da escolha deles. Na seção A do apêndice, temos um resumo da história do aprendizado de máquina, desde o surgimento e toda a evolução para o que temos hoje.

2.1. Aprendizado de Máquina

O aprendizado de máquina (*Machine learning*) é uma área da inteligência artificial ligada a um sistema que aprende e modifica o seu comportamento através das suas experiências anteriores. É uma ramificação da inteligência artificial (IA) que utiliza algoritmos e técnicas estatísticas para criar padrões a partir dos dados fornecidos [17] [18]. É empregado em diversos campos, como o de reconhecimento de voz e de imagem, diagnóstico médico e estatística de mercado, entre outras áreas.

Uma rede neural é um subconjunto do aprendizado de máquina que, por sua vez, é um subgrupo da inteligência artificial. A inteligência artificial é uma área da ciência da computação que se refere à capacidade de uma máquina de reproduzir o raciocínio do ser humano. Sempre que um computador realiza tarefas com base em um conjunto de regras programadas, os algoritmos, o computador está funcionando por meio da inteligência artificial. O aprendizado de máquina é um mecanismo em que são utilizados algoritmos para coletar dados, aprender com estes dados e, então, fazer uma determinação, predição ou até executar uma tarefa a partir dessas informações. Este mecanismo é um programa que consegue aprender por conta própria e aprimorar o seu desempenho com a experiência.

De forma ilustrativa, podemos dizer que o aprendizado de máquina serve como um motor para a inteligência artificial. Ao utilizar algoritmos complexos, o mecanismo consegue realizar uma coleta de dados, interpretá-los e tomar decisões, executando tarefas de modo automatizado. As redes neurais ou aprendizado de máquina profundo são uma forma de simular o cérebro humano em um nível mais avançado. As redes neurais atuam como camadas em cadeia que, de modo hierarquizado, conseguem analisar, por meio de um complexo processamento de dados, informações mais específicas e complexas. Este mecanismo é uma das formas de viabilizar o treinamento do aprendizado de máquina [19]. Há outras formas como a estrutura de dados e os processos gaussianos. Tais dispositivos serão discutidos mais à frente nesta dissertação.

A inteligência artificial é um sistema que possibilita que as máquinas imitem o comportamento do cérebro humano. O aprendizado de máquina é um dispositivo capaz de aprender de forma autônoma com base na observação e na análise de dados. A rede neural é uma tecnologia que não só aprende com esses dados, como também consegue criar padrões a partir destas informações, adaptando-se a diferentes cenários.

Outros tipos de inteligência artificial, além do aprendizado de máquina, são: raciocínio, processamento de linguagem natural e planejamento automatizado. O raciocínio é o processo lógico da máquina para inferir resultados. O desenvolvimento desta `IA` é feito com base nos passos lógicos que humanos fazem para resolver um problema e para fazer inferências. Porém, ao contrário de humanos, que também usam argumentos intuitivos para resolver problemas, o algoritmo do raciocínio não utiliza argumentos intuitivos. Um exemplo deste tipo de algoritmo são os exercícios de emergência utilizados em aeroportos para treinar os funcionários no caso de algum ataque terrorista [20]. O programa cria cenários a partir do raciocínio do que terroristas iriam fazer para ter êxito e os funcionários têm que aprender a solucionar os problemas que são criados. O processamento de linguagem natural serve como um tradutor que permite que a tecnologia entenda o usuário utilizando a sua linguagem natural. Exemplos disso são os sistemas de inteligência artificial em nossos celulares ou uma conversa com um robô das empresas que converte o que falamos em uma linguagem que a máquina entenda. Um exemplo deste tipo de robô é o `ChatGPT`[1], divulgado em novembro de 2022, que, através da `IA`, gera diálogos virtuais. O processamento de linguagem natural utiliza algoritmos que realizam o processamento de linguagem, possibilitando a interação humana com o computador, sendo assim necessário para que a máquina compreenda o que o usuário fala e possa responder da melhor forma [18]. No que tange ao planejamento automatizado, este é definido como um processo de análise da máquina que age de forma autônoma para construir uma sequência de ações até um objetivo final. Exemplos de planejamento automático são os gerenciadores de projeto e planejamento de processos de manufatura [21].

No âmbito do aprendizado de máquina temos outros algoritmos além das redes neurais. Exemplos são o "K-ésimos vizinhos mais próximos"(*K-Nearest Neighbors* - `K-NN`) e a árvore k-dimensional (*K-dimensional Tree* - `K-d Tree`). Todos esses algoritmos serão explicados detalhadamente mais à frente no texto.

2.1.1. Tipos de Aprendizado de Máquina

Há três tipos de aprendizados de máquina: o aprendizado de máquina supervisionado (*Supervised Learning*), o aprendizado de máquina não supervisionado (*Unsupervised Learning*) e o aprendizado por reforço (*Reinforcement Learning*) [22] [23] [24].

[1] https://openai.com/blog/chatgpt/qwe.shF../

No aprendizado de máquina supervisionado, a máquina aprende a partir de um conjunto de dados de treinamento, ou seja, são fornecidos à máquina os dados de entrada e de saída. Com isso, a rede cria um padrão com os dados fornecidos. Dessa forma, a máquina pode predizer os resultados a partir de novos dados. Esse tipo de mecanismo é utilizado em classificações e regressões.

A classificação é o algoritmo que atua para categorizar os dados, como a detecção de fraudes em identidades ou a classificação de imagens. A regressão é o algoritmo que fornece respostas contínuas, como a predição do crescimento de uma população ou a previsão do tempo.

No aprendizado de máquina não supervisionado, não há um resultado definido do conjunto de dados. A máquina examina os dados fornecidos e, com base nessas informações, cria um padrão, ou até prediz algum dado. Isso acontece mesmo que o usuário não diga o que está querendo como produto final. Esse tipo de máquina é utilizado em redução de dimensionalidade e associações.

Para a redução de dimensionalidade, temos, como exemplo, quando o usuário tem um conjunto de amostras muito grande e quer diminuí-lo de forma a facilitar a visualização dos dados, mas sem perder muita informação. Para as associações (*clustering*), temos os segmentos de clientes e marketing direcionado. Um exemplo conhecido de aprendizado de máquina não supervisionado é o Implante Estocástico Disperso Vizinho (*t-Distributed Stochastic Neighbour Embedding* - `t-SNE`) que usa a diminuição de dimensionalidade para transformar dados com muitas dimensões em dados com baixa dimensionalidade. Este algoritmo age calculando a probabilidade da proximidade dos pontos no espaço de muitas dimensões. A proximidade dos focos é determinada como a probabilidade condicional de um ponto **A** escolher o ponto **B** como seu vizinho. Então é feita a mesma coisa no espaço de poucas dimensões correspondentes. Depois disto, o algoritmo minimiza a diferença entre as probabilidades dos espaços com dimensões diferentes para a melhor representação da informação no espaço de menor dimensão. Isto significa que o programa coleta a correlação entre dois pontos em uma espaço de muitas dimensões e transforma essa correlação para um espaço de menor número de dimensões. Por conta disso, o `t-SNE` é útil para mapear informações multidimensionais como modelo de imagem e informação genômica.

No aprendizado por reforço, o algoritmo aprende o comportamento ao desempenhar ações. Como no aprendizado supervisionado, há um resultado desejado, porém a sua configuração é mais abstrata. A máquina realiza ações e, a partir dessas ações, a rede recebe uma resposta positiva, se for uma ação desejável para o usuário, ou uma resposta negativa, se for o contrário. O objetivo da rede é maximizar a quantidade de reforços positivos recebidos. Esse aprendizado envolve aprender sem ter os dados classificados, ou seja, a rede aprende a partir da sua própria experiência [25]. Esse tipo de rede neural é utilizado em jogos e decisões em tempo real.

Neste presente trabalho, foi utilizado o modelo de aprendizado de máquina supervisionada devido ao fato de que os dados utilizados já possuem o dado de saída de que precisamos, o redshift fotométrico. Isto significa que damos à máquina os dados de entrada e os de saída. Mais detalhes destes dados estão na seção 2.3.1.

2.1.2. Exemplo qualitativo de uma rede neural

Em se tratando de redes neurais, uma das formas mais fáceis de se entender o mecanismo é a partir da comparação com o cérebro humano e os seus respectivos neurônios. Estes últimos são os responsáveis por toda a nossa capacidade de pensar e lembrar. Cada um deles é conectado a até 200 mil outros neurônios através das sinapses [2]. Além disto, transportam a informação (impulso nervoso) através de um fenômeno eletroquímico que ocorre nas células nervosas. Tais neurônios têm diferentes funções e formatos, sendo classificados a partir destas diferenças [26] [27].

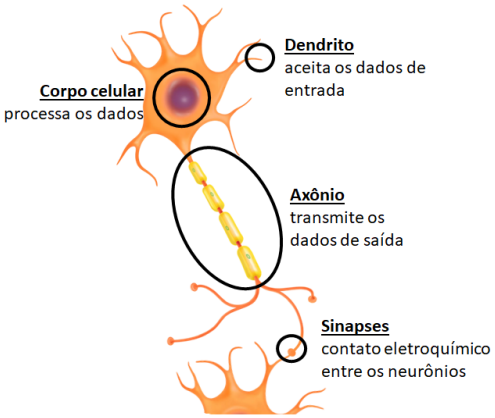


Figura 4: Representação simplificada do funcionamento de um neurônio. Para cada informação inserida no neurônio há uma cadeia de eventos a ser seguida. Primeiro, o dado entra no neurônio através dos dendritos. Depois, passa pelo corpo celular, onde fica o núcleo que processa os dados, neste caso, os estímulos excitatórios ou inibitórios. Após isso essa informação é transmitida para o axônio e por fim para os seus terminais que fazem a sinapse com outros neurônios, continuando o percurso do estímulo. Figura modificada do projeto Verification & Validation Of Neural Networks For Aerospace Systems [28].

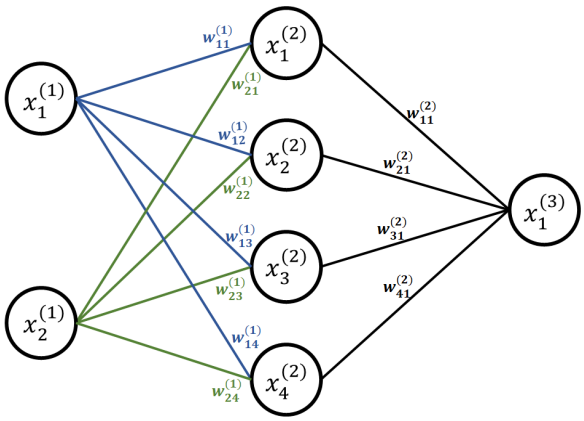


Figura 5: Exemplo de uma rede neural com três camadas: a camada de entrada, a camada interna e a camada de saída. Em cada nó o *input* será multiplicado pelos seus pesos. Os pesos que multiplicam o *input* de cada nó estão divididos por cor. Os *inputs* do primeiro nó ($x_1^{(1)}$) serão multiplicados pelos pesos em azul ($w_{n1}^{(1)}$), enquanto os *inputs* do segundo nó ($x_2^{(1)}$) serão multiplicados pelos pesos em verde ($w_{n2}^{(1)}$). Em que n é o número do nó em que o *input* será processado, nesse caso temos que n vai de 1 até 4, já que são 4 nós na segunda camada. Baseado na figura 4 do artigo Adcock et al. [29] .

[2] A sinapse é uma região de proximidade entre a terminação nervosa (dendrito) de um neurônio e outros neurônios, células musculares ou células glandulares por onde é transmitido o impulso nervoso [26]

Na figura 4, temos uma representação simplificada de como seria o neurônio. É importante notar que há uma entrada de dados (*inputs*), um corpo celular que processa os dados e a saída dos dados (*outputs*) que está conectada a outros neurônios através das sinapses. A partir dessa imagem, podemos ver que há três estruturas bem definidas do neurônio que usam os dados fornecidos. No núcleo do neurônio os *inputs* são processados de acordo com uma função determinada.

Ao compararmos com uma rede neural, podemos reconhecer semelhanças entre as duas estruturas. A figura 5 é um exemplo da configuração de uma rede neural com as três estruturas: a entrada de dados, o núcleo que processa os dados e a saída de dados. Os dados de entrada são colocados na máquina, no núcleo da máquina estes dados são multiplicados pelos pesos $w(n)$ e utilizados na função de ativação pré definida. Cada função pode produzir resultados que são propagados ao longo do núcleo para outras funções ou para a saída do neurônio.

2.1.3. Modelo de rede neural

Para criar o modelo da rede neural são necessários três conjuntos de dados: o conjunto de treinamento, o conjunto de teste e o conjunto de validação.

O **conjunto de dados de treino** (*training set*) contém os dados usados para criar o modelo. São as informações fornecidas à rede para criar os padrões.

O **conjunto de dados de teste** (*test set*) é utilizado para fazer uma avaliação imparcial do modelo final da rede. Na prática, este grupo tem o mesmo conceito do conjunto de validação, porém a sua análise não é usada no modelo, serve apenas para que o usuário possa avaliar a performance do modelo [30] [25].

O **conjunto de dados de validação** (*validation set*) é um grupo de dados utilizados para o treinamento do aprendizado de máquina que não pertence ao conjunto de treino e que serve para fazer uma avaliação imparcial do modelo. Auxilia a comparar os diferentes modelos e hiperparâmetros definidos pelo aprendizado de máquina. É como se a própria rede fizesse uma análise do seu modelo com outros dados além dos de treinamento e melhorasse os seus parâmetros a partir desses resultados.

A rede neural ainda pode ter baixa performance devido ao sub-ajuste (*underfitting*) e o sobre-ajuste (*overfitting*). Esse problema pode ter diversas causas, sendo as principais relacionadas aos dados de treinamento que contêm muito ruído[3], ou ainda, se os dados não foram pré-processados corretamente ou o conjunto de treinamento está incompleto (a máquina precisa de mais informações para criar um padrão) [31].

O **sub-ajuste** é quando o modelo de rede neural não consegue aprender as relações entre as variáveis dos dados ou predizer um novo dado corretamente. Isto significa que a

[3]Ruído são as flutuações indesejadas introduzidas em um sinal de origem durante a observação, armazenamento, transmissão ou processamento de seus dados. Quanto maior o ruído, menor a qualidade do dado.

máquina não conseguiu criar um padrão para os dados fornecidos. Tal acontecimento pode ocorrer quando o modelo foi treinado usando parâmetros errados ou o modelo definido é muito simples para a complexidade dos dados fornecidos. A consequência disto é que a rede neural não consegue observar a relação entre os dados de entrada e os de saída. Esse problema é fácil de se identificar ao olharmos o erro entre o dado real e o predito.

O **sobre-ajuste** ocorre quando a máquina configura todo ponto como dado importante da rede, incluindo o ruído e pontos fora do padrão. Desta forma, a rede tenta criar uma relação em que todas as variáveis dos dados sejam importantes. Isto faz com que a rede memorize os padrões desses dados sem generalizar, o que significa que a rede decorou os valores do conjunto de treino, então, se colocássemos os mesmos dados do conjunto de treino, a máquina conseguiria devolver os valores iguais aos dados de saída, resultando em uma acurácia de 100%. Porém, se colocarmos novos valores, veremos que a máquina irá devolver valores aleatórios e longe dos números dos dados de saída, porque não aprendeu o padrão, apenas decorou os valores do conjunto de treinamento. Neste caso, a rede tem pouca flexibilidade em predizer novos dados, focando apenas nos que já foram fornecidos. Este fato pode acontecer quando o modelo foi treinado usando parâmetros errados ou o modelo definido é mais complexo do que o necessário para os dados fornecidos e não foi treinado com um conjunto de dados mais variado. Podemos ver isto acontecer quando a rede neural tem uma melhor performance no conjunto de dados de treinamento do que no de testes. Em resumo, o modelo memorizou os dados, tornando a rede enviesada para os pontos de treino e, por isso, não consegue generalizar para outros dados de entrada [32] [33].

Na figura 6, temos um exemplo dos problemas citados para 30 pontos aleatórios marcados no diagrama xy. As curvas em azul são padrões que a rede neural conseguiu criar a partir desses pontos, neste caso, a rede criou um polinômio que tivesse o menor erro entre a curva e os pontos. Os pontos verdes são os que foram usados no treinamento e os pontos em vermelho são os pontos de teste. Na primeira imagem, o modelo retornou um polinômio de ordem 1. Podemos ver que não há um bom ajuste da curva em relação aos pontos, dando um erro de valor elevado. Na segunda curva, o modelo retornou um polinômio de grau 5. A curva se ajustou aos pontos de treino (verde) e conseguiu criar um padrão de forma que, quando essa curva é comparada com os pontos de teste (vermelho), o erro ficou dentro do esperado. Logo, a rede conseguiu criar um padrão com a informação fornecida a ele. Na terceira imagem, a rede retornou um polinômio de grau 12. O erro calculado para os dados de treino é o menor das três curvas. Porém, quando traçamos essa curva para os pontos de teste, podemos ver que o erro é grande. Portanto, a rede memorizou os pontos de treino, mas não conseguiu criar um padrão para generalizar para todos os pontos. Note-se que a curva está passando exatamente onde os pontos de treino estão localizados no diagrama, porém para os pontos de teste a máquina não consegue seguir o padrão desses pontos. Isto resulta em um erro elevado para os pontos de teste.

É possível ver esta configuração comparando com a segunda imagem em que a distância entre os pontos de teste da curva é menor do que na terceira imagem.

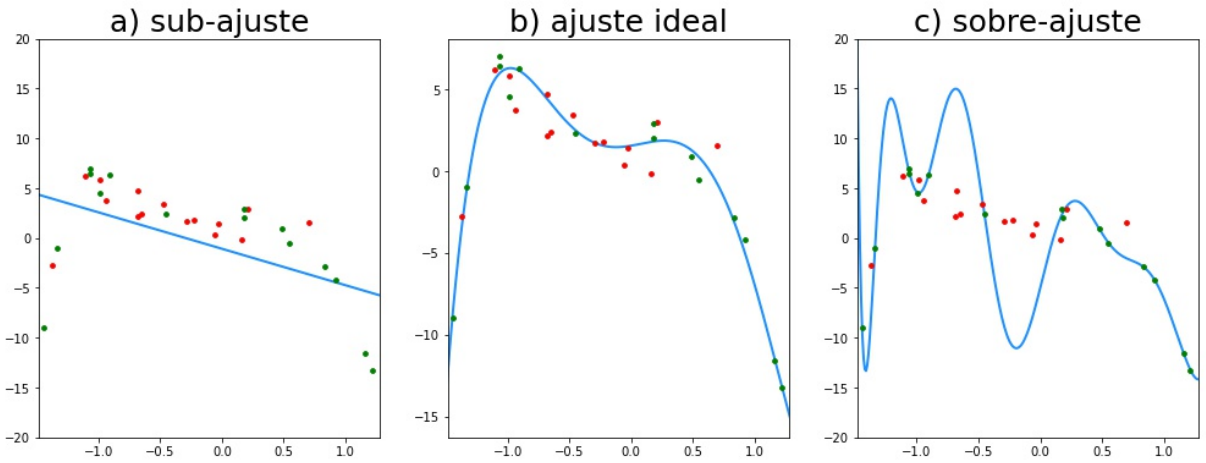


Figura 6: Exemplo do problema de ajustes que pode acontecer em uma rede neural. Os pontos vermelhos são os dados fornecidos à rede, os pontos verdes são os dados do conjunto de teste e as curvas em azul são o resultado do ajuste de cada um dos modelos. Na imagem 6a, temos que a rede neural não conseguiu descobrir um padrão entre os pontos, o que implica em um erro grande. A imagem 6b é o ajuste ideal, em que a rede tem um erro dentro do esperado e consegue predizer parte dos dados corretamente. Na imagem 6c, podemos ver que a rede neural decorou os dados, ou seja, a máquina é adequada apenas para os dados de treino e não consegue generalizar para outros dados de entrada.

A rede neural cria a função

$$y = \mathfrak{F}(\mathbf{x}) \quad , \tag{3}$$

onde x é o vetor com os parâmetros dos dados de entrada e y é o resultado que queremos calcular. O objetivo de uma rede neural é criar a função (3) que recebe os dados de entrada e dá como resultado os dados de saída, que o usuário quer. No caso deste trabalho, x serão as informações das magnitudes das galáxias enquanto y será o redshift fotométrico. O detalhe dessa rede neural para estas propriedades será explicado na seção 5.3. Primeiro, definimos uma rede neural simples para mostrar como a função F é caracterizada [25] [33].

No inicio desta subsecção, mencionamos que um modelo de rede neural necessita de três conjuntos de dados. Além disso, a rede neural também é composta por, no mínimo, três seções, a saber: a seção de entrada (*input layer*), a seção interna (*hidden layer*) e a seção de saída (*output layer*). As seções externas não variam em número. Já as seções internas podem ter uma, duas ou mais camadas, isso depende do propósito da rede neural. O dado de saída de cada camada serve como dado de entrada da próxima camada. Além disto, a rede pode ter diferentes números de nós em cada camada.

Usando a figura 5 para a construção de um exemplo de como funciona uma rede neural, vamos denominar o *input* da camada k como $x_i^{(k)}$ e o *output* dessa mesma camada é

definido como $y_i^{(k)}$. Esta rede neural apresenta apenas uma camada interna. Cada nó representa um neurônio e o peso associado entre dois nós de diferentes camadas caracteriza a "força"da ligação entre esses dois neurônios. Quanto maior o valor desse peso entre os nós, maior a influência do dado de entrada no nó conectado [29].

Podemos considerar um conjunto de dados iniciais de um número aleatório de objetos. Cada um desses objetos tem informações de um vetor na forma $\boldsymbol{x} = (x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, ..., x_d^{(1)})$, em que d é o número de parâmetros de cada dado de entrada. No caso do nosso exemplo, temos $d = 2$, logo teremos que $\boldsymbol{x} = (x_1^{(1)}, x_2^{(1)})$, em que $x_i^{(1)}$ será um número real dos dados. Isso significa que o número de dados de entrada (*inputs*) da primeira camada é igual ao número de parâmetros do vetor do dado de entrada da rede neural. O que sai dessa primeira camada de entrada para a rede é o vetor $\mathbf{x}$ com os dois *inputs*.

Os dados de entrada da primeira camada são multiplicados pelos pesos $\mathtt{W}^{(1)}$ antes de entrar na área com as camadas intermediárias. Estes dados serão processados dentro da rede, adicionando o viés ao seu valor e passando pelas funções de ativação das camadas intermediárias. Este processo será detalhado nesta seção mais à frente. Por fim, ao sair desta camada intermediária, o resultado será multiplicado pelos pesos $\mathtt{W}^{(2)}$ antes de entrar na camada de saída. Entre as camadas intermediárias, não há multiplicação de pesos aos dados. No exemplo, o dado de entrada da segunda camada é calculado com os pesos da forma

$$x_n^{(2)} = \sum_{t=1}^{2} w_{tn}^{(1)} x_t^{(1)} + b_n$$

, em que n é o número de nós na camada $k = 2$, no caso da figura temos 4 nós na segunda camada, ou seja, $n = 4$ e o b_n é o viés (bias) da rede nessa camada para cada nó.

O viés é um fenômeno que coloca o produto do modelo a favor ou contra o resultado esperado. É considerado um erro sistemático que ocorre no modelo da rede neural e pode ser causado por diversos motivos, tais como viés do algoritmo, viés da amostra, viés da exclusão e viés do preconceito. No viés do algoritmo temos um modelo mal escrito que não compreende a complexidade dos dados e por isso resulta em uma diferença considerável entre os resultado e o esperado. O viés da amostra é quando os dados não foram devidamente pré-processados apresentando um número insuficiente ou informações com muito ruído. O viés da exclusão é a supressão de dados importantes, o que faz com que o modelo crie um padrão fora do desejado. O viés de preconceito é quando os próprios dados já estão enviesados, por exemplo, um modelo que tem como objetivo trabalhar com a distribuição de profissões na ciência por gênero, mas a amostra fornecida tem apenas mulheres que são físicas e homens que são químicos. O modelo impreterivelmente prevê a existência apenas de mulheres na física e homens na química [34][35].

Em Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks de Russel Reed et al. lançado em 1999 [25], o bias, ou viés, é definido como

$$b(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta, \tag{4}$$

em que o θ é o valor real que nós temos dos dados de treinamento e $\mathbb{E}(\hat{\theta}_m)$ é o valor calculado pela rede neural para os dados de θ_m. Se $b(\hat{\theta}_m) = 0$ podemos dizer que a rede neural é não enviesada, ou seja, que $\mathbb{E}(\hat{\theta}_m) = \theta$

Na rede neural do nosso exemplo, teremos que o *output* da segunda camada é dado por $y_i^{(2)} = f(x_i^{(2)})$ em que f é uma função de ativação. É importante ressaltar que apenas as camadas internas têm função de ativação, ou seja, os dados na primeira camada e na última camada não passam por função de ativação. Essa função de ativação é a mesma para todos os nós dessa camada, porém para outras camadas pode-se ter diferentes funções de ativação. Isso significa que se a rede neural tivesse duas camadas internas, poderíamos ter a primeira camada interna ($k = 2$) com uma função f_2 enquanto que na outra camada interna ($k = 3$) teríamos outra função de ativação f_3, sendo que as funções podem ou não ser iguais, dependendo da configuração definida pelo usuário para a rede.

A função de ativação tem como objetivo adicionar não-linearidade à rede neural. Por exemplo, se não houvesse essa função, independentemente do número de camadas, o *output* final seria apenas uma transformação linear do *input* da primeira camada com o viés e os pesos, e basicamente todas as camadas iriam se comportar da mesma maneira. Desta forma, a nossa rede neural não conseguiria obter bons resultados para *outputs* mais complexos e seria apenas um modelo de regressão linear [33]. Além disto, também pode ser chamada de função restritiva, já que algumas funções limitam o intervalo da amplitude do sinal de saída a um valor finito, normalmente o intervalo vai de [0,1] ou de [-1,1]. Isto significa que os dados de entrada são normalizados pelas funções de ativação para ficarem dentro destes intervalos. Este limite ajuda a centralizar os dados, o que facilita o treinamento do método. Porém, esta limitação não é uma regra geral da função de ativação. Há outras funções que não limitam os dados. A escolha de limitar os dados depende de qual é a finalidade do aprendizado de máquina em questão.

As funções podem ser: sigmoid, tanh, binária, linear ou reLu, entre outras. Na função sigmoid, também chamada de função logística, é fornecido um valor real como dado de entrada e, como dado de saída, a função retorna valores de 0 a 1. Quanto maior o *input* (mais positivo), mais perto de 1, e quanto menor o *input* (mais negativo), mais perto de 0. A função é calculada como $f(x) = 1/(1 + e^{-x})$. A função da tangente hiperbólica toma os valores de um dado de entrada e fornece como saída valores entre -1 e 1, quanto maior o *input* (mais positivo), mais perto de 1, e quanto menor o *input* (mais negativo), mais perto de -1. A função de ativação da tangente hiperbólica é dada por $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$. A função binária depende do limite fixado pelo usuário

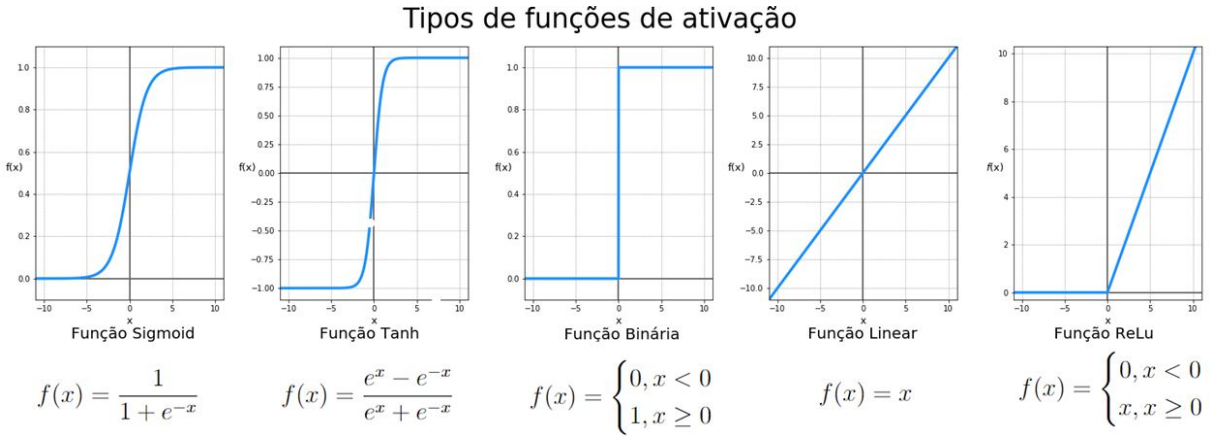


Figura 7: Representação gráfica das funções de ativação descritas no texto junto com as suas definições matemáticas.

para ativar o neurônio ou não, se escolhermos um valor qualquer a para ser o limite e o *input* for menor que este valor, a função dará como resultado 0, se o *input* for maior que a, a função dará como resultado o valor 1. A função de ativação linear é o que seria a rede neural sem função de ativação, onde a ativação é proporcional ao *input*, a sua função pode ser dada por $f(x) = cx + d$ em que c e d são números reais. A função ReLu (Unidade Linear retificada) ativa apenas os neurônios cujos dados de entrada são maiores que 0, para estes dados a função ReLu se comportará como função de ativação linear, conforme explicado anteriormente. Para os *inputs* menores que 0 (negativos), a função dará como resultado o valor 0 para a rede neural, isso significa que o valor deste dado de entrada não será mais considerado na rede. O gráfico de cada uma dessa funções pode ser visto na figura 7. Essas são as funções de ativação mais conhecidas, há outras que são leves modificações de alguma função citada acima e outras que se encaixam em redes neurais mais específicas [36].

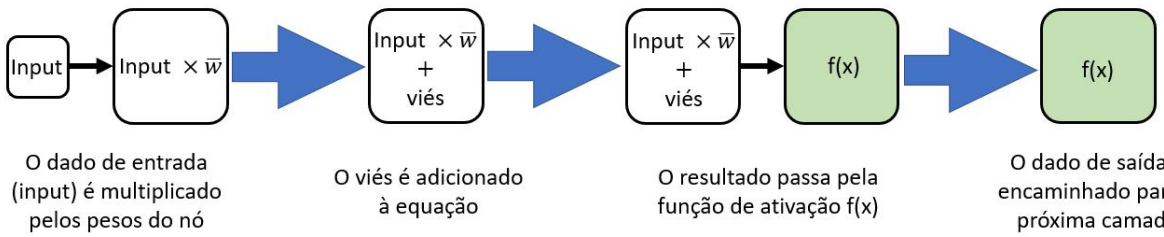


Figura 8: Representação dos passos lógicos que ocorrem com os dados na entrada, dentro e na saída de um nó de uma rede neural.

Para a rede neural do exemplo, a saída da camada interna é dada por $y_n^{(2)} = f(x_n^{(2)})$ e irá para a próxima camada, no nosso caso, a camada de saída. O esquema de como acontece em uma camada interna é representado na figura 8. Após passar pela função de ativação, o dado de saída de cada nó será dado por $y_n^{(2)} = f\left(x_n^{(2)}\right) = f(\sum_{t=1}^{2} w_{tn}^{(1)} x_t^{(1)} + b_n\Big)$ em que $n = 1, 2, 3, 4$ representa os 4 nós dessa camada [17].

Para a última camada, faremos o mesmo processo de multiplicar os dados de entrada pelos pesos. Logo, considerando os pesos da ligação entre os nós, podemos escrever que o resultado do nó dessa camada externa será dado por $x_1^{(3)} = \sum_{n=1}^{4} w_{n1}^{(2)} y_n^{(2)} + b$, em que b é o viés dessa camada. Como não há função de ativação, temos que $y_1^{(3)} = x_1^{(3)}$.

Reescrevendo as equações citadas acima para $x_1^{(3)}$ e $y_1^{(3)}$, temos que o produto final da rede neural será

$$x_1^{(3)} = \sum_{n=1}^{4} w_{n1}^{(2)} y_n^{(2)} + b \rightarrow x_1^{(3)} = \sum_{n=1}^{4} w_{n1}^{(2)} f\Big(\sum_{t=1}^{2} w_{tn}^{(1)} x_t^{(1)} + b_n \Big) + b = y_1^{(3)} \quad . \tag{5}$$

Considerando a equação (3) e a relação $\mathbf{x} = (x_1^{(1)}, x_2^{(2)})$, podemos dizer que

$$\mathfrak{F}(\mathbf{x}) = \sum_{n=1}^{4} w_{n1}^{(2)} f\Big(\sum_{t=1}^{2} w_{tn}^{(1)} x_t^{(1)} + b_n \Big) + b = y. \tag{6}$$

Outra forma de se obter este resultado é através de matrizes. Supomos que temos um conjunto de objetos em que, cada dado de cada objeto, tem 2 *inputs*. Então o dado de entrada na primeira camada será $\mathbf{x}$,

$$\mathbf{x} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix}. \tag{7}$$

Antes dos dados de entrada passarem pela segunda camada, são colocados pesos em cada um dos dados de entrada. Esse peso é dado pela matriz

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)} \end{bmatrix}. \tag{8}$$

A matriz do viés é dada por

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}. \tag{9}$$

A partir das equações (7), (8) e (9) podemos escrever

$$\left(\mathbf{x}^T.\mathbf{W}^{(1)}\right)^T + \mathbf{b} \rightarrow \tag{10}$$

$$\left(\begin{bmatrix} x_1^{(1)} & x_2^{(1)} \end{bmatrix} \times \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} & w_{14}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} & w_{24}^{(1)} \end{bmatrix} \right)^T + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} x_1 w_{11}^{(1)} + x_2 w_{21}^{(1)} + b_1 \\ x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} + b_2 \\ x_1 w_{13}^{(1)} + x_2 w_{23}^{(1)} + b_3 \\ x_1 w_{14}^{(1)} + x_2 w_{24}^{(1)} + b_4 \end{bmatrix} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \\ x_4^{(2)} \end{bmatrix}. \tag{11}$$

O vetor de entrada foi transposto de forma a combinar as dimensões das matrizes para o cálculo na rede neural [25], já que frequentemente as redes neurais processam matrizes de pesos e de dados de entrada de diferentes tamanhos [37]. A matriz 4x1 resultante dessa operação é o dado de entrada na camada interna e passará pela função de ativação f desta camada,

$$f\Big(\left(\mathbf{x}^T.\mathbf{W}^{(1)}\right)^T + \mathbf{b}\Big) = \begin{bmatrix} f\left(x_1^{(1)} w_{11}^{(1)} + x_2^{(1)} w_{21}^{(1)} + b_1\right) \\ f\left(x_1^{(1)} w_{12}^{(1)} + x_2^{(1)} w_{22}^{(1)} + b_2\right) \\ f\left(x_1^{(1)} w_{13}^{(1)} + x_2^{(1)} w_{23}^{(1)} + b_3\right) \\ f\left(x_1^{(1)} w_{14}^{(1)} + x_2^{(1)} w_{24}^{(1)} + b_4\right) \end{bmatrix} = \begin{bmatrix} f\left(x_1^{(2)}\right) \\ f\left(x_2^{(2)}\right) \\ f\left(x_3^{(2)}\right) \\ f\left(x_4^{(2)}\right) \end{bmatrix} \quad . \tag{12}$$

Antes de multiplicar essa matriz de saída pela matriz do peso, transpomos a matriz. Isso acontece para que o produto final seja uma matriz 1x1, ou seja, o dado de saída terá apenas 1 dimensão,

$$\left(f\Big(\left(\mathbf{x}^T.\mathbf{W}^{(1)}\right)^T + \mathbf{b}\Big)\right)^T \times \mathbf{W}^{(2)} \rightarrow \tag{13}$$

$$\begin{bmatrix} f(x_1^{(2)}) & f(x_2^{(2)}) & f(x_3^{(2)}) & f(x_4^{(2)}) \end{bmatrix} \times \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \\ w_{41}^{(2)} \end{bmatrix} \tag{14}$$

$$= \left[f(x_1^{(2)}) w_{11}^{(2)} + f(x_2^{(2)}) w_{21}^{(2)} + f(x_3^{(2)}) w_{31}^{(2)} + f(x_4^{(2)}) w_{41}^{(2)} \right]. \tag{15}$$

Adicionando o viés dessa camada, que é um valor de dimensão 1, já que há um viés para cada nó e a camada externa tem apenas 1 nó, temos o resultado

$$\left[f(x_1^{(2)}) w_{11}^{(2)} + f(x_2^{(2)}) w_{21}^{(2)} + f(x_3^{(2)}) w_{31}^{(2)} + f(x_4^{(2)}) w_{41}^{(2)} + b \right]. \tag{16}$$

Portanto, temos que a função da rede neural é dada por

$$\begin{aligned}\mathfrak{F}(\mathbf{x}) =& \left(f\left(\left(\mathbf{x}^T.\mathbf{W}^{(1)}\right)^T + \mathbf{b} \right) \right)^T \times \mathbf{W}^{(2)} + b \\ =& f\left(x_1^{(1)}w_{11}^{(1)} + x_2^{(1)}w_{21}^{(1)} + b_1\right)w_{11}^{(2)} + f\left(x_1^{(1)}w_{12}^{(1)} + x_2^{(1)}w_{22}^{(1)} + b_2\right)w_{21}^{(2)} + \\ & f\left(x_1^{(1)}w_{13}^{(1)} + x_2^{(1)}w_{23}^{(1)} + b_3\right)w_{31}^{(2)} + f\left(x_1^{(1)}w_{14}^{(1)} + x_2^{(1)}w_{24}^{(1)} + b_4\right)w_{41}^{(2)} + b \\ =& y. \end{aligned} \tag{17}$$

Comparando as equações (6) e (17) podemos ver que chegamos ao mesmo resultado dos coeficientes da função $\mathfrak{F}(\mathbf{x})$ da rede neural em ambas.

Fixada a configuração da rede neural, o aprendizado de máquina determina os valores dos pesos e dos seus vieses de forma aleatória. Ao receber os dados de treino, estas informações passam pela rede neural e o resultado da máquina é comparado com o dado de saída real. A rede aprende o padrão destes dados e ajeita os valores dos seus parâmetros de forma a minimizar a diferença entre os valores reais e os preditos. O processo de refinar os valores dos seus parâmetros é feito através da retro propagação (*backpropagation*).

A retro propagação calcula a taxa de erro de um aprendizado de máquina e fornece esta informação à rede neural para que a máquina possa ajustar os pesos e os vieses em cada camada. Estas taxas de erros são calculadas através das derivadas parciais $\partial C/\partial\omega$ e $\partial C/\partial b$ em que C é a função de perda (*loss function*) da rede neural. Esta função calcula o erro da rede neural e pode ser definida de diversos modos pelo usuário. A tabela 2 mostra três funções de perda usadas em aprendizado de máquina.

O cálculo das taxas de erros ocorre a cada mudança dos valores dos parâmetros da rede neural. Através destas taxas, a rede neural consegue dizer se é necessário mudar os valores ou não. Como a função de perda depende da diferença entre o *output* e o valor real, quanto menores as taxas de erro, maior a acurácia do aprendizado de máquina. Porém, avaliar as taxas de erro que consideram toda a rede neural não ajuda a máquina a localizar quais são os valores a serem mudados. Por exemplo, a taxa de erro da rede neural pode ser alta devido a um neurônio específico que possui o valor do viés b muito alto para o dado treinado. Então, a retro propagação utiliza quatro equações, (20), (22), (23) e (24), para calcular os erros em cada camada e em cada neurônio para identificar onde é necessário mudar os valores dos parâmetros na rede neural.

A tabela 2 mostra 3 das funções de perda mais usadas. Nesta tabela temos $\mathfrak{F}^L(\boldsymbol{x}_i)$ que é o valor final calculado pelo aprendizado de máquina para o dado de entrada $\boldsymbol{x}_i$, como vimos na equação (17) e $y_(\boldsymbol{x}_i)$ é o valor real do dado de saída para um dado de entrada $\boldsymbol{x}_i$. Em uma rede neural perfeita teríamos $y_(\boldsymbol{x}) - \mathfrak{F}^L(\boldsymbol{x}) = 0$. n é o número de objetos dos dados iniciais.

A função do erro absoluto médio (*mean absolute error* - `MAE`) é normalmente usada em problemas de regressão. Esta função calcula a média da diferença entre o dado de saída

Tipos de funções de perda	
Nome	**Fórmula**
Erro médio absoluto	$C = \frac{1}{n}\sum_{i=1}^{n} \lvert y(\boldsymbol{x}_i) - \mathfrak{F}^L(\boldsymbol{x}_i)\rvert$
Erro médio quadrático	$C = \frac{1}{n}\sum_{i=1}^{n} \left(y(\boldsymbol{x}_i) - \mathfrak{F}^L(\boldsymbol{x}_i)\right)^2$
Perda de Huber	$C = \begin{cases} \frac{1}{2}(y(\boldsymbol{x}_i) - \mathfrak{F}^L(\boldsymbol{x}_i))^2 \; se \quad \lvert y_(\boldsymbol{x}_i) - h^L(\boldsymbol{x}_i)\rvert \leq \delta \\ \delta((y(\boldsymbol{x}_i) - \mathfrak{F}^L(\boldsymbol{x}_i)) - \frac{1}{2}\delta) \; se \quad \lvert y(\boldsymbol{x}_i) - \mathfrak{F}^L(\boldsymbol{x}_i)\rvert > \delta \end{cases}$

Tabela 2: Tabelas dos tipos de funções de perda mais comuns para redes neurais.

original e o dado de saída predito pelo aprendizado de máquina. Como esta função é linear, os valores fora do padrão (valores muito altos e destoantes da maioria) não acrescentam muito à média final dos erros.

A função do erro quadrático médio (*mean squared error* - `MSE`) também é usado em aprendizado de máquina de regressão. Esta função calcula a diferença quadrática entre o dado de saída original e o predito pelo aprendizado de máquina. Pelo fato de a diferença ser elevada ao quadrado, os valores dos erros fora do padrão se sobressaem na equação e aumentam consideravelmente o valor da média. Esta função é útil quando se acredita que os *inputs* da máquina estejam perto de um valor médio e o usuário da máquina quer penalizar o mecanismo se houver muitos valores fora do padrão. Para isso, a máquina vai calcular o erro quadrático médio. Se estiver com um valor acima do que o usuário considera limite para o mecanismo, o usuário terá de mudar a sua configuração e os valores dos seus parâmetros para ajeitar a máquina de modo a obter um dado de saída mais parecido com o dado de saída original.

A função de perda de Huber (*Huber loss*) é uma combinação do erro absoluto médio com o erro quadrático médio [38, 39, 40]. O termo δ é um valor real unidimensional que é definido pelo usuário como o limite dos erros. Para os valores da diferença entre os dados de saída menores que δ, a função calcula o erro quadrático médio. Para valores de erros fora deste intervalo, a função calcula o erro absoluto médio. A função é sensível a valores fora da padrão apenas dentro do intervalo definido pelo usuário. Esta função é usada quando não queremos que os valores fora do padrão sejam considerados apenas para valores de erros menores (dentro do intervalo definido por δ) e para valores maiores, a função não é sensível a estes valores.

Para exemplificar como funciona o algoritimo da retro propagação, vamos considerar a função de perda do erro quadrático médio mostrada na tabela 2,

$$C = \frac{1}{n} C_{x_i} = \frac{1}{n} \sum_{i=1}^{n} \left(y_(\boldsymbol{x}_i) - \mathfrak{F}^L(\boldsymbol{x}_i) \right)^2 \quad . \tag{18}$$

A retro propagação utiliza a função de perda para calcular as taxas de erros através das derivadas parciais $\partial C/\partial \omega$ e $\partial C/\partial b$ [30]. Há um conjunto de quatro equações que definem a retro propagação e que serão mostradas a seguir nesta seção.

O algoritmo calcula o quanto variam os resultados da rede neural ao se alterar os valores dos pesos e dos vieses no aprendizado de máquina. A diferença entre o valor ideal e o valor calculado em um neurônio j na camada l é denominado erro δ_j^l. A retro propagação nos ajudará a calcular o erro δ_j^l e relacioná-lo com $\partial C/\partial \omega_{jk}^l$ e $\partial C/\partial b_j^l$.

Para entender como funciona o erro δ_j^l, vamos supor que o dado, após passar pelo neurônio j seja $f(z_j^l)$, em que f é a função de ativação do neurônio e z_j^l é o dado de entrada do neurônio já com o peso. Porém, a rede não está bem treinada e adiciona um termo Δz_j^l a esse peso. Logo, o valor final do dado de saída no neurônio j na camada l é dado por $f(z_j^l + \Delta z_j^l)$. Esta mudança é propagada ao longo da rede de modo que custará à máquina o erro de $\frac{\partial C}{\partial z_j^l} \Delta z_j^l$

O objetivo de uma rede bem treinada é minimizar este valor da função de perda. Logo, se $\partial C/\partial z_j^l$ for muito grande, o termo Δz_j^l pode ajudar a diminuir este valor. Se $\partial C/\partial z_j^l$ for pequeno, o termo Δz_j^l não será de muita utilidade. Logo, $\partial C/\partial z_j^l$ é a medida do erro em um neurônio. Podemos definir a medida do erro em um neurônio j na camada l pela equação

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \quad . \tag{19}$$

Com isso, $\boldsymbol{\delta}^l$ será o vetor dos erros associados à camada l. A retro propagação permitirá o calculo dos erros δ^l para cada camada e, então, relacionar estes erros com as quantidades de interesse: $\partial C/\partial \omega_{jk}^l$ e $\partial C/\partial b_j^l$. Para isso, são utilizadas quatro equações da retro propagação.

A primeira equação da retro propagação é o cálculo do erro δ_j^H da última camada da rede neural, definida como

$$\delta_j^H = \frac{\partial C}{\partial \mathfrak{F}^{jH}} f'(z^{jH}). \tag{20}$$

A equação (20) é a equação do erro na camada de saída da rede neural. O termo H é o número de camadas na rede neural, no caso do nosso exemplo de rede neural usamos $H = 3$ (camada de entrada, camada intermediária e camada de saída). A taxa de erro

$\partial C/\partial \mathfrak{F}_J^H$ mede a variação da função de perda em função do *output* j da rede neural. A derivada $f'(z_j^H)$ mede o quão rápido a função de ativação f muda em z_J^H.

Podemos reescrever a equação (20) na forma matricial,

$$\delta^H = \nabla_a C \odot f'(z^H), \tag{21}$$

em que $\odot$ é a multiplicação de Hadamard entre duas matrizes. $\nabla_a C$ é um vetor cujos componentes são as derivadas parciais $\partial C/\partial \mathfrak{F}_j^H$, ou seja, é a taxa da variação de C considerando os dados de saída da rede neural.

A segunda equação,

$$\delta^l = \left((\omega^{l+1})^T \delta^{l+1}\right) \odot f'(z^l), \tag{22}$$

calcula o erro δ^l em termos do erro da próxima camada δ^{l+1}. $(\omega^{l+1})^T$ é a transposta da matriz ω^{l+1} para a camada $l+1$. Isto significa que, se soubermos o erro δ^{l+1}, ao aplicarmos a transposta de ω^{l+1}, estamos movendo o erro para trás através da rede neural, nos dando o valor do erro na camada anterior. A multiplicação escalar $f'(z^l)$ move o erro para trás através da função de ativação na camada l, dando-nos o erro δ^l na camada l.

Combinando as equações (20) e (22), podemos calcular o erro δ^l para qualquer camada l da rede. A equação

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \rightarrow \frac{\partial C}{\partial b} = \delta \tag{23}$$

calcula o erro em um neurônio j. O erro δ_j^l é igual à taxa de mudança $\partial C/\partial b_j^l$. A equação (23) nos dá a taxa de mudança da função perda considerando o viés do neurônio. A diferença principal entre esta equação e as equações (20) e (22) é que a equação (23) calcula o erro de apenas um neurônio da camada, as outras equações calculam o erro total das camadas (considerando todos os neurônios).

A equação

$$\frac{\partial C}{\partial \omega_{jk}^l} = \mathfrak{F}_k^{l-1} \delta_j^l \rightarrow \frac{\partial C}{\partial \omega} = \mathfrak{F}_{entrada} \delta_{saida} \tag{24}$$

calcula a derivadas parciais $\partial C/\partial \omega_{jk}^l$ em função de $\mathfrak{F}_k^{l-1}$ e δ_j^l.

Pela equação (24) podemos ver que se $\mathfrak{F}_k^{l-1}\delta_j^l \approx 0$, o gradiente $\partial C/\partial \omega$ também será pequeno. Um gradiente pequeno significa que o peso aprende de forma devagar. Quando isso acontece, podemos dizer que o neurônio foi saturado e tanto o peso quanto o viés não variam, ou seja, a rede para de aprender os padrões ou aprende devagar.

2.2. Cosmologia

A cosmologia é a área da astronomia que estuda a origem e a evolução do universo desde o Big Bang até os dias atuais. É o estudo científico de propriedades de grandes escalas do universo como um todo. Este campo da ciência começou como uma subárea da física teórica no início do século XX e se desenvolveu em uma das áreas mais ativas da atualidade [41, 42, 43]. Isto aconteceu em grande parte devido à aplicação de suas descobertas na física atômica e na física nuclear, além da grande quantidade de dados provenientes dos telescópios que fazem observações em todo o espectro eletromagnético [44, 45].

Na cosmologia considera-se o princípio cosmológico, cujo pressuposto é que, para grandes escalas, o universo é homogêneo e isotrópico. Assim sendo, ser homogêneo significa que a parte que conseguimos observar do universo é uma amostra representativa do universo. E ser isotrópico significa que as mesmas leis da física que se aplicam para um determinado ângulo de observação, podem ser aplicadas para todo o universo. A hipótese do princípio cosmológico é apoiada por diversas observações, como os fótons na radiação cósmica de fundo em micro-ondas que advêm de diferentes partes do universo com praticamente a mesma temperatura [46, 47].

Einstein (14/03/1879 - 18/04/1955) associou o princípio às ideias de Ernst Mach (18/02/1838 - 19/02/1916), que diz que as leis da física são determinadas pela distribuição da matéria em largas escalas. A partir disso, foi possível pensar em modelos cosmológicos relativamente simples [45, 44].

No início do século XX, Vesto Slipher (01/11/1875 - 08/11/1969) descobriu que as luzes de galáxias distantes estavam deslocadas para a área vermelha do espectro ("redshift") [48]. Algum tempo depois, isso foi interpretado como galáxias se afastando da Terra. Dez anos mais tarde, Alexander Friedmann (16/06/1888 - 16/09/1925) utilizou as equações de Einstein de Relatividade Geral para mostrar provas teóricas de que o universo está se expandindo [49]. O astrônomo Knut Lundmark (14/06/1889 - 23/04/1958) foi a primeira pessoa a descobrir evidência observacional para a expansão em 1924 [50]. Edwin Hubble (20/11/1889 - 28/09/1953) confirmou essas observações cinco anos depois. Tudo isso confirmou que as galáxias estão se afastando umas das outras, logo o universo está se expandindo.

2.2.1. Expansão do universo

A expansão do universo nos diz que a distância entre os objetos astronômicos era menor em tempos passados do que nos dias de hoje. Para quantificar esse efeito, é usado o fator de escala a, cujo valor atual é igual a 1, mas, na época no início do universo, era bem menor, na ordem de 10^{-6}. Uma forma mais fácil de ver essa representação é através da figura 9 que mostra como é a relação entre a distância comóvel e o fator de escala a. A

distância entre dois pontos no sistema continua sendo igual a 1, o que varia é o valor de a, que aumenta com o tempo. Enquanto a distância física entre esses dois tempos aumenta com o tempo também.

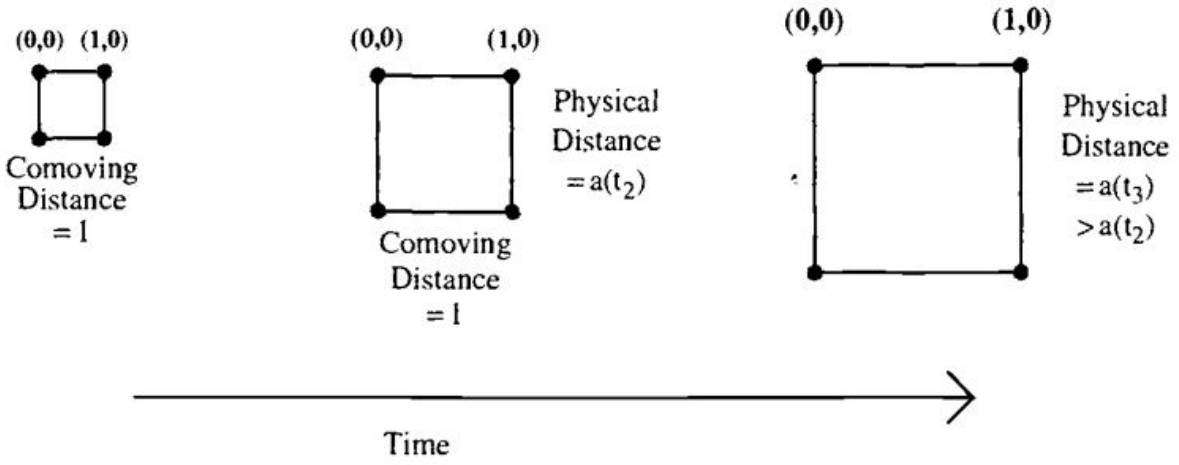


Figura 9: Representação do aumento da distância física em relação à distância no sistema de dois pontos com relação ao tempo e proporcional ao fator de escala a. Figura retirada do livro de Scott Dodelson (2003) [45]

A consequência direta é que o comprimento de onda de uma luz emitida por um objeto distante é transladada no espectro de potência para o vermelho ("*redshift*- z) de forma proporcional a a. A luz deslocada para a área vermelha do espectro significa aumento do comprimento de onda. Isso é similar ao efeito Doppler, quando um objeto se afasta do observador e vai para a parte mais avermelhada do espectro eletromagnético, diminuindo o valor da frequência. Este fenômeno é descrito por Scott Dodelson e Fabian Schmidt [45] como,

$$\frac{\lambda_{obs}}{\lambda_{def}} = \frac{a_{obs}}{a_{def}} = \frac{1}{a_{def}} = 1 + z \quad , \tag{25}$$

em que o λ_{obs} é o comprimento de onda da emissão ou absorção de um elemento no objeto astronômico observado desde a Terra e λ_{def} é o comprimento de onda desse mesmo elemento observado quando o objeto está em repouso. Além disso, $a_{obs} = a_{atual} = 1$ como definido anteriormente. Temos, portanto, a definição de um parâmetro chamado redshift (z). Se o objeto estivesse se aproximando, isso implicaria no comprimento de onda se deslocando para a área mais azul do espectro eletromagnético, chamado de *blueshift*, que implica em z negativo. A equação (25) traz a definição cosmológica do redshift.

2.2.2. Formas de obter o redshift

A observação do redshift é feita através da espectroscopia[4]. Através deste espectro, podemos ver as linhas de emissão e absorção do objeto observado e comparar com o espectro dos mesmos elementos quando observados em laboratórios na Terra (fixos). Ao compararmos esses comprimento de onda fixos (λ_{def}) com os que são vistos no espectro do objeto astronômico (λ_{obs}), conseguimos ver se houve deslocamento do comprimento de onda na linha do espectro. Se o objeto está vindo na nossa direção, vemos esse deslocamento da linhas de espectro se mover para o azul ("*blueshift*"), enquanto que, se o objeto estiver se distanciando de nós, vemos o espectro mais avermelhado. A partir da equação (25) temos o valor do redshift,

$$z_{spec} = \frac{\lambda_{obs} - \lambda_{def}}{\lambda_{def}} \tag{26}$$

em que o λ_{obs} é o comprimento de onda observado de um elemento no objeto astronômico através da espectroscopia e o λ_{def} é o comprimento de onda desse mesmo elemento no referencial do laboratório da Terra [51]. O redshift z_{spec} dado pela espectroscopia nos diz quanto tempo demorou para a luz do objeto alcançar a Terra. Este redshift é definido de forma que, na época do Big Bang, o seu valor seria infinito e nos dias atuais temos $z = 0$.

É possível ver uma representação clara disso na figura 10, que reúne o espectro de estrelas e galáxias. Podemos ver que, com o aumento do redshift, os comprimentos de onda de emissão e absorção dos elementos estão se deslocando para o vermelho. Considerando que um z maior significa que a luz do objeto precisou percorrer uma maior caminho, isso significa que o objeto está mais longe do referencial (Terra). Logo, temos que, para objetos astronômicos com redshift maior, o deslocamento do comprimento de onda é maior.

Essas figuras do espectro foram feitas pelo Sloan Digital Sky Survey[5] (`SDSS`), que é um levantamento astronômico que utiliza um telescópio ótico para observar os objetos [52]. É um projeto que já tem mais de 20 anos de duração, com resultados disponíveis na internet através de um servidor[6].

Esses gráficos foram feitos a partir da espectroscopia, que é medir o espectro de radiação eletromagnética. Essa técnica é utilizada também para estudar as propriedades físicas de diversos objetos astronômicos. Esse modo de calcular o redshift consome muito tempo e, para objetos mais distantes, a espectroscopia não consegue observar as suas linhas do espectro. A razão disto é porque para objetos mais distantes, há a possibilidade de interferência na medição causada por diversos motivos, como poeira, outro objeto astronômico muito perto do objetivo, detritos espaciais no caminho da medição, entre outras causas.

[4]Espectroscopia é o processo de medir a absorção ou emissão da luz no espectro eletromagnético.
[5]`https://classic.sdss.org/`
[6]`http://skyserver.sdss.org/dr10/en/tools/toolshome.aspx`

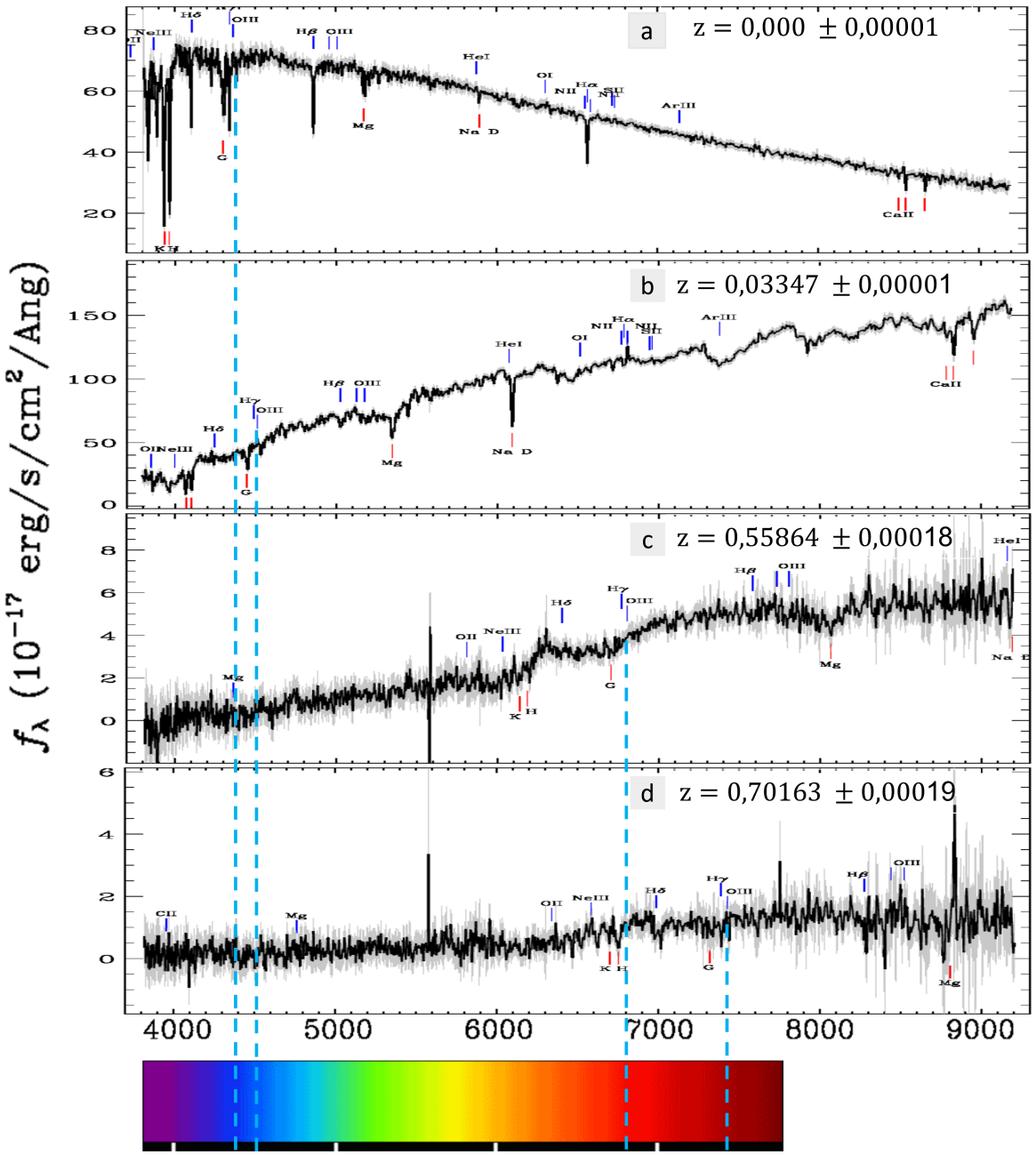


Figura 10: Demonstração de como o comprimento de onda de um elemento se desloca para o vermelho com o aumento do redshift. Nessa figura foi destacado o comprimento de onda da emissão do oxigênio duplamente ionizado (`O III`), todos observados pelo `SDSS`. Na figura foram agrupados diferentes espectros de objetos astronômicos em função do fluxo de densidade do espectro. a) Estrela (id:`J112149.55+264535.3`). b) Galáxia (id:`J162009.52+374753.5`), cujo nome comum é AT 2019riy. c) Galáxia (id:`J120923.68+264046.7`). d) Galáxia (id:`J144344.63+532751.1`).

Além disso, para surveys com uma grande área de cobertura, é impossível observar a espectroscopia de todos os objetos, especialmente para objetos com alto redshift [53], devido a restrição de tempo do telescópio. Nesses casos, são utilizadas as observações da

luminosidade dos objetos através de diversos filtros para calcular o redshift fotométrico (z_{phot}), isto é, utilizando a fotometria (magnitude e cores) do objeto. Em nosso trabalho, vamos calcular o redshift fotométrico das galáxias. Essa propriedade utiliza a fotometria dos objetos, nesse caso, as magnitudes.

A magnitude é uma medida, sem dimensão, do brilho de um objeto astronômico em um determinado filtro. Este filtro corresponde a um intervalo de comprimento de ondas do qual o sinal passa apenas se estiver nesse mesmo intervalo. Qualquer sinal que estiver fora dessa faixa (ou intervalo) não será observado e a partir deste filtro é possível calcular a magnitude.

Neste trabalho utilizamos catálogos com sistema de magnitude AB, consistindo na utilização do valor do fluxo de densidade do espectro (como visto na figura 10). Esse sistema utiliza o logaritmo do fluxo sobre uma constante em Jankys, que é uma unidade para fluxos de densidade dada por $1J_y = 10^{-23} ergs^{-1} Hz^{-1} cm^{-2}$. Pela figura 10 podemos ver que as unidades dos valores dentro do logaritmo se cancelam e, dessa forma, a magnitude vira um valor sem dimensão, como podemos ver na equação (1).

2.2.3. Redshift fotométrico

Redshifts fotométricos são redshifts calculados utilizando a fotometria de objetos astronômicos (tais como galáxias) no lugar da espectroscopia. A acurácia do redshift fotométrico é menor que a do redshift espectroscópico, pois depende do conjunto de filtros e da precisão fotométrica do telescópio. Em contrapartida, a partir de uma amostra de redshifts espectroscópicos para galáxias, é possível obter os redshifts fotométricos de grandes conjuntos de galáxias de forma mais rápida. Isso implica que, para grande parte das aplicações cosmológicas, o redshift fotométrico, apesar de não ter a mesma acurácia do redshift espectroscópico, pode ser utilizado.

Há duas formas de se calcular o redshift fotométrico, seja a partir da utilização da distribuição de energia espectral (*Spectral Energy Distribution* - `SED`) ou a partir do modelo empírico.

O primeiro método é o mais antigo, desenvolvido na década de 60 do século passado. Em nosso trabalho, utilizamos a segunda forma, pois é a mais rápida para grandes amostras de galáxias e nos permite controlar o processo de modo a obter a maior acurácia possível.

Distribuição de Energia Espectral Esta técnica foi desenvolvida por Baum, em 1962 [54], a partir da utilização de um fotômetro fotoelétrico. Neste instrumento, a luz observada pelo telescópio passa por um filtro e deste para uma superfície sensível à luz (catodo). O catodo emite os elétrons, que são multiplicados dentro de um fotomultiplicador, permitindo que o sinal seja facilmente medido. Assim, cada fóton detectado pelo catodo gera um pulso, sendo que o número de pulsos por segundo é diretamente proporcional ao brilho

do objeto. Na fotometria de alta velocidade, os pulsos são contados em intervalos de 10 mili segundos e os melhores fotômetros podem medir magnitudes com acurácia chegando em 0,001 mag [55].

Além do fotômetro fotoelétrico, Baum utilizou 9 filtros no intervalo de 3730Å até 9875Å. Com este sistema, Baum observou a distribuição de energia espectral (`SED`) de 6 galáxias elípticas do Aglomerado de Virgem e 3 outras galáxias com a mesma morfologia do aglomerado de Coma. A partir destas informações, Baum calculou a média da `SED` destes objetos e colocou as duas distribuições no mesmo gráfico, utilizando uma escala logarítmica. Com isso, Baum conseguiu medir o deslocamento entre as duas distribuições de energia, e consequentemente, o redshift do segundo aglomerado. Baum calculou o redshift fotométrico deste aglomerado como $z = 0,19$ sendo que o seu valor espectroscópico é $z = 0,192$, ou seja, um erro de pouco mais de 1% [56].

Baum ainda estendeu esta técnica para outros aglomerados, cujos redshifts eram desconhecidos. A técnica de Baum tinha bastante acurácia, porém, por depender da descontinuidade de 4000Å (era o ponto guia para comparar as duas `SED`s), só poderia funcionar para galáxias elípticas.

Em 1985, Koo [57] tentou uma abordagem diferente ao utilizar placas fotométricas no lugar de um fotômetro. As placas fotométricas são suportes fotográficos que têm uma lâmina de vidro coberta por uma emulsão sensível a luz. Além disso, utilizou apenas 4 filtros e o diagrama cor-cor para calcular o redshift. Koo trabalhou com uma amostra de 100 galáxias com redshift no intervalo de $z = 0,025$ e $z = 0,700$.

O método do cálculo do redshift fotométrico feito pela `SED` é baseado no ajuste da forma do espectro eletromagnético do objeto astronômico e na detecção de propriedades espectrais fortes. Para obter resultados mais precisos, é interessante escolher um conjunto de filtros que possa observar certas propriedades como a descontinuidade de 4000Å, a descontinuidade de Balmer ou a descontinuidade de Lyman. Em suma, é necessário que o intervalo de comprimento de onda dos filtros esteja localizado em espaço que tenha as linhas de emissão fortes, que são fáceis de reconhecer. Isso porque, em geral, filtros de banda larga (*broad-band*) detectam apenas descontinuidades e não são sensíveis à presença de linhas de emissão.

O ajuste da `SED` usa o processo de minimizar χ^2, a diferença entre média das `SED`s observadas de uma galáxia cujo redshift espectroscópico é conhecido e um conjunto de amostras de galáxias do espectro cujos redshifts ainda serão calculados. χ^2 é calculado através da equação

$$\chi^2(z) = \sum_{i=1}^{N_{filters}} \left[\frac{F_{obs,i} - b \times F_{temp,i}(z)}{\sigma_i} \right]^2, \tag{27}$$

em que $F_{obs,i}$ é a média do fluxo observado das galáxias do conjunto de treinamento, $F_{temp,i}$ são os fluxos da amostra para o cálculo do redshift fotométrico, σ_i é a incerteza no filtro i e b é a constante de normalização.

Através da equação (27), calcula-se o fluxo de um conjunto de amostras de galáxias para cada filtro. Para a galáxia cujo redshift queremos descobrir, são colocados os fluxos de cada filtro na equação (27) e então é feito o ajuste até obtermos o menor possível χ^2. Então os valores de $F_{temp,i}$ (que é a única variável da equação) são modificados e testados até que seja obtido o menor valor de χ^2. A partir deste valor da minimização, é possível calcular o redshift fotométrico, já que teremos o espectro do objeto calculado pela fotometria.

Mais tarde nos anos 90, o interesse no cálculo do redshift fotométrico aumentou com o desenvolvimento de catálogos astronômicos de campos profundos, como o Hubble Deep Field (`HDF`). Connolly et al. propuseram em 1996 [58] uma forma de obter o redshift criando uma relação empírica entre magnitudes e redshifts. Para isso, Connolly et al. utilizaram um conjunto de amostras cujos valores de redshift espectroscópicos eram conhecidos. Nos anos seguintes, esta técnica de comparar os gráficos das galáxias foi utilizada por Pello et al. em 1996 [59], por Miralles et al em 1997 [60] e por Bruzual e Cahrlot em 1993 [61], entre outros.

As principais vantagens de se utilizar um modelo empírico em relação ao ajuste da `SED`, é a facilidade no cálculo do redshift fotométrico para grandes conjuntos de objetos astronômicos, uma precisão estatística maior e flexibilidade. Enquanto o ajuste da `SED` é feito de forma individual para cada objeto, ao criarmos um modelo com a relação entre as magnitudes e a fotometria e fixarmos os seus parâmetros, podemos colocar diversos dados de entrada e obter o redshift fotométrico rapidamente. A precisão estatística do modelo empírico se dá pela possibilidade de tornar o modelo mais complexo, de forma a obter o z_{phot}, enquanto o ajuste da `SED` é apenas uma fórmula que é constante e simples. Por fim, o ajuste da `SED` funciona apenas para o conjunto de objetos que estão no mesmo intervalo do espectro, para qualquer outro objeto no mesmo espectro não será possível conseguir calcular a distribuição da energia. O modelo empírico permite que objetos com fotometrias diferentes tenha o seu z_{phot} calculado, por isso é o método mais flexível.

Modelos Empiricos Neste trabalho, utilizamos o segundo método, o de modelos empíricos. Este método busca criar uma relação matemática entre a fotometria e a magnitude [62, 63].

No exemplo dado na seção 2.1.3, temos a equação (17), em que são colocados *inputs* na rede neural e temos como resposta y. Se fossemos usar esta rede neural para calcular o redshift fotométrico, usaríamos como dado de entrada as magnitudes.

Pela equação

$$\mathbf{x} = \begin{bmatrix} m_g^{(1)} \\ m_h^{(1)} \end{bmatrix} \tag{28}$$

temos o um dado de entrada $\mathbf{x}$ cujos componentes são as magnitudes de uma galáxia nos filtros g e h.

Poderíamos então reescrever a equação (17), do modelo empírico entre as magnitudes e o redshift fotométrico da forma

$$\begin{aligned} \mathfrak{F}(\mathbf{x}) =& f\big(m_g^{(1)} w_{11}^{(1)} + m_h^{(1)} w_{21}^{(1)} + b_1\big) w_{11}^{(2)} + f\big(m_g^{(1)} w_{12}^{(1)} + m_h^{(1)} w_{22}^{(1)} + b_2\big) w_{21}^{(2)} + \\ & f\big(m_g^{(1)} w_{13}^{(1)} + m_h^{(1)} w_{23}^{(1)} + b_3\big) w_{31}^{(2)} + f\big(m_g^{(1)} w_{14}^{(1)} + m_h^{(1)} w_{24}^{(1)} + b_4\big) w_{41}^{(2)} = z_{phot}. \end{aligned} \tag{29}$$

A rede neural construída neste trabalho utilizará o mesmo princípio da equação (29) porém com mais nós na camada intermediária.

2.3. Catálogos astronômicos

Catálogos astronômicos (*surveys*) são mapeamentos de uma região do céu. Um catálogo contém informações sobre observações de objetos astronômicos em determinada área. Estas informações podem ser de fotometria, de espectroscopia ou rádio e são obtidas através de telescópios.

2.3.1. VIPERS

O VIMOS Public Extragalactic Redshift Survey (VIPERS) [13] é um programa que tem como objetivo mapear com detalhes a distribuição de galáxias com redshift entre 0,5 e 1,2. Para isso, mede os aglomerados de galáxias, a estrutura de crescimento e as propriedades de galáxias em um período em que o universo tinha metade da sua idade atual. O alvo das amostras das galáxias vem dos dados fotométricos de 5 filtros do Canada-France-Hawaii Telescope Legacy Survey Wide catalogue (CFHTLS-Wide).

O VIsible Multi-Object Spectrograph (VIMOS) é um espectrógrafo visível de Múltiplos Objetos localizado no Observatório Europeu (ESO) que contém diversas informações sobre as galáxias e, entre elas, o redshift (z). Um mapa das coordenadas mapeadas pelo catálogo do `VIPERS` pode ser vista na figura 11.

O `VIPERS` foi feito para conseguir amostras de objetos com redshift perto de 0,7. Para obter qualidade das amostras de espectro em um tempo de exposição curto no espectrógrafo do VIMOS, foi adotado um limite para a magnitude do filtro `i` de $m_i = 22,5$. Isso gerou dois problemas para uma seleção de amostras eficiente. Com essa limitação, muitas galáxias estariam fora da restrição proposta pelo `VIPERS` de ter redshifts fotométricos entre 0,5, e 1,2. Além disso, através de estudos passados [64][65], sabe-se que limitar

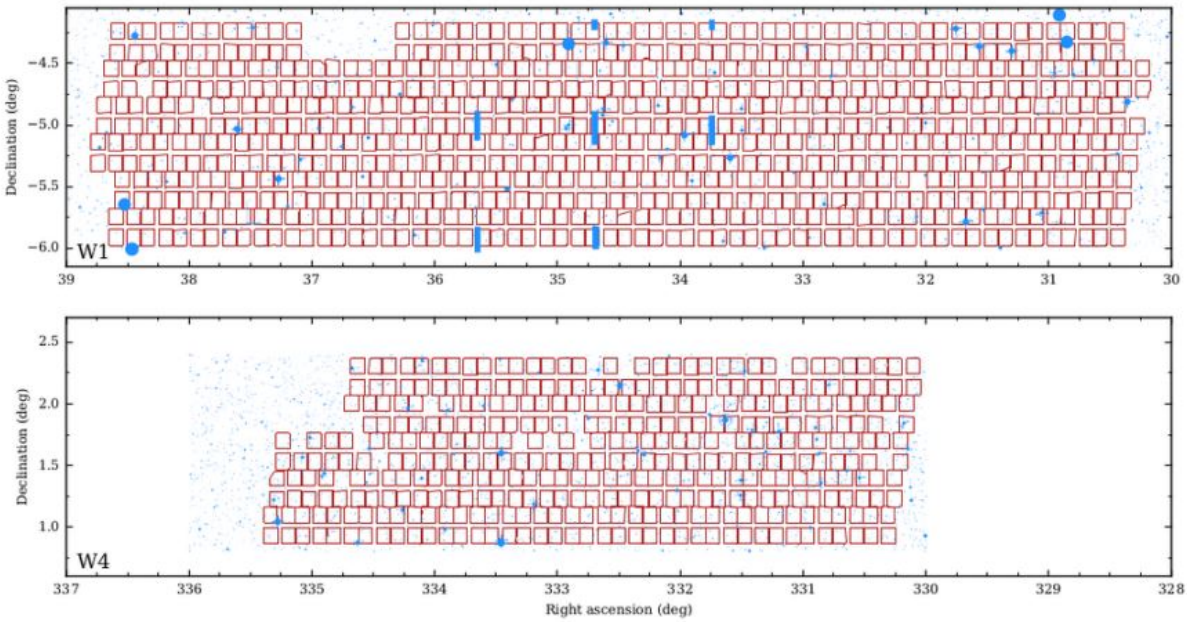


Figura 11: Figura da área mapeada pelo catálogo astronômico `VIPERS`. Os contornos dos quadrados vermelhos são os pontos observados pelo telescópio VIMOS. Os pontos azuis correspondem a regiões em que as observações estão corrompidas ou não foram possíveis de serem feitas devido à presença de objetos estranhos, tais como estrelas muito brilhantes.

as amostras somente pelo valor da magnitude pode causar até 30% de contaminação de estrelas. Isso significa que, em uma amostra de galáxias, haveria uma porção considerável de estrelas misturadas consideradas galáxias.

Para se fazer a seleção de galáxias e estrelas seria preciso classificar cada objeto em relação a sua propriedade morfológica. Para isso, foram usados os dados fotométricos do CFHTLS combinados com a informação espectroscópica dos catálogos do `VVDS-DEEP` e `VVDS-Wide`. Os dados fotométricos do CFHTLS são bastante precisos já que foram planejados para lentes gravitacionais, o que facilita a identificação de objetos pontuais. Isso permite fazer a seleção de galáxias e estrelas com precisão e usar o tempo de telescópio de forma eficiente. O primeiro passo é criar um critério para a seleção dos objetos.

Para obter o melhor sistema de seleção, usamos os catálogos `VVDS-DEEP` e `VVDS-Wide`. O catálogo `VVDS-DEEP` fornece redshifts de mais de 10 mil galáxias, AGN e estrelas com $m_i \leq 24$. O catálogo `VVDS-Wide` inclui o espectro de mais de 11 mil galáxias e 7 mil estrelas com $m_i = 22.5$ [66]. Ambos os catálogos são limitados pela magnitude, por isso, representam um conjunto de treinamento ideal para testar a completeza e contaminação da função de seleção. Para isso, são utilizados os objetos da ambos os catálogos que são classificados com os números 3 e 4. Um objeto identificado como 4 é um objeto com grande confiança nos seus dados, ou seja, certeza no redshift cujos valores são confirmados pelas propriedades espectrais. Os objetos de nível 3 também têm bastante confiança em seus valores graças às propriedades do espectro. O grau de confiança de objetos sinalizados

como 3 e 4 é de 99%. Objetos classificados com outros números não apresentam tanta confiança em seus valores quanto os citados anteriormente.

O método para classificar as estrelas e as galáxias do VIPERS combina o conhecimento do tamanho do objeto (raio de meio fluxo) com o ajuste na distribuição de energia espectral (*spectral energy distributions* - SED), obtida ajustando os modelos nos 5 filtros disponíveis. O raio de meio fluxo (r_h) é o raio de um círculo ao redor do objeto em que metade do seu fluxo está contida. A distribuição de energia espectral é o fluxo da energia em função da frequência ou comprimento de onda do objeto astronômico, e esta distribuição é feita através da observação espectroscópica de um objeto. A partir de um catálogo de objetos astronômicos de mesma morfologia, é possível criar um modelo de SED, este modelo torna-se referência para os outros objetos de mesma morfologia. O ajuste da SED é a técnica de adaptar a distribuição de outros objetos na curva criada pelo modelo e, a partir desta curva, obter uma ou mais propriedades físicas destes objetos [67]. Os dados obtidos do telescópio sugeriram que o tamanho do objeto astronômico deveria ser o critério prioritário na divisão dos objetos.

Os objetos astronômicos foram divididos em dois grupos, de acordo com a sua luminosidade na banda do filtro i (m_i). O primeiro grupo foi composto de objetos com $17,5 < m_i < 21$ e o segundo grupo de objetos com $m_i > 21$. Para o conjunto mais brilhante (menor valor absoluto de magnitude), foi utilizado apenas um critério, o do raio de meio fluxo, enquanto que, para as galáxias menos brilhantes, foi preciso utilizar os dois critérios citados anteriormente.

Para o primeiro grupo, e utilizando o r_h como fator de divisão, foram selecionados os objetos com $17,5 < m_i < 21$, que é a região onde as estrelas são dominantes. Desta propriedade, foi estabelecida uma gaussiana na distribuição do raio de meio fluxo. Essa distribuição estatística dos tamanhos dos objetos pode ser descrita pela sua média μ_{r_h} e o seu desvio padrão σ_{r_h}. Foi definido em Guzzo et al. [68] que as estrelas seriam os objetos que obedecem a inequação $r_h < \mu_{r_h} + 3\sigma_{r_h}$.

Este critério é bom para galáxias mais brilhantes. Para as menos brilhantes, $m_i > 21$, pequenas galáxias poderiam ser confundidas com estrelas e classificadas erroneamente. Para evitar isso, foi adicionada a informação fornecida pelo ajuste da distribuição de energia espectral dos objetos. Isto é obtido ao ajustar os cinco filtros com o código para o redshift fotométrico Le phare. O código Le Phare é um programa na linguagem de programação fortran para calcular o redshift fotométrico usando o método de ajustar os dados de entradas (magnitudes) em uma equação cujo resultado será o redshift fotométrico [69] [70]. Assim, foi encontrado o melhor ajuste para classificar os objetos. Este método é limitado por conta da degenerescência das cores de algumas estrelas, o que causa uma contaminação na amostra. Por conta disto, este método não pode ser usado sozinho. Então, para galáxias menos brilhantes, há a combinação dos dois métodos: o ajuste da SED e o uso do raio de meio fluxo r_h.

Por fim, cerca de 21% dos objetos no catálogo foram removidos por serem classificados como estrelas, 32% foram removidos porque foram classificados como galáxias com redshift baixo, e assim os 47% restante se tornaram o catálogo do `VIPERS`.

2.3.2. DES

A fim de obtermos apenas galáxias do `DES` e pegarmos as magnitudes na banda `i` que estão perto fotometricamente do VIPERS, utilizamos uma query para baixar apenas os dados necessários de cada pixel, já que os dados do `DES` estão todos localizados em diferentes tabelas divididas pelos pixeis. As magnitudes na banda `i` são os valores calculados pela equação (1) que estão limitados aos comprimento de onda $7000\hat{A} < i < 8600\hat{A}$. Query é um pedido de informação da base de dados, podendo essa ser apenas uma tabela ou a combinação delas, tal como no DES. Uma query pode tanto ser uma requisição de resultados da base de dados, quanto uma ação a ser feita com os dados. A query pode fazer diversos cálculos como somar, subtrair e até contas mais complexas, pode combinar as tabelas, pode mudar dados ou deletar dados [71]. Há diversas linguagens para escrever uma query, no caso do `DES`, é utilizada a linguagem chamada Structured Query Language (SQL). Essa linguagem permite acessar uma grande base de dados com apenas um comando; além disso, retorna os dados em tabelas, que é o formato que precisamos usar em nossos programas.

Os dados consistem em imagens captadas por um telescópio localizado no Observatório Interamericano de Cerro Tololo, no Chile, em 345 noites (durante um período de 3 anos). Um novo artigo com a inclusão de mais dados, coletados em um período de 6 anos, foi publicado no início de 2022 [12].

O conjunto desses dados possui inúmeras informações como Ascensão Reta, a Declinação e o identificador de cada galáxia, denominados no catálogo como RA, DEC e ID. O identificador de uma galáxia é um número, que pode ter entre 8 ou 9 dígitos, que é diferente para cada objeto e o distingue de todos os outros; como se fosse o nome da galáxia, mas de forma fácil para guardar esses dados de mais de 6 milhões de galáxias. Até agora nós focamos nas magnitudes das galáxias, que medem justamente suas luminosidades. Estas galáxias foram sujeitas a cinco filtros fotométricos, em que cada um limita todas as cores vindo das galáxias exceto por uma. Nos dados liberados pelo `DES` os filtros utilizados são: `g, r, i, z` e m_Y. Cada filtro ou banda é uma limitação da luz em determinados comprimentos de onda, e a figura 1 mostra os limites de cada uma dessas bandas. Um mapa das coordenadas mapeadas pelo catálogo do `DES` pode ser visto na figura 12.

O conjunto estudado do `DES` tem cerca de 300 milhões de galáxias e o do `VIPERS` tem cerca de 90 mil e essa diferença se deve por duas razões: a primeira é que o telescópio que coletou os dados do `DES` observou um espaço físico maior do universo; e a segunda é que há um limite do quanto cada telescópio do `VIPERS` consegue distinguir de galáxias. Portanto, apesar de termos um espaço com muitas galáxias, o telescópio só consegue captar um

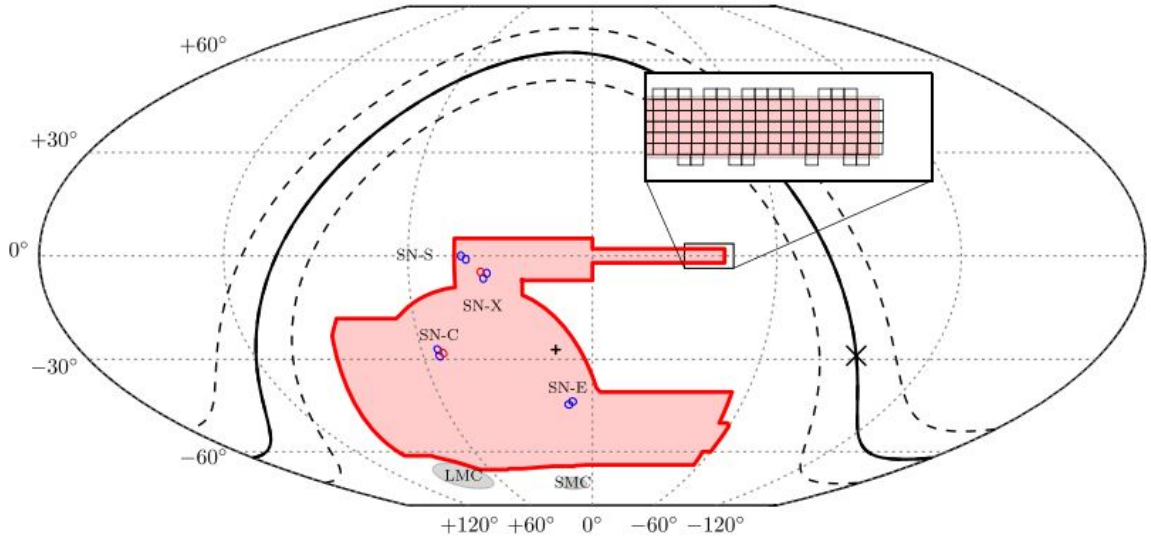


Figura 12: Figura da área mapeada pelo catálogo astronômico `DES`. A área observada pelo `DES` está com a cor vermelha no mapa. Os círculos são regiões com supernovas, sendo que as de cor azul são supernovas mais próximas e as de cor vermelha estão mais distantes. A Via Láctea é mostrada através de uma linha sólida com linhas tracejadas identificando $\pm$ 10º em relação à galáxia. O centro da galáxia está marcado com "×". O polo sul galático, que é o ponto mais ao sul da Via Láctea, com "+". As nuvens de Magalhães estão identificadas nas regiões cinzas, sendo que a escrita SMC (*Small Magellanic Cloud*) se refere à pequena nuvem de Magalhães e a LMC (*Large Magellanic Cloud*) refere-se à grande nuvem de Magalhães.

número de galáxias menor. Isso significa que há outras galáxias com luminosidade, e, consequentemente, cor, parecidas com as do `VIPERS`, mas que não foi possível obter o redshift.

Para selecionar os dados desejados do `DES`, foi preciso olhar quais colunas seriam úteis para fazer essa partição. As tabelas do `DES` têm 215 colunas, em que as primeiras são a identificação da galáxia, assim como o identificador do pixel em mapas com diferentes resoluções e as suas coordenadas RA e DEC. Após isso, estão as propriedades obtidas de cada objeto astronômico. Como o `VIPERS` tem uma restrição da magnitude na banda `i` de até 22,5 [13], fizemos também essa restrição. Porém, para aumentar a nossa possibilidade de cálculo e não ficar no limite dos valores da magnitude, escolhemos pegar objetos com magnitudes na banda `i` até 24. Para a separação das galáxias entre estrelas e galáxias, nós escolhemos utilizar a propriedade que classifica morfologicamente os objetos com uma acurácia de 99%. Os objetos podem ter a seguinte classificação: 0 grande confiabilidade de o objeto ser estrela, 1 confiabilidade de o objeto ser estrela, 2 provavelmente é galáxia, 3 grande confiabilidade de o objeto ser galáxia, -9 sem dados para classificar [12]. A fim de pegarmos apenas galáxias, restringimos os dados para objetos com valores de classificação maior que 2. O código escrito em `SQL` para baixar os dados está descrito no apêndice B.

3. Pré-processamento de dados

Antes de iniciar a fase de treino dos programas, foi preciso preparar os dados que seriam utilizados neste trabalho. Para o treinamento do aprendizado de máquina para o cálculo do redshift, só será preciso lidar com as galáxias do VIPERS. Para a estrutura de dados, será preciso trabalhar com as galáxias do DES em conjunto com as galáxias do VIPERS. Nas próximas seções descreveremos o pré-processamento dos dados do VIPERS e depois focaremos no tratamento das galáxias para a árvore de dados.

3.1. Tabela representativa das galáxias do VIPERS para o cálculo do redshift fotométrico

Para um melhor desempenho dos programas, é preciso usar uma amostra de galáxias do VIPERS que seja representativa das galáxias do DES. Isto significa que as galáxias de treinamento precisam estar na mesma área do céu das galáxias que serão utilizadas para o cálculo do z_{phot} (galáxias do DES). Observando as figuras 11 e 12 podemos ver que há uma sobreposição do espaço ocupado pelas galáxias em cada catálogo.

Com os dois catálogos baixados, foi feito um programa em python focado nas coordenadas RA e DEC das galáxias. Foi criado um código que permite fazer a correspondência entre os dois catálogos usando a distância entre as galáxias. Se a distância espacial entre duas galáxias (uma do VIPERS e outra do DES) for menor que a determinada pelo usuário, significa que se trata de uma galáxia do VIPERS representativa de uma galáxias do DES. No nosso caso, decidimos que aceitaríamos uma distância de 1 arcsec. As galáxias do VIPERS que estivessem a menos de 1 arcsec de qualquer galáxia do DES seriam selecionadas para uma nova tabela. Um resumo deste processo pode ser visto na figura 13. A tabela final ficou com 47 646 galáxias do VIPERS, quase a metade do total de objetos astronômicos do catálogo inicial do VIPERS.

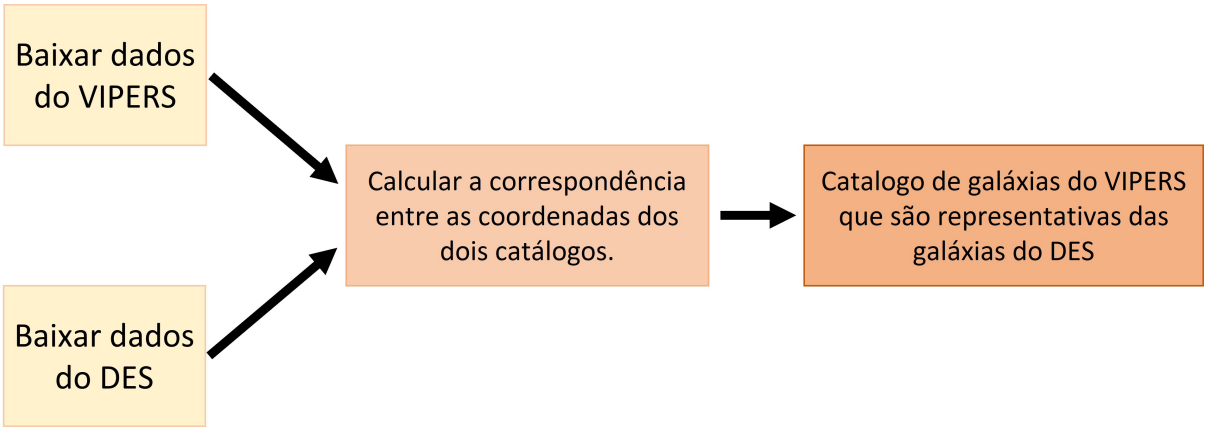


Figura 13: Fluxograma do processo de criar um catálogo de galáxias do VIPERS que estão a menos de 1 arcsec de ao menos uma galáxia do DES.

Após isso, foi feito um tratamento dessas galáxias para retirar os valores fora do padrão das magnitudes. O telescópio não consegue observar algumas galáxias em certos filtros e isto pode acontecer por diversos motivos, como poeira ou um objeto mais brilhante no caminho da observação. Para evitar que fosse colocada a terminologia `NaN` (***N**ot **a** **N**umber*), que faz com que o código trave e pare de rodar, foi atribuído um valor alto e fora do padrão para sinalizar que não foi possível observar a galáxia no filtro. No caso do catálogo do `VIPERS`, o valor fixado foi 99. Desta forma, podemos identificar os valores fora do padrão sem travar o programa. A nossa solução foi substituir esses valores atípicos pelo valor médio da tabela do `VIPERS` naquela magnitude. Por exemplo, para o filtro m_g das 47 646 galáxias da tabela, tivemos 265 galáxias com valores acima de 99, isto representa 0,55% do total. Calculamos a média dos valores de todas as galáxias dessa magnitude sem incluir os valores discrepantes e obtivemos a média de 23.6757. Então, todos os valores do filtro m_g que eram iguais a 99 foram substituídos pelo valor médio desse filtro, que é 23.6757.

Esta tabela, com as galáxias do `VIPERS` representativas das galáxias do `DES` e com os valores discrepantes retirados, foi utilizada no treinamento das três máquinas que calculam o redshift fotométrico.

3.2. Tabela de galáxias para a `K`-d Tree

Para a separação de galáxias que será feita pela estrutura de dados `K`-d Tree, é preciso construir uma tabela que contenha galáxias do `DES` que também estejam no `VIPERS` e que não estejam no catálogo do `VIPERS`. Esta tabela é necessária para fazermos a comparação das informações fotométricas das galáxias, que seria feita na forma de uma classificação binária. As galáxias que estão em ambos os catálogos serão classificadas como 1 e as que não estão serão classificadas como 0.

Esta divisão será feita a partir do número identificador de cada galáxia, que é único para cada galáxia em ambos os catálogos. Isto significa que, se houver uma galáxia em cada catálogo com o mesmo número de identificador, estamos lidando com a mesma galáxia e lhe será atribuído o valor 1 na coluna binária. Se a galáxia do `DES` não tiver nenhuma correspondente no catálogo do `VIPERS`, então será atribuído o valor 0 na coluna binária.

Para acelerar o processo, ao invés de pegarmos todos os objetos do `DES`, cerca de 700 milhões de linhas de uma tabela para serem analisadas, foi feito um refinamento das galáxias através das coordenadas `RA` e `DEC`.

O primeiro passo foi fazer a conversão dos valores dos pixels em cada catálogo. No `DES` os pixels foram divididos no formato `RING` e o `VIPERS` no formato `NESTED`. O formato `NESTED` é quando os pixels são ordenados de acordo com os doze pixels-base, que são os de menor resolução. No formato RING os pixels são enumerados contando do polo norte

até o polo sul ao longo de cada pixel na mesma latitude. Pode-se entender isto olhando a figura 14, retirada da documentação da biblioteca do `HEALPix` [72]. Esta biblioteca está na linguagem de programação python. Para cada método de ordenamento, há uma forma de enumerar os pixels.

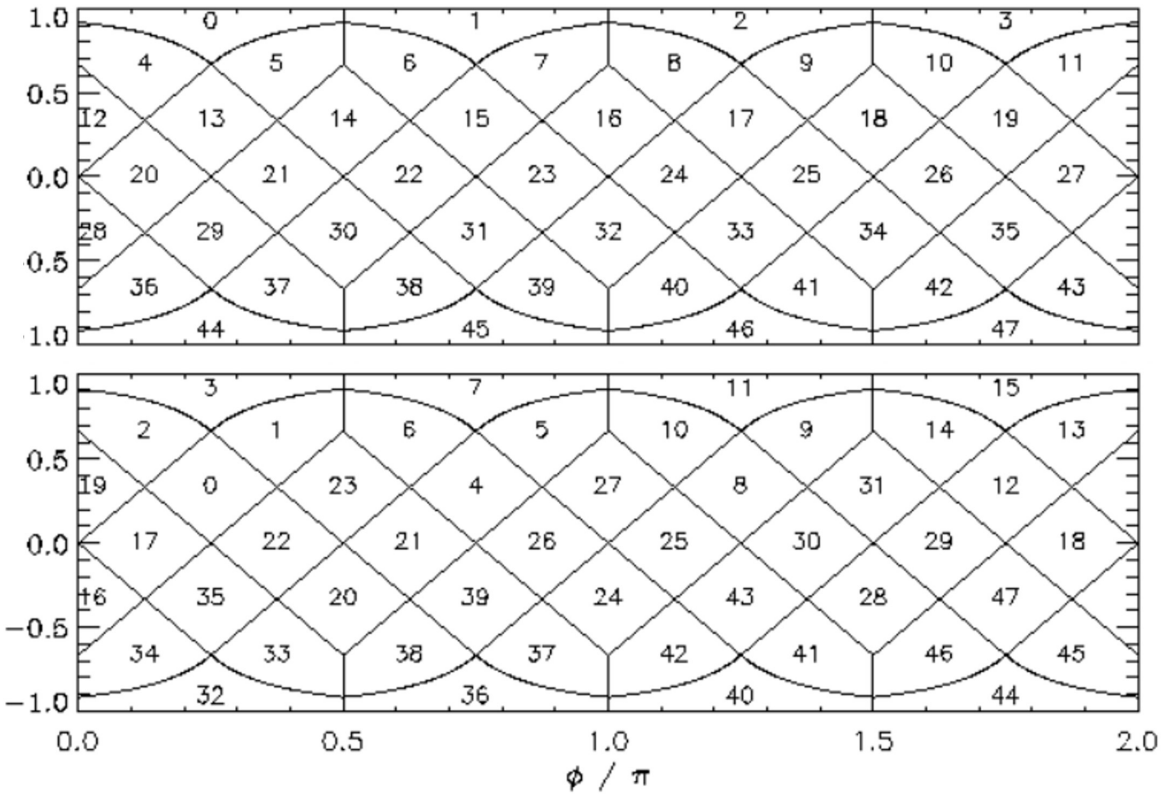


Figura 14: Figura adaptada da documentação do pacote de programação `HEALPix` produzida por Gólski et al. [72]. Em ambas as ilustrações, temos a área com a mesma resolução, ou seja, o mesmo espaço foi dividido na mesma quantidade de pixels. Na ilustração de cima mostramos o esquema de numeração dos espaços chamado `RING` e o `NEST` na segunda figura. É possível ver que, apesar da delimitação espacial das áreas ser a mesma, atribuímos números a cada área da região de forma diferente. O método `RING` enumera os pixels da esquerda para direita em cada linha, após todos os pixels da mesma linha terem sido enumerados, o método passa para a linha de baixo e enumera os pixels da esquerda para direita, e assim por diante. Enquanto isso, o método de enumeração do `NESTED` trabalha com conjuntos de pixels maiores, que, no caso da figura, agrupam quatro pixels, e para cada conjunto, o `NESTED` enumera o pixels dentro dele e então passa para outro conjunto de pixels e enumera os quatro pixels dentro, e assim por diante.

A partir da tabela de galáxias do `VIPERS`, pegamos a coluna que identificava o pixel da galáxia em um mapa com resolução 64, ou seja, uma parte do mapa astronômico foi dividido em 2^6 partes ($N_{side} = 6$). Com estes valores, foi feita a transformação dos pixels para descobrirmos quais pixels do `DES` estão na mesma área geográfica que os pixels do `VIPERS`. Então, esta parte do programa foca em converter os métodos de ordenamento de

cada catálogo de forma a equiparar os pixels para que seja possível fazer o refinamento das galáxias do DES.

Após essa conversão dos pixels do VIPERS de NESTED para RING, foram baixados apenas os pixels do DES que estivessem na mesma área que os pixels do VIPERS. Um mapa da área que foi utilizada no próximo passo pode ser vista na figura 15. Observando as figuras 12 e 15, podemos ver que a quantidade de galáxias do catálogo do DES usadas no treinamento é consideravelmente menor que a quantidade total de dados desse catálogo. Na figura 15, temos cerca de 1 milhão de galáxias, o que é menos de $0,19\%$ do catálogo do DES.

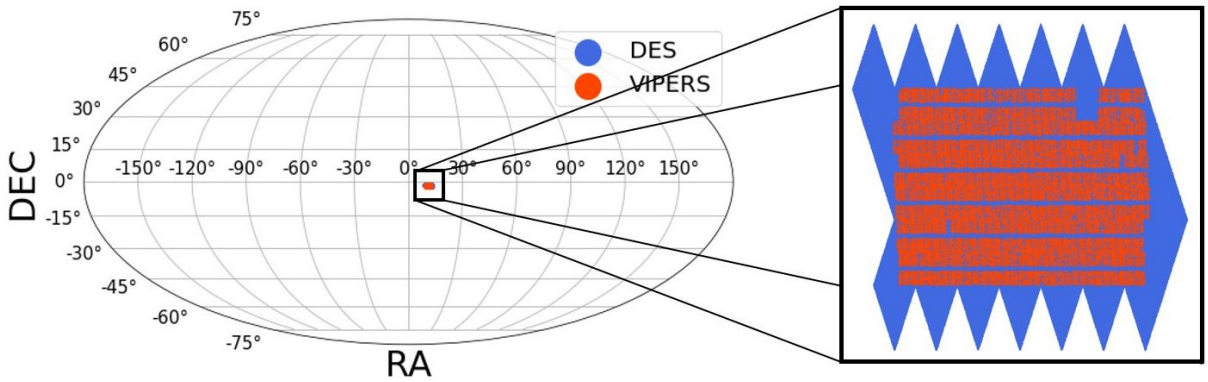


Figura 15: A figura é uma projeção do céu e mostra a área ocupada pelas tabelas de cada catálogo após o processo de refinamento. A área em laranja é a das galáxias do VIPERS e a região azul é a das galáxias do DES.

Com o refinamento da tabela, foi criada uma coluna binária que classificou as galáxias do DES de acordo com o fato de estar ou não nos dois catálogos. Para isso, foi analisado o identificador de cada galáxia. Se o identificador aparecer nos dois catálogos, a mesma galáxia está tanto na tabela do VIPERS quanto na do DES, e foi classificada com o número 1. Se a galáxia estiver apenas no DES, terá sido classificada com o número 0.

Na coluna binária de classificação, a porcentagem de galáxias com valor igual a 1 representou menos de 5% do total. Se treinássemos um aprendizado de máquina com esses dados desequilibrados, teríamos um modelo descalibrado. Uma imagem ilustrativa desse problema é a figura 16, que mostra a disparidade entre a classificação das galáxias antes desse corte. Se este conjunto de amostras fosse usado para treinamento, com esta desproporção, a máquina poderia ignorar a classe minoritária por falta de amostras. Isso implicaria na máquina prever que grande parte das galáxias teria classificação 0 [73]. Portanto, haveria um sobre-ajuste, a máquina iria memorizar o *output* como 0 e, quando fosse operar com as galáxias do DES, só teríamos o valor 0 como resultado.

Para evitar o problema de sobre-ajuste, foi feita uma diluição das galáxias de classificação iguais a 0 de modo que a tabela tivesse a mesma quantidade das galáxias de cada classe [74] [75] [76]. Esta diluição foi feita de forma aleatória, de maneira que a rede neural pôde ser treinada sem uma tendência de classe dominadora. Assim, ficamos com

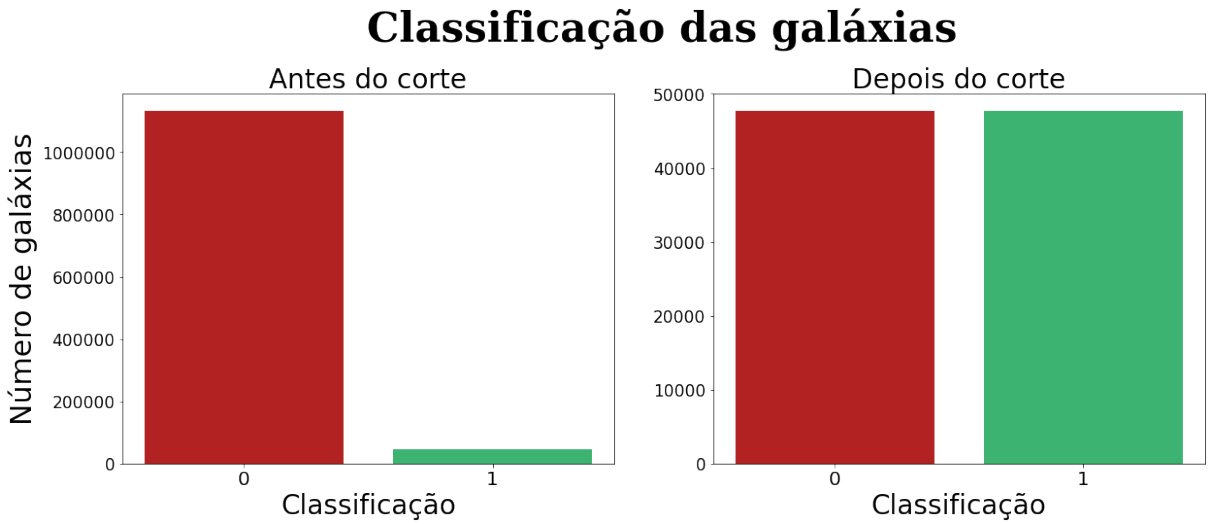


Figura 16: A figura mostra o número de galáxias classificadas antes e depois do corte. O corte selecionou de forma aleatória galáxias de classificação 0, de forma que o conjunto de amostras ficasse equilibrado.

uma tabela de quase 100 mil galáxias, com metade das galáxias classificadas como 1 e a outra metade classificada como 0.

Por fim, foi feito o mesmo tratamento detalhado na seção de substituir os valores atípicos pelo valor médio da tabela do VIPERS naquela magnitude. Isso significa que os mesmo valores médios utilizados anteriormente também foram usados nessa etapa.

A figura 17 é um fluxograma das etapas descritas acima e necessárias para montar o catálogo utilizado pela estrutura de dados K-d Tree.

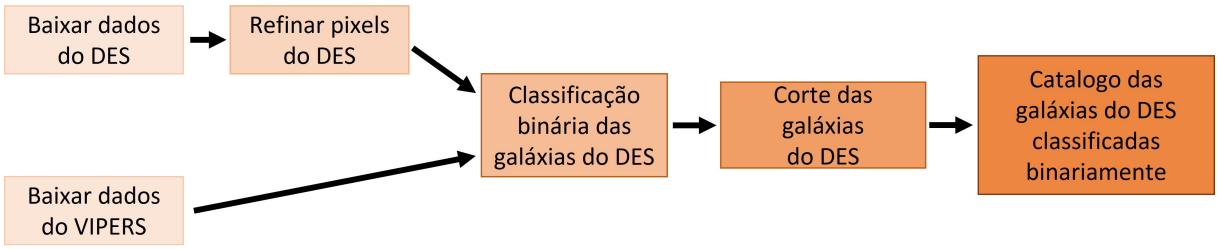


Figura 17: Fluxograma do processo de criar um catálogo de galáxias do DES que são classificadas binariamente em relação às galáxias do VIPERS.

4. Estrutura de dados

4.1. Árvore de dados K dimensional

O catálogo do DES contém milhões de galáxias que ocupam um espaço no céu bem maior do que as galáxias do VIPERS, como pode ser visto nas figuras 12 e 11. Isto implica em uma variedade maior de dados fotométricos destes objetos. Somos levados a uma desvantagem de agregar galáxias que não são parecidas, em relação a fotometria, com as galáxias usadas no treinamento. Se o conjunto de amostras tem galáxias do VIPERS que não são representativas do conjunto de treinamento, o redshift predito pela máquina terá um maior erro em relação ao redshift real. Isto quer dizer que não teremos um bom desempenho para essas galáxias.

Para evitar o uso de valores errados, foi utilizada uma estrutura de dados que divide as amostras de acordo com a sua similaridade fotométrica. Esta estrutura irá separar as galáxias em grupos de acordo com os valores nas propriedades fotométricas. A estrutura se chama árvore K dimensional (K-d Tree).

A estrutura de dados agrupa os pontos (no caso do nosso trabalho, são as galáxias) de acordo com a sua localização em um espaço de K dimensões [77]. Para o nosso trabalho, decidimos separar as galáxias em 4096 grupos, de acordo com as suas magnitudes. Em cada um destes grupos, foi construído um aprendizado de máquina que fornece a probabilidade delas serem um grupo representativo das amostras de treinamento.

A K-d Tree foi apresentada ao mundo pela primeira em vez em 1975, por Jon Louis Bentley, no paper *Multidimensional binary search trees used for associative searching*, que descreveu uma árvore de pesquisa binária multidimensional. O objetivo da árvore é armazenar dados de modo que fosse fácil e rápido fazer pesquisa de vizinho mais próximo (algoritmo K-ésimo vizinho mais próximo detalhado na seção) e consultas rápidas a grupos de interesse. Isso porque as árvores multidimensionais possibilitam o armazenamento de um conjunto de dados grande em subgrupos, o que otimiza a procura de dados específicos.

O mecanismo da árvore consiste em separar os pontos com base em uma dimensão a cada ciclo. Isto significa particionar os pontos em 2 subconjuntos considerando um eixo de cada vez. Na figura 18, temos um exemplo dessa estrutura para o número de 2 dimensões (k= 2). Em nosso trabalho, usamos 2 K-d Trees com 4 e 5 dimensões, mas, a fim de exemplificar como funciona, mostraremos a estrutura em um espaço com menos dimensões.

Na figura 18, temos 66 pontos localizados no diagrama xy. A estrutura começa o corte pelo eixo X, depois pelo Y, então volta para o X, e assim por diante de forma cíclica. O primeiro passo é escolher um ponto como o central - o nó - (os pontos azuis na figura 18). Na imagem 18a, o ponto central (1) é o primeiro ponto da metade da direita em relação ao eixo X. Com isso, separamos os dados em dois grupos, como se vê na figura 4.3. Quando há um número ímpar de pontos, o nó é o ponto que fica exatamente no meio

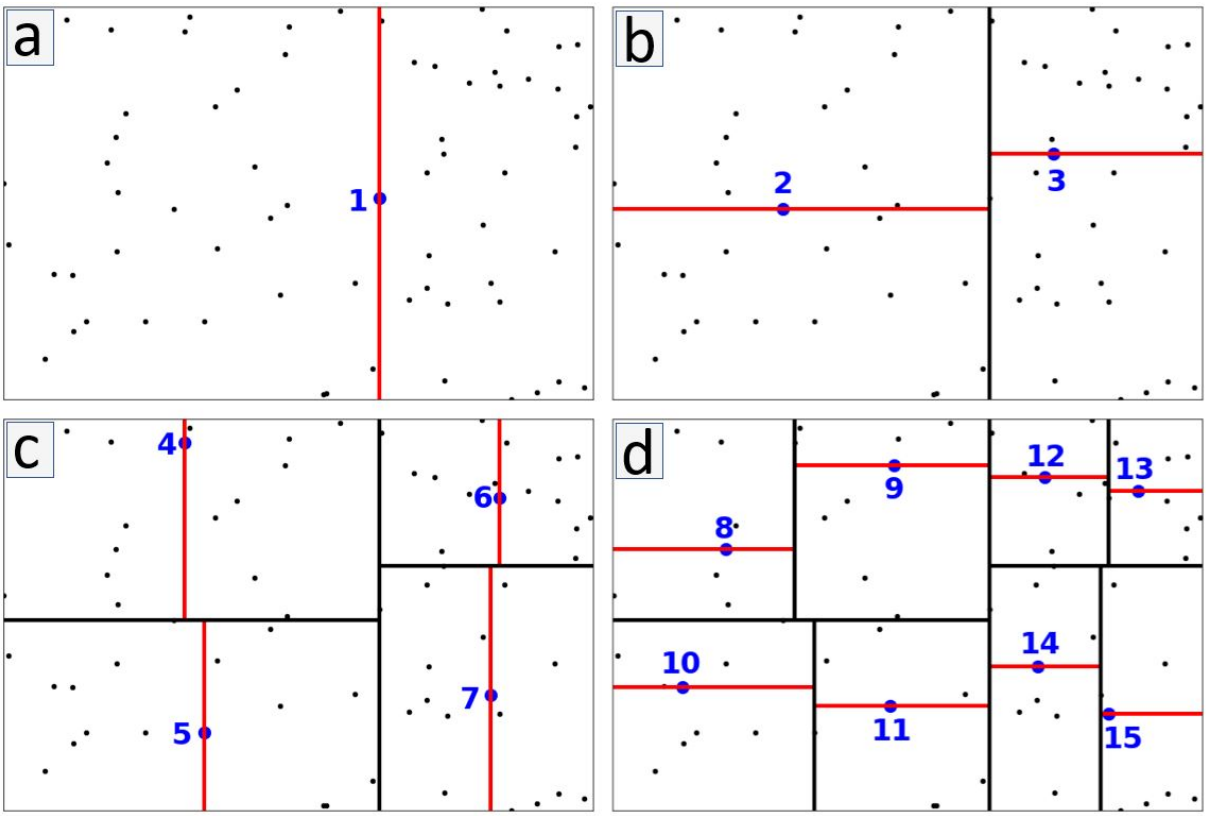


Figura 18: Representação de uma estrutura de dados de dimensão 2 (`k` = 2) com 4 camadas, resultando em 8 subgrupos

em relação ao eixo da camada. Quando há um número par de pontos, ainda haverá um ponto escolhido como nó, neste caso o ponto central será o primeiro ponto após a metade dos pontos. Por exemplo, se o conjunto de dados tem seis pontos, teremos três pontos de um lado da divisão, um ponto que será o nó, e os outros dois pontos do outro lado da divisão. A partir deste ponto, cada grupo (folha) é separado de forma que as folhas fiquem independentes entre si.

Na imagem 18b, cada grupo é dividido novamente, mas, agora, em relação ao eixo Y. É possível ver que em ambos os grupos o nó não têm o mesmo valor no eixo Y, ou seja, a divisão em cada grupo foi independente do outro grupo. Agora, as quatro folhas são independentes entre si, e, como a mudança das dimensões em cada camada é cíclica, e estamos usando apenas os eixos XY, temos que a próxima camada terá a sua divisão baseada nas coordenadas dos pontos no eixo X. Isto é, se estivéssemos em um espaço tridimensional com os eixos xyz, poderíamos ter a seguinte ordem para uma árvore de 4 dimensões: eixo x, eixo y, eixo z e eixo x de novo. Assim, na imagem 18c, temos as quatro folhas se dividindo novamente em subconjuntos, tendo o nó definido pelos valores dos pontos no eixo X em cada grupo. Por fim, na imagem 18d, temos uma divisão nas folhas através do ponto central em relação ao eixo Y. Observa-se que a cada camada o número de pontos em cada folha diminui, já que os pontos se restringem em grupos cada vez menores.

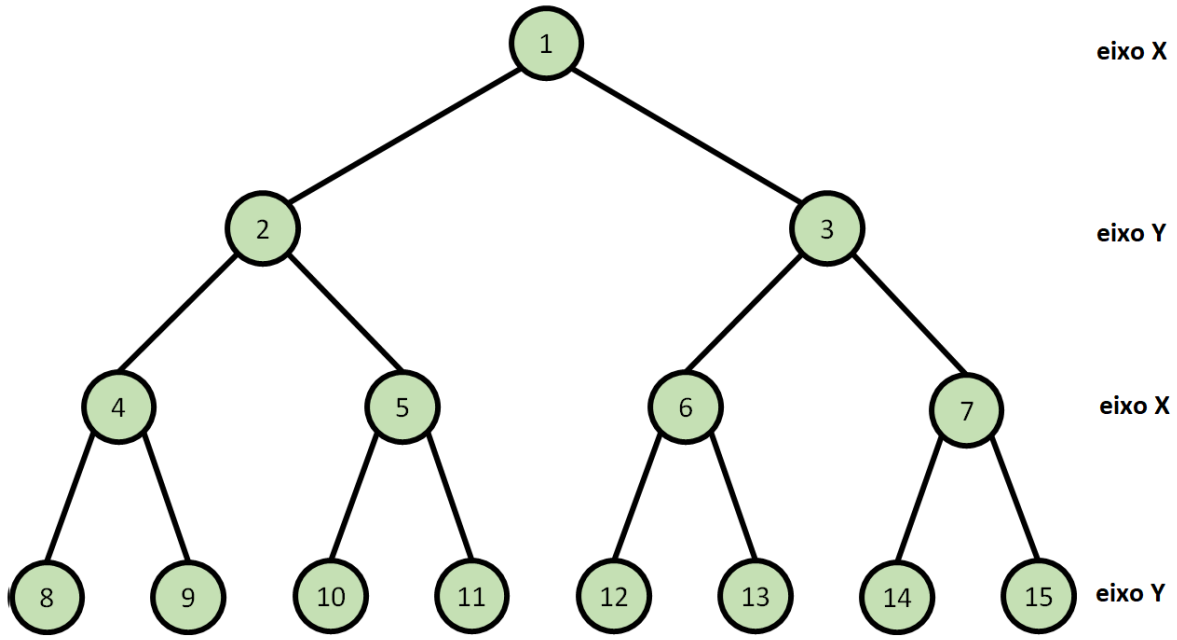


Figura 19: Diagrama da divisão da árvore de dados de 2 dimensões.

No caso deste trabalho, foram utilizados dois tipos de conjunto de dados: as cores das galáxias (conjunto `a`) e o das cores das galáxias junto com a magnitude m_i (conjunto `b`). Isso significa que foram construídas duas árvores: uma de 4 (`K` $= 4$) dimensões e outra de 5 (`K` $= 5$) dimensões. E, para uma maior precisão da rede neural em cada grupo, fizemos mais camadas além do exemplo. Inicialmente, foram feitas 9 camadas resultando em 512 grupos. No entanto, em grupos mais populosos de galáxias, mostrou-se necessário dividir ainda mais estes grupos de modo que fosse mais fácil, com a rede neural em cada folha, distinguir as galáxias do `DES` com fotometria parecida com a do `VIPERS`. Por isso, adicionamos mais 2 camadas, ou seja, a árvore ficou com 12 camadas e as galáxias foram divididas em 4096 grupos.

Para o caso da árvore `k` $= 4$ foram usadas as 4 cores $m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$ nessa ordem para formar a estrutura de dados. Desta forma, começamos comparando os pontos em relação à primeira dimensão: $m_g - m_r$, na próxima camada dividimos os dois grupos através dos valores de $m_r - m_i$, e assim por diante. Após as quatro camadas, voltamos a comparar as diversas folhas pelos valores $m_g - m_r$ das galáxias. Assim, foi feita esta mudança de camadas de forma cíclica até que se completassem 12 camadas, resultando em 4096 folhas.

De forma análoga à árvore `k` $= 4$, a árvore `k` $= 5$ teve os mesmos princípios, tendo as seguintes dimensões disponíveis para o corte: $m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$ e magnitude m_i. A ordem para a permutação das camadas foi exatamente a ordem escrita, ou seja, começamos com $m_g - m_r$, depois dividimos os dois grupos através dos valores de $m_r - m_i$, e assim por diante. Após as cinco camadas, voltamos a comparar as diversas folhas pelos valores $m_g - m_r$ das galáxias. Esta permutação das camadas foi feita de forma

cíclica, até a árvore chegar a 12 camadas, o que resultou em uma estrutura de dados com 4096 grupos, tal qual a árvore de 4 dimensões.

4.2. Aprendizado de máquina da K-d Tree

Com os grupos definidos, foi construída uma rede neural que, a partir dos dados do VIPERS (dado de treino), pudesse predizer quais galáxias do DES tinham valores fotométricos parecidos com os das galáxias de treino. As redes neurais usadas para calcular o redshift usaram os valores de cores e magnitudes das galáxias do VIPERS como base; contudo, é importante notar que nem todas as galáxias do DES são similares fotometricamente com as do VIPERS. Isso implica que tais galáxias, apesar da rede neural calcular o redshift fotométrico, podem ter um resultado com grande erro em relação ao valor real, já que fogem da área de treino da rede neural. Assim, a rede neural só consegue fazer predições com uma boa margem de certeza se as galáxias forem parecidas (quanto à cor). Por exemplo, podemos ver na figura 20 a posição fotométrica das galáxias dos dois surveys. Esta imagem é um gráfico representativo das 4 cores ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$) utilizadas no estudo. A figura 20 mostra como há uma interpolação entre as galáxias vistas pelo DES e pelo VIPERS. A figura também mostra uma grande área somente com pontos azuis (galáxias do DES). É nesta área que, apesar de os aprendizados de máquina calcularem o redshift fotométrico, há uma grande probabilidade deste z_{phot} ter um valor longe do real valor. O objetivo do aprendizado de máquina na K-d Tree é calcular esta probabilidade. Esta rede neural foi aplicada em cada folha da estrutura de dados.

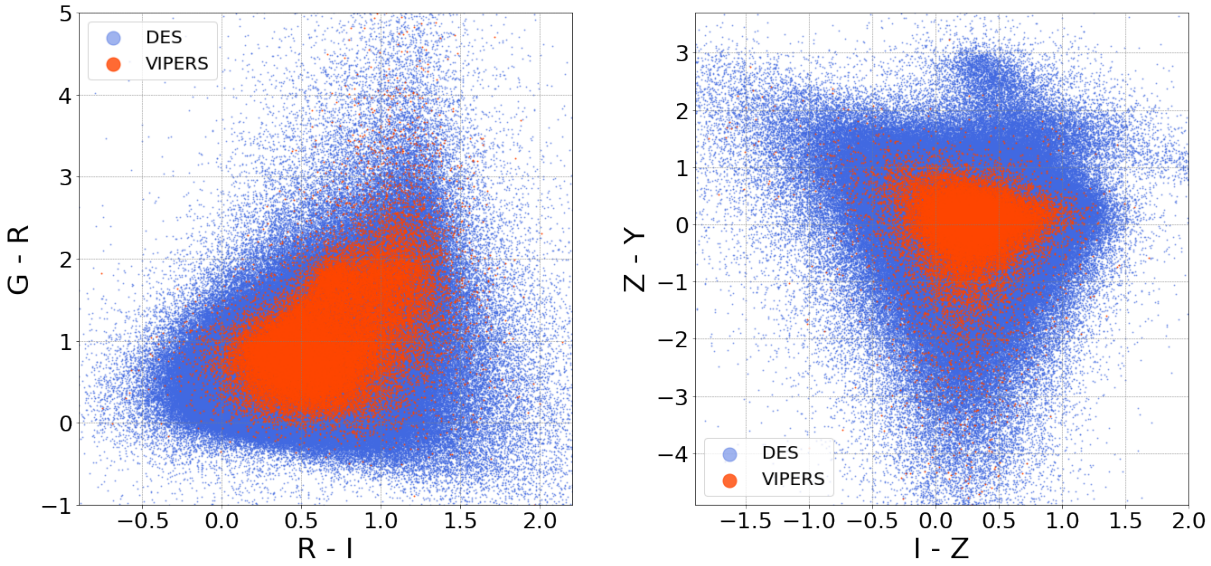


Figura 20: Gráfico das cores das galáxias do DES (azul) e das galáxias do VIPERS (laranja). A imagem contém cerca de 46 mil galáxias do VIPERS e 600 mil galáxias do DES.

No capítulo 3, foi detalhado o processo de criação da tabela de galáxias do DES para o treino das redes neurais. A partir dessa tabela, foram testados diferentes aprendizados

de máquina, de modo que se obtivesse o melhor método para este trabalho. O processo de treino foi igual para todos os mecanismos e a diferença entre as máquinas são os algoritmos utilizados. O catálogo das galáxias foi dividido em três partes: 65% das galáxias para treinamento, 10% das galáxias para validação e 25% das galáxias para o teste. O resultado do conjunto de testes será a probabilidade λ das galáxias serem parecidas fotometricamente. A partir deste resultado, podemos fazer uma avaliação dos métodos. Em cada conjunto, foi utilizada a tabela detalhada na seção 3.2 que contém os dados fotométricos das galáxias e a classificação binária. Os algoritmos testados estão descritos a seguir.

`Regressão linear` (*Linear Regression*): É um método que tenta ajustar os dados de entrada em uma equação linear, de modo que o resultado dessa equação seja o dado de saída. A equação

$$\lambda = \boldsymbol{a}\boldsymbol{X} + b, \tag{30}$$

onde $\boldsymbol{X}$ é o dado de entrada da máquina. Se o *input* tiver mais de uma dimensão, $\boldsymbol{a}$ também será um vetor de mais de uma dimensão de forma que o resultado da multiplicação $\boldsymbol{a}\boldsymbol{X}$ seja um valor unidimensional. Por exemplo, no caso de $\boldsymbol{X} = [x_1, x_2, ..., x_m] \in \mathbb{R}^m$ com m dimensões teremos $a = [a_1, a_2, .., a_m] \in \mathbb{R}^m$. A multiplicação do termo $\boldsymbol{a}\boldsymbol{X}$ será feita como,

$$a\boldsymbol{X} = \begin{bmatrix} a_1 & a_2 & ... & a_m \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_m \end{bmatrix} = \begin{bmatrix} a_1x_1 + a_2x_2 + ... + a_mx_m \end{bmatrix} \quad . \tag{31}$$

A equação (30) é a fórmula que o algoritmo da regressão linear utiliza para calcular a probabilidade. O vetor $\boldsymbol{a}$ é composto por constantes em cada dimensão (a_i é constante para qualquer i), assim como b é uma constante. Os valores destas constantes são definidas na fase de treinamento da máquina. Nesta fase o λ é a classificação binária das galáxias, isto implica no resultado na fase de treinamento entre 0 e 1. Para as fases subsequentes, a máquina calcula λ de todo o conjunto fornecido ao aprendizado de máquina e normaliza a partir do maior valor, desta forma todo o resultado está entre 0 e 1.

O algoritmo de regressão linear otimiza os valores dos coeficientes da equação (30) através do erro quadrático médio,

$$EQM(\hat{\lambda}) = \frac{1}{N}\sum_{i=1}^{N}(\lambda - c_i)^2 \quad . \tag{32}$$

O objetivo é obter o menor valor de $EQM(\hat{y})$ para o conjunto de teste [78] [79]. Na equação (30), λ é a probabilidade calculada pelo aprendizado de máquina e c_i é a classe da galáxia i.

`K-ésimo vizinho mais próximo` (*K-Nearest Neighbors*): É um método que implementa o algoritmo do K-ésimo vizinho mais próximo. Para não confundirmos com a estrutura de dados, dizemos que, para este método, a variável t representa o número de vizinhos mais próximos. Este algoritmo usa os valores das galáxias próximas ao conjunto de treino para predizer o dados de saída [80] [81] [82].

Por exemplo, suponhamos que definimos $t = 4$ como o número de vizinhos. No treinamento de uma galáxia, a máquina observará a classificação dos 4 vizinhos mais próximos fotometricamente dessa galáxia. Denominando uma dada galáxia como `a` e outras 4 galáxias mais próximas como `b, c, d` e `e`, temos que a distância é calculada através dos valores dos dados de entrada. Então, para a árvore treinada apenas com as cores, dizemos que a distância é calculada por

$$d = \sqrt{(a_{g-r} - b_{g-r})^2 + (a_{r-i} - b_{r-i})^2 + (a_{i-z} - b_{i-z})^2 + (a_{z-Y} - b_{z-Y})^2} \quad , \tag{33}$$

em que σ_{f-t} é a cor `f-t` da galáxia σ. Com a distância das 4 galáxias mais próximas calculadas, ou seja, as 4 galáxias com menor d para a galáxia `a`, a máquina observa a classificação de cada uma delas. A métrica utilizada para calcular a probabilidade será o inverso da distância entre duas galáxias. Para cada galáxia será atribuído um peso ω_i que é o inverso da distância entre duas galáxias, ou seja, para as galáxias `a` e `b` teríamos $\omega_{ab} = m/d_{ab}$, em que m é uma constante que normaliza os pesos de forma que a soma dos pesos seja 1 ($\omega_{ab} + \omega_{ac} + \omega_{ad} + \omega_{ae} = 1 \rightarrow m/d_{ab} + m/d_{ac} + m/d_{ad} + m/d_{ae} = 1$). A probabilidade λ calculada para este método será dada pela equação

$$\lambda = \frac{1}{t} \sum_{a,b,c,d} c_i \omega_{ai} = \frac{1}{t} \big(c_b \omega_{ab} + c_c \omega_{ac} + c_d \omega_{ad} + c_e \omega_{ae} \big) \quad . \tag{34}$$

Acima, c_i é a classificação da galáxia i, no nosso caso as duas classificações possíveis são 1 e 0. Se todas tivessem classificação 1, a galáxia `a` teria a sua probabilidade determinada como 1, ou seja, a probabilidade dessa galáxia ser parecida fotometricamente com o conjunto de treino é de 100%. O mesmo acontece na direção de oposta, ou seja, se todas as galáxias tivessem classificação 0, a `a` teria probabilidade 0%. No caso em que as galáxias tivessem classificações diferentes, iria ser considerada a distância de cada galáxia para a galáxia `a` para o cálculo da probabilidade. Se as duas galáxias mais próximas tivessem classificação 1 e as duas mais distantes classificação 0, a galáxia `a` teria a sua probabilidade mais perto de 1 do que de 0.5, dado que as galáxias mais próximas têm um peso maior no cálculo da sua probabilidade. Os pesos da distância são os parâmetros calculados na fase de treino e aperfeiçoados na fase de validação. Para normalizar a probabilidade λ, a

equação (34) é dividida pelo número t de vizinhos escolhidos pelo usuário. Isto faz com que a probabilidade esteja entre 0 e 1.

Para definir o número de galáxias vizinhas que seriam consideradas, foram calculadas as probabilidades em 100 máquinas diferentes. A cada aprendizado de máquina foi adicionada mais um vizinho, a primeira máquina foi treinada para 1 vizinho, a segunda, para 2 vizinhos, até a centésima camada, que foi treinada para 100 vizinhos. Para determinar o melhor valor de vizinhos, foi usada a métrica do erro quadrático médio - equação (32). Deste estudo, foi determinado que o número de galáxias vizinhas ideal para o cálculo foi $t = 20$. Um número menor de vizinhos calculava a probabilidade com uma alta taxa de erro do valor real, já que considerava poucas galáxias em seu cálculo. Um número de vizinhos maior não conseguiria diferenciar as configurações fotométricas das galáxias de cada classe, resultando em diversas galáxias com probabilidade perto de 0,5, não sendo possível definir em qual classe a galáxia deveria ser colocada.

`Classificador Bayesiano Gaussiano` (*Gaussian Naive Bayes*): No Classificador Bayesiano Gaussiano considera-se que os *inputs* são independentes entre si. O teorema de Bayes pode ser formulado como

$$P(A|B) = \frac{P(A) \times P(B|A)}{P(B)} \quad , \tag{35}$$

em que $P(A)$ é a probabilidade de um evento A acontecer, $P(B)$ é a probabilidade do evento B acontecer e $P(A|B)$ é a probabilidade de A acontecer, dado que B já aconteceu e vice versa para $P(B|A)$. Para o nosso caso, $P(A)$ é a probabilidade de a galáxia ser fotometricamente similar (ser classificada como 1 - $P(1)$) e $P(B)$ de ser classificada como 0 - $P(0)$. Estas probabilidades são calculadas a partir da fração de galáxias classificadas como 0 ou 1 no conjunto de treinamento.

O cálculo do termo $P(B|A)$ é feito pela equação

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(- \frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right) \quad , \tag{36}$$

em que são considerados os dados de entrada do aprendizado de máquina. Acima, σ é o desvio padrão e μ, a média de uma distribuição gaussiana de cada parâmetro dos dados de entrada. Para cada parâmetro do *input*, será feita uma distribuição gaussiana das probabilidades de cada galáxia com base no conjunto de treinamento. Isto significa que, para a árvore de 4 dimensões, teremos 4 valores de σ e 4 de μ.

Podemos então reescrever a equação (35) como

$$\lambda = \frac{P(1) \times P(x_i|y)}{P(0)} \quad . \tag{37}$$

`Classificador de árvore de decisão` (*Decision Tree Classifier*): É um método que se assemelha à árvore de dados, pois divide os dados de entrada em conjuntos. Teremos um nó que dividirá o conjunto em dois grupos, cada um se dividindo em dois conjuntos. A diferença é que a divisão não é feita pelos parâmetros do *input*, visando subconjuntos com o mesmo número de galáxias; a divisão é feita de acordo com parâmetros definidos pela máquina no conjunto de treinamento. Por exemplo, para dados de entrada com as 4 cores ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$) segmentamos as galáxias em grupos com $m_g - m_r < 0 \quad \& \quad m_i - m_z > 1$ e $m_g - m_r \geq 0 \quad \& \quad m_i - m_z \leq 1$. Cada divisão pode conter 1 ou mais critérios para dividir as galáxias. Em cada grupo, será calculada a probabilidade da galáxia de pertencer à classe i. Quanto mais camadas, mais complexa a árvore. A próxima etapa da construção deste método é calcular a probabilidade p_i de cada classe. Consideramos o número de galáxias da classe i no subgrupo definido pela árvore pelo número de galáxias no total do conjunto,

$$p_i = \frac{\begin{array}{c}número\ de\ galáxias \\ de\ classe\ i\end{array}}{\begin{array}{c}número\ total \\ de\ galáxias\end{array}}. \tag{38}$$

Com isso, é calculada a entropia do sistema,

$$\Delta S = -\frac{p_i}{p_i + p_j} log_2\left(\frac{p_i}{p_i + p_j}\right) - \frac{p_j}{p_i + p_j} log_2\left(\frac{p_j}{p_i + p_j}\right) \quad , \tag{39}$$

em que i e j são as probabilidades de cada classe. Em cada camada será calculada a entropia dada pela equação (39). Para uma árvore com n camadas, teremos que a galáxia de teste será colocada em um subgrupo definido pela última camada da árvore e a sua entropia final será dada por $\Delta S = \Delta S_n - \Delta S_{n-1}$. Sendo ΔS_{n-1} a entropia calculada no grupo da camada n da qual a galáxia pertence e ΔS_n é a entropia do grupo em que a galáxia está na última camada da árvore. Se olhássemos a figura 4.3 e as galáxias pertencentes ao grupo 8, $\Delta S_n = \Delta_4$ seria a entropia calculada pela equação (39) no grupo 8 e $\Delta S_{n-1} = \Delta S_3$ a entropia calculada no grupo 4 da forma descrita por (39). Com o treinamento, o mecanismo da árvore de decisão irá definir um valor de entropia ΔS_{fixo}. As galáxias com entropia acima deste valor serão classificadas como galáxias pertencentes à classe i e galáxias com entropia abaixo deste valor serão classificadas como galáxias pertencentes a classe j [83] [84].

`Classificador de floresta aleatório` (*Random Forest*): É um método que consiste em construir diversas árvores de decisão. Cada árvore foi construída da forma descrita no método anterior, mas em cada árvore os valores para as divisões dos grupos são diferentes [85] [84]. Por exemplo, se um grupo foi dividido em dois conjuntos pelo critério de

$m_g - m_r \leq 1$ & $m_i - m_z > 1$ e $m_g - m_r > 1$ & $m_i - m_z \leq 1$, uma outra árvore terá o grupo dividido pelo critério $m_r - m_i \leq 2$ & $m_i - m_z > 0 e 1 m_r - m_i > 2$ & $m_i - m_z \leq 0$. É possível definir o número de árvores criadas. O número padrão é 100, ou seja, são criadas 100 árvores de decisão com critérios, para a divisão de grupo, diferentes.

A galáxia de teste passará por todas estas árvores e terá a entropia calculada para cada árvore de decisão; no fim, este método terá 100 valores de entropias para uma galáxia. Como no método *Decision Tree Classifier*, para cada árvore de decisão, haverá um valor ΔS_{fixo} que irá nos dizer a qual classe a galáxia pertence. Teremos 100 resultados e cada um destes resultados diz a qual classe a galáxia pertence com base nos critérios de cada árvore. A probabilidade de a galáxia pertencer a classe 1 será dada por

$$\lambda = \frac{1}{100} \sum_{m=1}^{100} c_m, \tag{40}$$

em que c_m é a classe designada para a galáxia por cada uma das 100 árvores. Isto significa que, se a árvore m definiu a galáxia como pertencente a classe 1, teremos $c_m = 1$; se esta árvore tiver definido a galáxia para a classe 0, teremos $c_0 = 0$.

`Classificador AdaBoost` (*AdaBoost Classifier*): É um método que usa estímulo adaptativo (*Adaptative Boosting*). É um método que utiliza o mecanismo da rede neural para calcular os seus resultados. Neste método os dados de entrada são os parâmetros fotométricos das galáxias e a saída é a probabilidade de a galáxia pertencer à classe 1.

Este método combina diversas redes neurais em série, o que significa que a galáxia irá passar por diversas redes neurais até ter o seu resultado. Para este método é construída uma rede neural (primeira iteração) que utiliza os dados de entrada do conjunto de treinamento para calcular os pesos, camadas e funções de ativação e o seu resultado será um grupo de dados que pode ter n parâmetros (não necessariamente o mesmo número de parâmetros dos dados de entrada e nem dos dados de saída). Este resultado é multiplicado por um peso ω, cujo valor é definido no treinamento, e passa por outra rede neural (segunda iteração) com diferente configuração que calcula um novo resultado. Este resultado é então multiplicado por novos pesos e passa por outra rede neural (iteração 3).

Um exemplo de como funciona este algoritmo é a figura 21. Neste exemplo estamos lidando com um aprendizado de máquina com 3 redes neurais, ou seja, 3 iterações.

Considerando a equação (17) do capítulo 2 e que $\mathfrak{F}$ é o resultado de uma rede neural, podemos dizer que o resultado deste algoritmo será dado por

$$\lambda = \mathfrak{F}_n(\mathfrak{F}_{n_1}(...(\mathfrak{F}_2(\mathfrak{F}_1(\mathbf{x})))...)) \quad , \tag{41}$$

em que n é o número de redes neurais definido pelo usuário. Para o nosso trabalho, foi definido que o mecanismo AdaBoost teria 50 redes neurais ($n = 50$) para calcular o resultado, que é a probabilidade de a galáxia pertencer à classe 1.

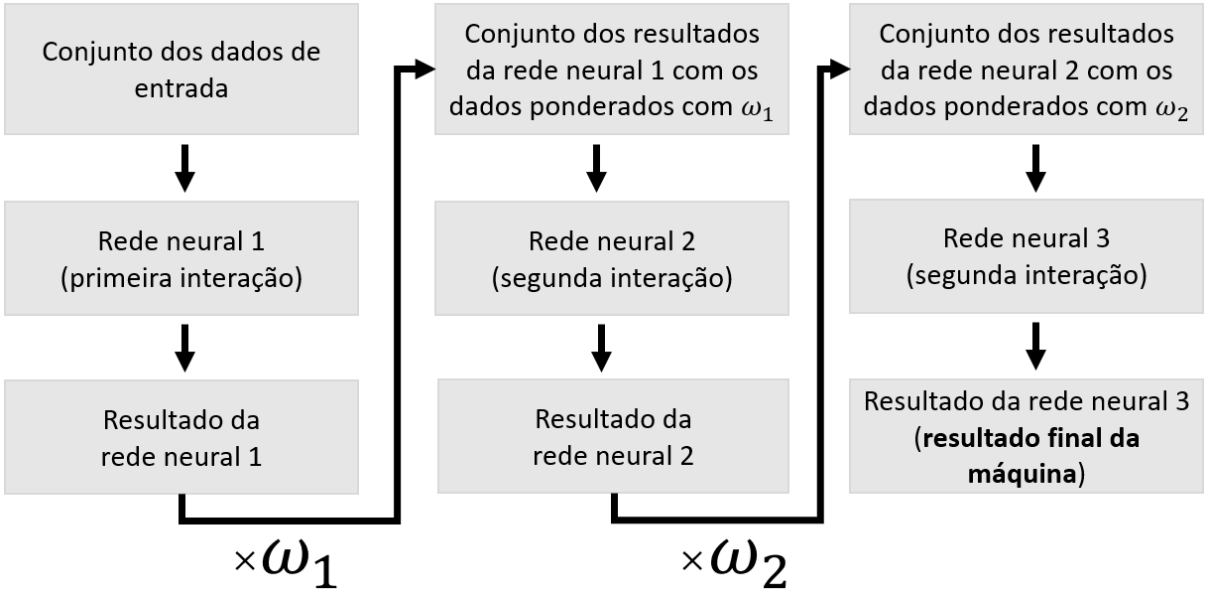


Figura 21: Diagrama do exemplo da estrutura do algoritmo Classificador AdaBoost. As setas representam a sequência de passos da máquina. O primeiro passo é fornecer o conjunto de dados para a primeira rede neural que então calcula o *output*, este dado de saída será um conjunto de valores diferentes do dado de saída dado pelo conjunto de treinamento, pois será processado pela próximas redes neurais [86] [87]. O resultado da rede neural 1 é multiplicado pelo peso ω_1 e fornecido à segunda rede neural. O processo é repetido até o resultado final dado pela terceira rede neural, este resultado tem que estar de acordo com o *output* do dado de saída do conjunto de treinamento. Os valores dos parâmetros das redes neurais e dos pesos serão definidos na fase de treinamento da máquina.

`Análise quadrática` (*Quadratic Discriminant Analysis*): aqui calcula-se a covariância Σ_i de cada classe i. A função da probabilidade de cada classe é dada por

$$\delta_i(\boldsymbol{x}) = \frac{1}{2} log|\Sigma_i| - \frac{1}{2}(\boldsymbol{x} - \mu_i)^T \Sigma_i^{-1}(\boldsymbol{x} - \mu_i) + logp_i \tag{42}$$

com $\boldsymbol{\mu_i}$ sendo o vetor das médias de cada parâmetro das informações fotométricas das galáxias para cada classe i. Para $k = 4$ dimensões, teremos $4\boldsymbol{\mu_i}$. O termo p_i é a probabilidade de cada galáxia pertencer a uma classe, sendo o número de galáxias na classe i do grupo de treinamento dividido pelo número total de galáxias do conjunto de treinamento, logo $\Sigma_{i=1}^{2} p_i = 1$, para duas classes de galáxias. O termo $\boldsymbol{x}$ é o dado de entrada do aprendizado de máquina na forma de vetor, em que cada termo do vetor é um parâmetro. Para um aprendizado de máquina com as 4 cores ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$), o vetor $\boldsymbol{x}$ tem 4 parâmetros. Os valores dos elementos das covariâncias Σ_i de cada classe serão determinados através do conjunto de treinamento das galáxias com um erro quadrático médio, equação (32), mínimo. A probabilidade de a galáxia ser classificada como da categoria da classe 1 é dada por $\lambda = \delta_1/(\delta_1 + \delta_2)$ [88] [89].

É importante ressaltar que, embora esses métodos sejam classificadores, escolhemos que a sua saída não fosse um número binário, e sim a probabilidade de a galáxia ser classificada como pertencente a classe 1 (galáxias do DES com dados fotométricos parecidos com os das galáxias do VIPERS). Na figura 22 podemos ver o resultado de cada método. A imagem traz seis histogramas da probabilidade de as galáxias do conjunto de teste serem classificadas como galáxias da classe 1 por cada método, além da classe original com que foram categorizadas. As galáxias em azul foram classificadas com 0, ou seja, não são similares, enquanto as em laranja foram classificadas com 1.

Na imagem 22 há histogramas das probabilidades das galáxias serem similares fotometricamente ao conjunto de treino VIPERS dos aprendizados de máquina para calcular o redshift fotométrico. Isso significa que, se esse mecanismo classificar uma galáxia com probabilidade baixa, significa que o valor predito do redshift provavelmente estará longe do real. Os resultados de cada algoritmo nos ajudaram a decidir qual procedimento iríamos usar. Idealmente um indicativo de um bom método seria ter um histograma com dois picos separados de galáxias e que não tivesse muita sobreposição das galáxias classificadas diferentemente. Assim, o algoritmo prediz a probabilidade perto do valor real.

No histograma da regressão linear é possível ver que os valores das probabilidades estão muito próximos. Isso significa que, apesar de haver dois picos, há uma gama de galáxias que serão classificadas de forma errada. Portanto, não saberíamos dizer se o redshift foi predito de maneira correta para grande parte das galáxias.

O resultado do K-ésimo vizinho mais próximo tem os dois picos bem definidos que indicam uma maior concentração de galáxias preditas corretamente. Além disto, há pouca área de sobreposição, o que sugere poucas galáxias classificadas de maneira errônea.

Na imagem do classificador Bayesiano Gaussiano, temos curvas parecidas com a do K-ésimo vizinho mais próximo, porém há uma maior concentração de galáxias em áreas sobrepostas. Outra coisa é o corte de galáxias perto da probabilidade 1 e o mecanismo não consegue desenvolver o padrão das galáxias do DES que são iguais às galáxias do VIPERS de treinamento.

O classificador de árvore de decisão é o histograma ideal em relação às suas curvas. Todavia, a região das galáxias que teve a sua classificação errada é bem grande, ou seja, a região em que as curvas azuis e laranjas se sobrepõem, o que faz com que esse método não leve a um bom resultado no cálculo da probabilidade.

O histograma do classificador de floresta aleatório não apresenta um pico de valores de probabilidade com galáxias classificadas como 0. Portanto, o código não consegue predizer com acurácia as galáxias que não são parecidas fotometricamente. Além disso, há poucas galáxias que tiveram a sua probabilidade determinada como maior que 0.7. Como sabemos que o conjunto de dados tinha 50% das galáxias classificadas com 1, o algoritmo não consegue identificar as galáxias desta classe.

O classificador AdaBoost não consegue analisar a diferença entre as galáxias das diferentes classes. O indicativo disto é que a probabilidade da maioria das galáxias é 50% de serem similares fotometricamente.

O método de análise quadrática teve um bom resultado com dois picos. O seu histograma se parece com o do classificador Bayesiano Gaussiano e K-ésimo vizinho mais próximo. Porém, este método tem uma região considerável de galáxias que são classificadas como 0, com uma probabilidade grande. Isto pode ser visto na curva da região sobreposta perto da probabilidade 1. O método tem muitos falsos positivos.

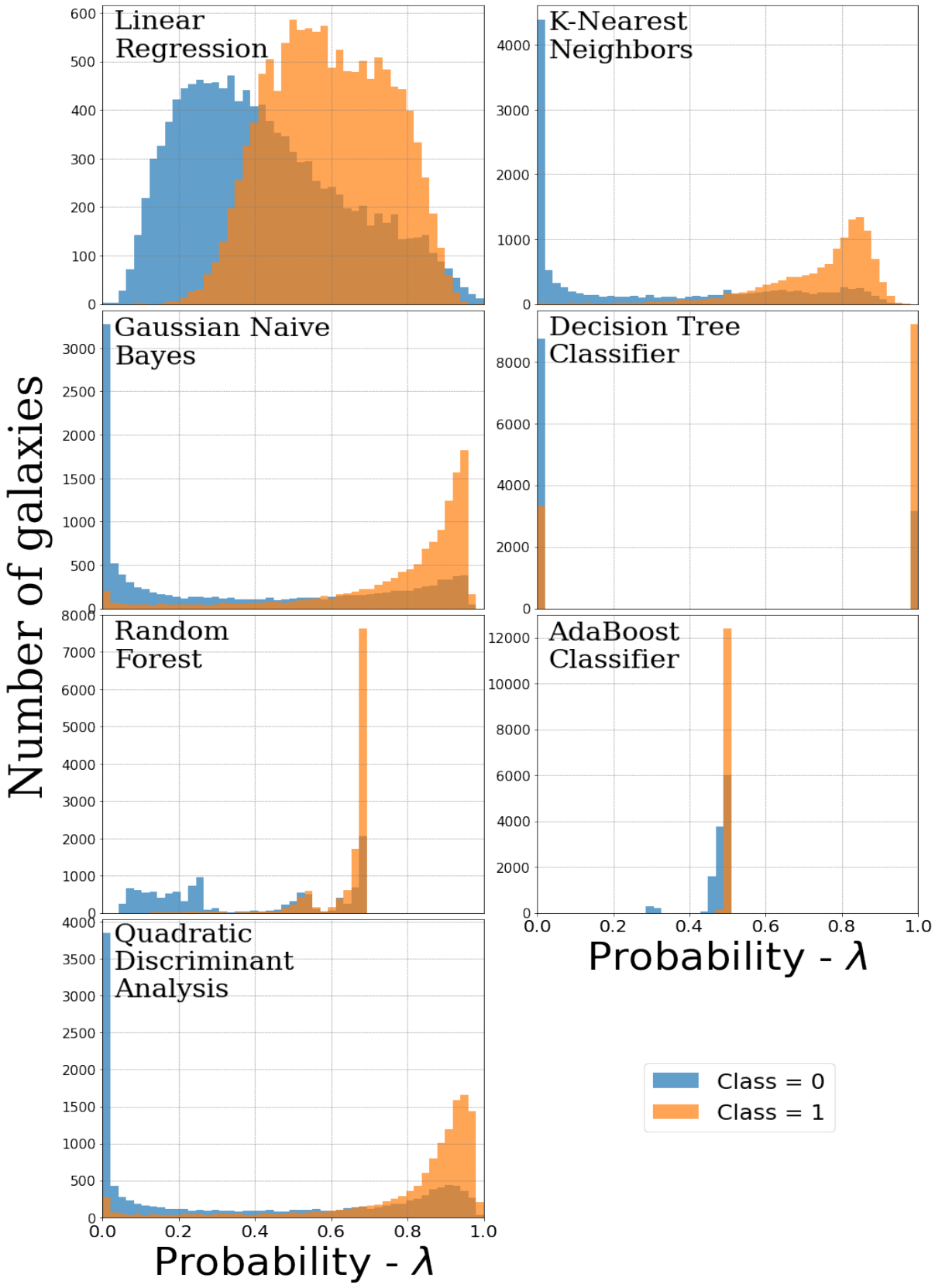


Figura 22: Conjunto de histogramas referente aos métodos testados para o cálculo da probabilidade λ. No eixo x representa as probabilidades λ das galáxias calculadas por cada método e no eixo y é o número de galáxias para cada valor da probabilidade. Essa probabilidade é o quão parecida fotometricamente uma galáxia do `DES` é do conjunto de galáxias do `VIPERS` que será usado para predizer o redshift fotométrico. As galáxias em azul são as que foram classificadas com 0 e em laranja as que foram classificadas com 1.

Uma forma de melhor perceber as diferenças dos resultados dos métodos é através da curva característica de operação do receptor, a curva ROC (*receiver operating characteristic*). É uma curva que caracteriza o comportamento dos algoritmos citados acima e que é obtida através da razão entre a taxa de verdadeiros positivos (sensibilidade) pela taxa de falsos positivos. O gráfico é um quadrado de área 1 em que a acurácia de cada algoritmo é dada pela área abaixo da sua curva. Verdadeiros positivos são galáxias cujo λ está perto de 1 e o seu valor real é 1. Falsos positivos são galáxias em que $\lambda \sim 1$, porém o seu valor real é 0. As galáxias com $\lambda \sim 0$, mas cujo valor real é 1, fazem parte do grupo dos falsos negativos. E galáxias com $\lambda \sim 0$ e com valores reais da classificação como 0, são chamadas de verdadeiros negativos. Então, teremos as seguintes equações para as taxas da curva ROC,

$$TVP = \frac{Verdadeiro\ positivo}{Verdadeiro\ positivo + Falso\ positivo}, \quad (43) \qquad TFP = \frac{Falso\ positivo}{Verdadeiro\ positivo + Falso\ positivo}. \quad (44)$$

As equações mostram como as taxas da curva ROC são calculadas. A taxa de verdadeiros positivos (TVP) considera o número de verdadeiros positivos sobre a soma de todas as galáxias classificadas como positivas ($\lambda \sim 1$). Enquanto a taxa de falsos positivos (TFP) considera o número de falsos positivos sobre a soma de todas as galáxias classificadas como positivas ($\lambda \sim 1$). A partir dessas definições, foi feita a figura 23. Para cada método foram fixados 100 valores de λ_{corte} entre 0 e 1 com passos de 0,01. Para cada λ_{corte}, os resultados das probabilidades foram divididos em dois grupos de galáxias. As galáxias abaixo de λ_{corte} foram classificadas como 0 e as galáxias acima deste valor foram classificadas como 1. Isto aconteceu para os 100 valores de λ_{corte}. A partir disso, foram contabilizadas as galáxias em cada categoria e comparadas com a classificação original delas. Deste modo, foram calculadas as taxas de verdadeiros positivos e falsos positivos, como mostrado nas equações (44) e (43) para cada λ_{corte}. Deste cálculo, foram obtidos diversos pontos para a curva ROC, cujo eixo x representa a taxa de falsos positivos e o eixo y representa a taxa de verdadeiros positivos como mostrado na figura 23.

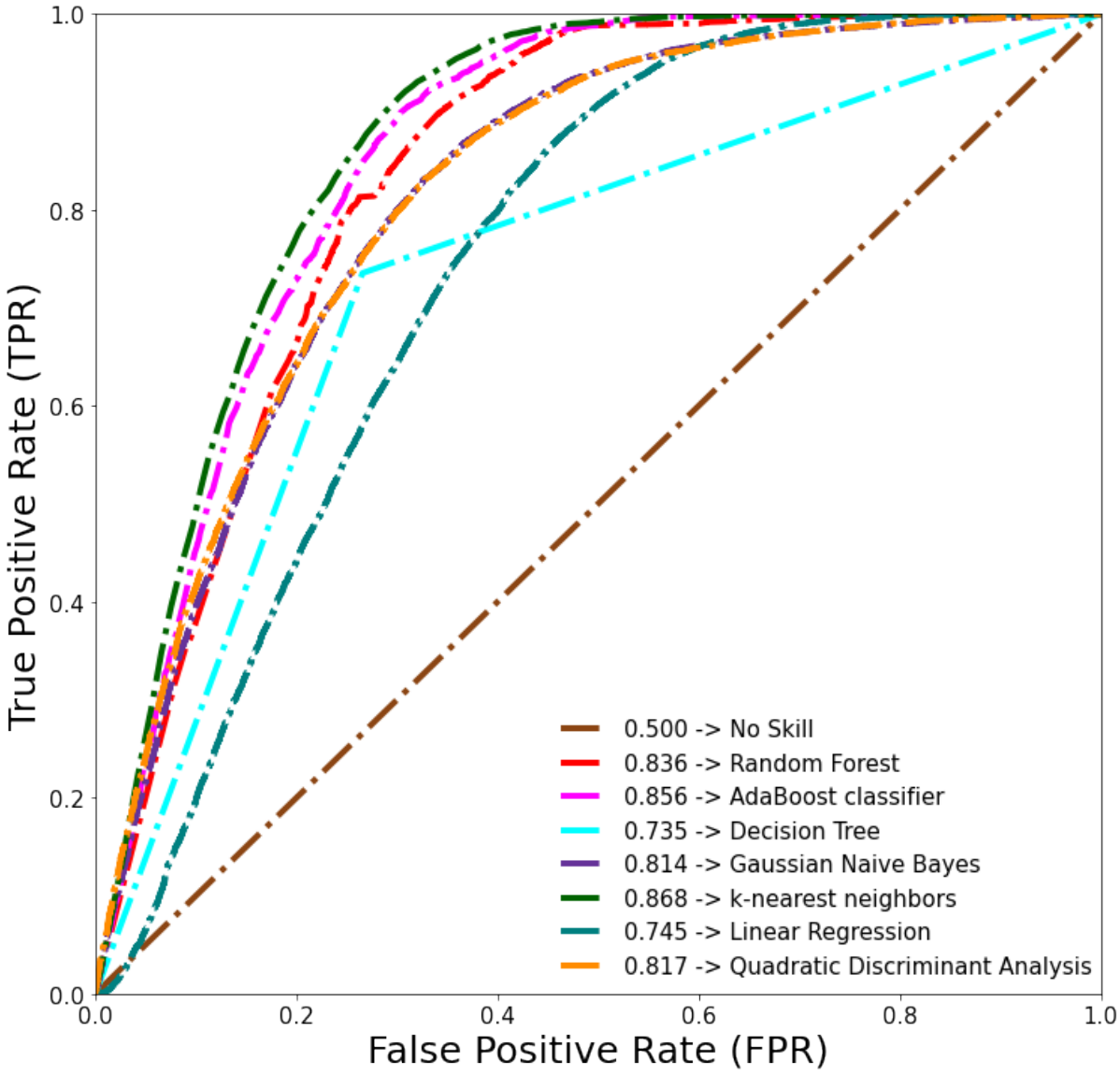


Figura 23: Gráfico das curvas ROC dos métodos testados para a K-d Tree. A área abaixo das curvas representa a acurácia desses métodos. A reta azul é a diretriz do gráfico e representa um classificador aleatório. O valor da acurácia de cada método, ou seja, a área embaixo da curva de cada método, é descrito na legenda do gráfico. O melhor método é definido pelo que tem a maior acurácia, neste caso, seria o algoritmo do K-ésimo vizinho mais próximo, enquanto o método com menor acurácia é o algoritmo da árvore de decisão.

A figura 23 mostra que o melhor método é o que tem a maior área sobre a sua curva. A linha azul é a diretriz do gráfico chamada de classificador aleatório (*no skill*). Qualquer curva que estiver abaixo dela seria descartada porque indica um classificador pior do que um classificador aleatório. Um classificador aleatório é uma máquina que designa apenas uma classe para todo o *input*. No nosso caso, seria uma máquina que indicaria que todas as galáxias pertencem à classe 1 ou 0. E como trabalhamos com um conjunto de dados equilibrado, significa que a acurácia deste classificador aleatório seria de 50%, como representada na figura.

Os valores da acurácia de cada método estão no próprio gráfico. É possível ver que o melhor método foi o que usa o algoritmo do K-ésimo vizinho mais próximo. Então, este foi o algoritmo escolhido para calcular as probabilidades λ de as galáxias serem fotometricamente similares.

4.3. O código

O código da K-d Tree foi escrito na linguagem de programação python e criado para este trabalho. Há pacotes de K-d Tree abertos ao público, como o do scipy[7] e o do sklearn[8], porém ambos não têm liberdade quanto à configuração da estrutura de dados. Por exemplo, não é possível escolher o número de camadas, o que é essencial para o nosso trabalho, pois em nossos testes vimos que K=9, ou seja, 512 grupos era pouco para as galáxias à nossa disposição. Além disso, estes pacotes foram construídos para conjuntos de dados pequenos; o padrão de configuração destes pacotes é ter menos de 50 folhas, enquanto o nosso código suporta 4096 folhas sem perder a precisão e a rapidez. Por fim, como é um código fechado, não conseguiríamos fazer a seleção das galáxias para as duas estruturas de dados (k=4 e k=5) mais o cálculo da probabilidade λ ao mesmo tempo. Isto significaria fazer cada etapa separadamente o que acarretaria em um custo de tempo mais alto do que o atual, cuja estrutura foi feita totalmente por nós.

A figura 24 é um fluxograma das etapas necessárias para a construção da estrutura de dados, o código detalhado está no apêndice D. O primeiro passo é baixar a tabela de galáxias do DES com a classificação binária das galáxias descrita na seção 3.2. Depois disto, há dois caminhos a serem seguidos: a configuração da estrutura da árvore e o treinamento do aprendizado de máquina para calcular a probabilidade λ. Neste código, criamos funções que são utilizadas pela estrutura de dados.

Na primeira função (`distancia`) é feito o cálculo da distância fotométrica entre as galáxias, o cálculo da distância é mostrado na equação (33). Serão fornecidos a função os dados fotométricos de duas galáxias, para $K = 4$, os dados serão as cores, e, para $k = 5$, os dados serão as cores e a magnitude no filtro i.

A segunda função (`menor_distancia`) determina a qual subconjunto de cada camada a galáxia pertence. São fornecidos a função, os valores fotométricos do ponto central (galáxias a) da camada e os valores fotométricos da galáxia a ser avaliada (galáxia b). Esta divisão é feita pelo processo descrito na seção 4.1. Cada camada fará o corte em uma determinada dimensão. Olhando o parâmetro do corte (dimensão) nas galáxias a e b, podemos decidir em qual grupo a galáxia b será colocada. Se a galáxia b tiver um parâmetro menor que a galáxia do nó, a galáxia avaliada será colocada no grupo da esquerda (em relação à figura 4.3); se for maior, a b será colocada no grupo da direita. Por exemplo, olhando a figura temos na segunda camada o nó de número 2, este ponto central

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html
[8] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html

divide as galáxias em dois subgrupos, o subgrupo 4 e o subgrupo 5. Esta função determina, através da dimensão de corte da camada, para qual subgrupo a galáxia deve ir. A função é recorrente, então, para a próxima camada, a galáxia `b` também será designada para um subgrupo, de acordo com os seus valores fotométricos e as informações fotométricas do ponto central. Este processo é recorrente para a galáxia `b` até alcançar as 12 camadas configuradas na K-d Tree.

A terceira função (`kdtree_ponto_mais_perto`) define os grupos das galáxias. Isto significa que esta função é a que faz o que é mostrado na figura 18. Dada uma dimensão, esta função definirá o ponto central que irá fazer a divisão do grupo em um subgrupo. Este nó (ponto azul na figura 18) será fixado na árvore e as galáxias subsequentes passarão pela segunda função para determinar em qual subgrupo serão colocadas. Esta terceira função é recursiva, então, depois disso, a função muda de camada e faz o mesmo processo até chegar na décima segunda camada da árvore.

A quarta função (`construir_kdtree`) é a que constrói a estrutura da K-d Tree. Esta função define as galáxias que serão fixadas como nós do sistema. Este programa analisa os grupos e determina onde será feita a divisão das galáxias em cada dimensão.

Criadas essas funções, há um código que é rodado de forma a combinar estas funções da árvore e determinar os 4096 nós da décima segunda camada. Na fase de treinamento, o código retorna os 4096 subgrupos da K-d Tree. O código salva as galáxias fixadas como pontos centrais e a estrutura da árvore. Assim, se quisermos dividir um outro conjunto de galáxias (tanto do `DES` quanto de outro catálogo) pela fotometria, a configuração da estrutura de dados está salva e pode ser utilizada nesta divisão com base nos valores fotométricos das galáxias do `VIPERS`. Ao salvarmos o grupo de nós, garantimos que todas as galáxias passarão pela mesma estrutura e terão o mesmo conjunto de subgrupos para serem empregadas.

O outro caminho do código tem como objetivo configurar a máquina para predizer a probabilidade λ das galáxias serem parecidas fotometricamente. O primeiro passo é estruturar a máquina; como foi decidido usar o algoritmo do k-ésimos vizinhos mais próximos na seção 4.2, o aprendizado de máquina usará este algoritmo para o cálculo da probabilidade.

Depois disso, foi feita a divisão das galáxias em três grupos: de treinamento, de validação e de teste. Como dito anteriormente, foi decidido por 65% das galáxias irem para o grupo de treinamento, 10% para validação e o restante dos 25%, para o conjunto de teste.

Por fim, a máquina foi treinada com a tabela de galáxias do `DES` que contém as informações fotométricas das galáxias e a classificação binária detalhada na seção 3.2. Após o treinamento, o mecanismo da máquina foi salvo para ser usado nas galáxias do `DES`.

Todo esse aparato permitiu que tivéssemos a estrutura de dados K-d Tree junto da configuração do aprendizado de máquina para todas as galáxias do DES. O código em python desse processo está no apêndice D.

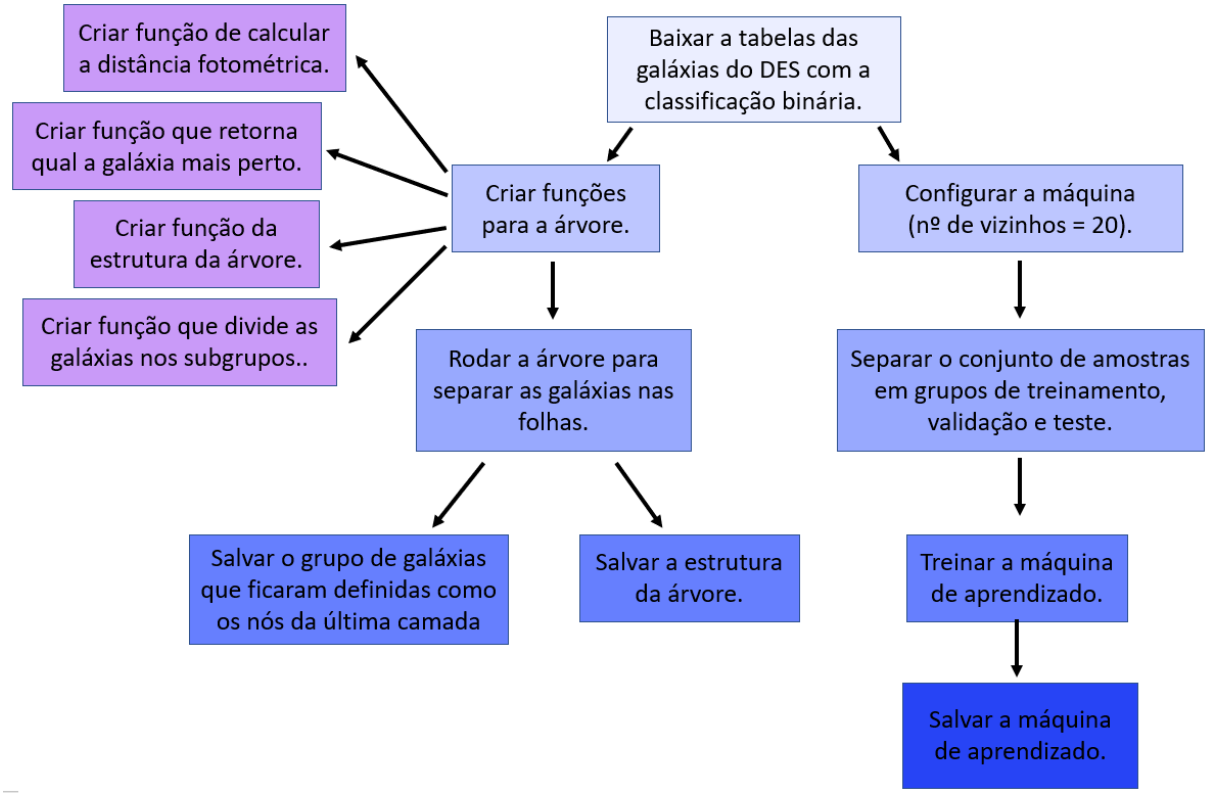


Figura 24: A figura acima é um fluxograma das etapas necessárias para construir a estrutura de dados detalhada neste capítulo. Após a criação da tabela com a classificação binária das galáxias, há dois caminhos para serem percorridos: a construção da estrutura de dados e o treinamento do aprendizado de máquina. Após estes caminhos serem trilhados, toda esta configuração é salva para ser usada em todas as galáxias do catálogo do DES.

4.4. Resultados

Com o algoritmo do aprendizado de máquina para o cálculo de λ definido, foi possível criar a árvore de K dimensões e rodar o aprendizado de máquina em cada uma das 4096 folhas. Para ver a diferença entre as duas árvores de dimensão $K = 4$ e $K = 5$, foi calculada a média dos λ em cada folha e a sua variância. Esse resultado pode ser visto na figura 25.

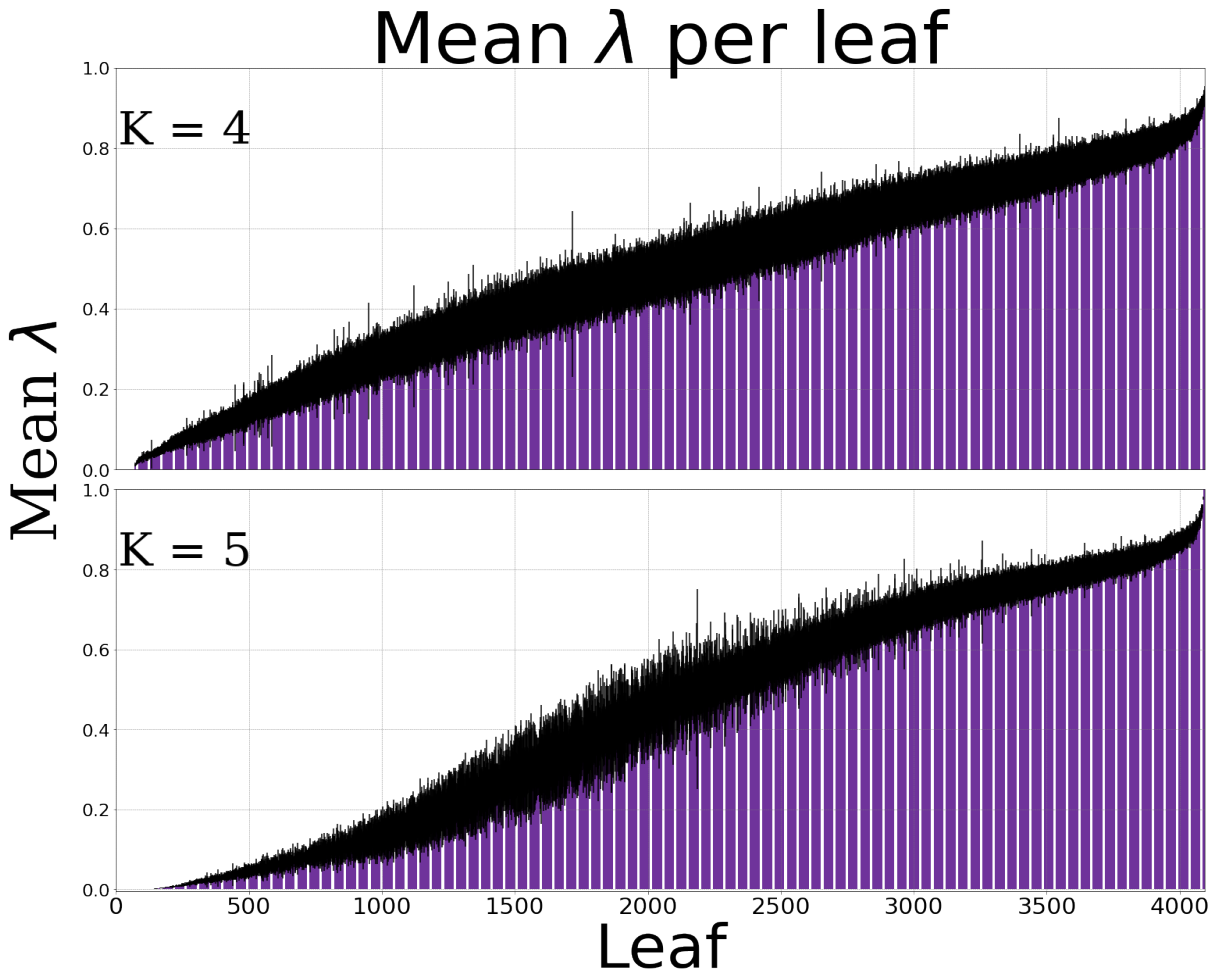


Figura 25: Gráfico da média dos λ calculados em cada folha junto com a sua variância. As folhas foram ordenadas com base na média do λ de cada grupo. As barras roxas representam as médias dos λ e as linhas pretas são a variância do λ em cada folha. A primeira imagem traz este estudo para a árvore de 4 dimensões, e a segunda imagem para a árvore de 5 dimensões.

A imagem 25 tem os valores médios de λ dos 4096 grupos que estão ordenados de acordo com estes valores. As retas pretas são a variância de λ em cada um desses grupos. Comparando ambas as árvores, podemos ver que há uma diferença em relação ao comportamento dos λ médios. A árvore de 5 dimensões tem uma mudança de valores mais abrupta, ou seja, os valores vão de λ médio baixo ($\lambda \sim 0,1$) para um valor alto ($\lambda \sim 0,8$) mais rapidamente do que na árvore de 4 dimensões. Na primeira imagem, a árvore de $\texttt{k} = 4$ tem a mudança dos valores de λ feita de forma mais gradual. Esta diferença no comportamento das curvas indica que, ao utilizarmos a magnitude m_i, o aprendizado de máquina consegue distinguir melhor a classificação das galáxias, fornecendo resultados com picos de histograma mais espaçados. Galáxias com $\lambda \sim 0,5$ significa que a máquina não conseguiu determinar a qual classe a galáxia pertence, por isso a probabilidade desta galáxia pertencer a qualquer classe é praticamente igual. Enquanto galáxias com λ cujo valor está em um dos extremos (perto de 1 ou perto de 0), significa que a máquina conse-

guiu predizer com mais confiança a qual classe a galáxia pertence. Uma forma visual de perceber esta distinção é olhar o histograma da probabilidade das galáxias; se houver dois picos nos extremos, significa que a máquina teve mais confiança para predizer a classe das galáxias. Esse cenário é o que queríamos ao olharmos a figura 22.

Para o conjunto de teste, pegamos uma folha na região mais populosa de galáxias. Podemos ver que, no histograma da figura 26, há características que queríamos para a nossa máquina, como os dois picos nos extremos da probabilidade. Em ambas as árvores as galáxias apresentam dois picos bem definidos e espaçados em cada canto do gráfico, o que significa que o aprendizado de máquina conseguiu identificar e classificar as galáxias com bastante confiabilidade. Assim, a máquina conseguiu determinar se a galáxia tinha ou não similaridade fotométrica com o conjunto de galáxias do `VIPERS`. Entre os picos maiores há picos menores de galáxias. Para a árvore $\texttt{K} = 4$, há 4 desses montantes nas regiões de probabilidade perto de 0,2, 0,4, 0,6, e 0,8. Para a árvore $\texttt{K} = 5$, há 3 desses montantes nas regiões de probabilidade perto de 0,3, 0,5 e 0,75. Isso implica que há regiões de galáxias cujos dados fotométricos são bem parecidos entre si.

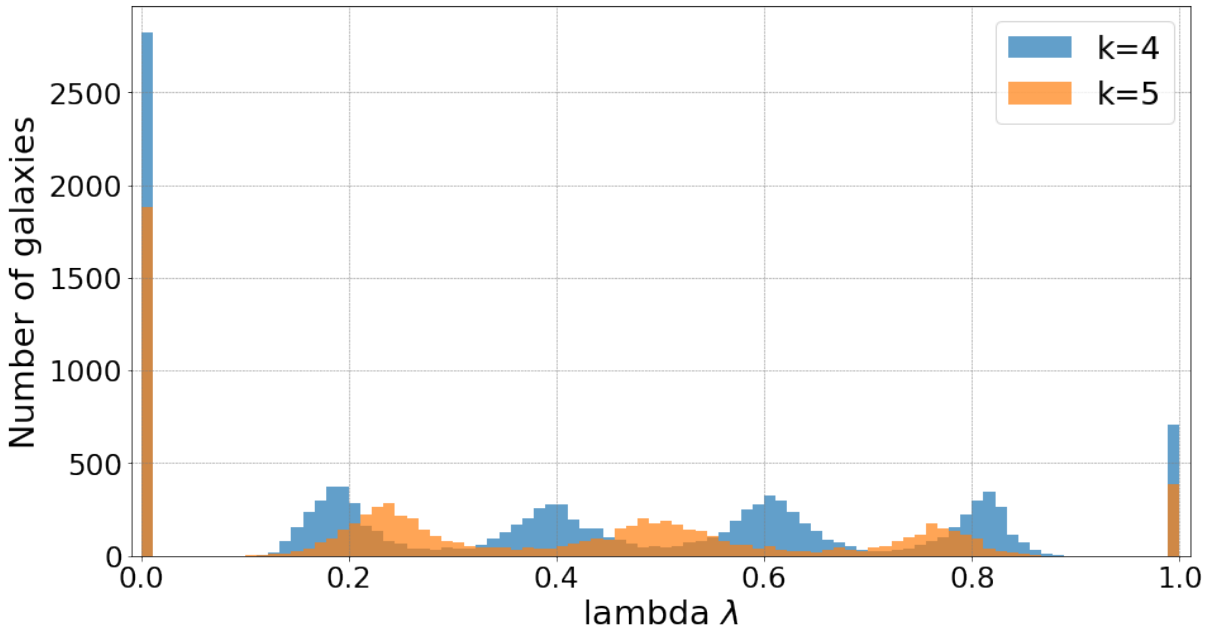


Figura 26: Histograma da probabilidade das galáxias do `VIPERS` serem representativas das galáxias do `DES` em uma folha na região mais populosa do conjunto do `DES`. O histograma tem as probabilidades para ambas as árvores construídas, a de $\texttt{k} = 4$ e $\texttt{K} = 5$. Em ambas há dois picos maiores bem definidos nas extremidades e picos menores entre os picos maiores com uma concentração de galáxias com a probabilidades parecidas.

5. Aprendizados de máquina usados no cálculo do redshift fotométrico

5.1. GPz

O primeiro aprendizado de máquina que utilizamos para calcular o redshift fotométrico foi a chamada `GPz`, que usa processos Gaussianos em seus cálculos. Este programa[9] foi escrito para o cálculo do redshift fotométrico de diversos catálogos astronômicos. Este método considera que as incertezas nas medições fotométricas fazem com que haja pouca acurácia dos valores do redshift calculado. A variância que está associada a estes valores pode ser não uniforme para todas as observações e esse fenômeno estatístico é conhecido como heterocedasticidade [90]. No artigo do Almosallam et al.[14], é apresentada uma forma de aprendizado de máquina que otimiza o modelo considerando a média do valor calculado do redshift fotométrico e a sua variância. Portanto, ao invés de se ter apenas as cores como dados de entrada, a máquina também recebe as incertezas das magnitudes (fornecidas pelos catálogos astronômicos) como *input*.

Um processo gaussiano é um procedimento estocástico em que cada ponto do sistema está associado a uma variável. O conjunto de variáveis obedece a uma distribuição normal multivariada definida por uma matriz de covariância, Σ. Para referências ver, por exemplo, [91] [92].

O código do `GPz` tem como resultado o redshift fotométrico e um erro associado a este redshift para cada galáxia. O redshift fotométrico é calculado através de uma relação matemática entre os *inputs* e o redshift, e o valor das constantes desta equação são calculados na fase de treinamento. O erro do redshift fotométrico é calculado através de uma distribuição gaussiana.

Para melhor visualização do processo do cálculo do redshift fotométrico, considere o conjunto de dados de entrada $\boldsymbol{X} = {\boldsymbol{x}_i}^n = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n] \in \mathbb{R}^{nxd}$ em que cada dado tem `d` dimensões, logo $\boldsymbol{x}_i = (a_1, a_2, ..., a_d)$ com a sendo um valor unidimensional e n é o número de amostras. Estes dados de entrada têm como dados de saída $\boldsymbol{y} = y_{i\,i=1}^{n} \in \mathbb{R}^n$ em que n é o número de amostras.

O modelo mais simples para resolver esse problema seria o de regressão linear, porém esse método só seria válido se a relação entre $\mathbf{X}$ e $\mathbf{y}$ fosse linear. Aqui, usaremos um sistema não linear, onde o valor predito y_i é gerado pela combinação linear de m funções não lineares $\boldsymbol{\Theta}(\boldsymbol{x}_i) = [\phi_1(\boldsymbol{x}_i), \phi_2(\boldsymbol{x}_i), .., \phi_m(\boldsymbol{x}_i)] \in \mathbb{R}^m$ mais o ruído ϵ_i, este ruído é uma

[9] https://github.com/OxfordML/GPz

constante para cada *input* i. O termo $\phi_j(\boldsymbol{x}_i)$ é não linear, dado pela equação definida em Almosallam et al como

$$\phi_J(\boldsymbol{x}_i) = exp\bigg(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{p}_j)^T \Gamma_j^T \Gamma_j (\boldsymbol{x}_i - \boldsymbol{p}_j)\bigg), \tag{45}$$

em que $\boldsymbol{P} = \{\boldsymbol{p}_i\}_{i=1}^m \in \mathbb{R}^{mxd}$ é definido como o conjunto de vetores associados com as funções de base e $\Gamma_j^T\Gamma_j$. $\Gamma_j \in \mathbb{R}^{dxd}$ é uma matriz que tem relação com os autovetores e autovalores da matriz de covariância definida na equação (51). Considerando que d seja o número de dimensões de cada *input*. O código GPz permite diferentes estruturas de Γ_j para as funções ϕ_j. Considerando que $\boldsymbol{x}_i = (a_1, a_2, ..., a_d)$, temos que $\phi_j(\boldsymbol{x}_i) = \phi_j(a_1, a_2, ..., a_d)$.

Com tudo isso em mente, podemos dizer que o dado predito y_i é calculado através da equação

$$y_i = \boldsymbol{\Theta}(\boldsymbol{x}_i)\boldsymbol{\omega} + \epsilon_i \quad , \tag{46}$$

onde $\boldsymbol{\omega} = [\omega_1, \omega_2, ..., \omega_m]$ é um vetor de comprimento `m` função dos parâmetros do modelo. No caso de se predizer o redshift, $\mathbf{X}$ são as medições fotométricas e as suas incertezas com d dados de entrada e n objetos de treinamento. Os valores de $\boldsymbol{\Theta}$ e ω são definidos na fase de treinamento do `GPz` com as galáxias do `VIPERS`.

Para diminuir o custo operacional, foi feita uma aproximação para $\phi_J(\boldsymbol{x}_i)$ usando a transformação linear para calcular os seus parâmetros. A função $\boldsymbol{\phi}$ tem o seu valor dependendo exclusivamente da distância entre o dado de entrada e um ponto fixo. Este ponto pode ser a origem ou um outro ponto definido e fixo.

Observando a equação (46) e escolhendo apenas um dado de entrada $\boldsymbol{x}$, podemos reescrevê-la como

$$y = \phi(\boldsymbol{x})\omega + \epsilon. \tag{47}$$

A equação (47) mostra como calcular o redshift fotométrico para o `GPz`, transformando as variáveis desta equação nas variáveis utilizadas em nosso trabalho,

$$z_{phot} = \phi(\boldsymbol{X})\omega + \epsilon \quad . \tag{48}$$

Para o cálculo do erro do redshift fotométrico, consideramos uma distribuição gaussiana de n dimensões e uma matriz de covariância Σ de tamanho $n \times n$. Desta matriz temos que os valores da diagonal são as variâncias σ_{ii} de cada variável i, enquanto fora da diagonal é a covariância σ_{ij} entre as diferentes variáveis. A matriz Σ é simétrica, $\sigma_{ij} = \sigma_{ji}$. Σ modela a variância ao longo de cada dimensão e determina como as diferentes variáveis estão correlacionadas. $\boldsymbol{\mu}$ é o vetor da média da distribuição gaussiana [93]. Cada compo-

nente do vetor $\boldsymbol{\mu}$ descreve o valor médio da distribuição em cada dimensão. A função da distribuição gaussiana é dada pela equação

$$f(x_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2} \quad , \tag{49}$$

em que μ é a média da distribuição e σ é o desvio padrão. E cada x_i é um valor do vetor $\boldsymbol{X} = (x_1, x_2, ..., x_n)$. A função $f(x)$ da equação (49) também pode ser escrita como $N(\mu, \Sigma)$. Para um conjunto de dados de duas dimensões, podemos definir a distribuição gaussiana como mostrado na equação

$$f(\boldsymbol{X}) = \mathtt{N}(\mu, \Sigma) = \mathtt{N}\left(\begin{bmatrix} \mu_i \\ \mu_j \end{bmatrix}, \begin{bmatrix} \sigma(x_i, x_i) & \sigma(x_i, x_j) \\ \sigma(x_j, x_i) & \sigma(x_j, x_j) \end{bmatrix} \right) \quad . \tag{50}$$

A matriz de covariância Σ descreve o formato da distribuição e pode ser definida por

$$\Sigma = Cov(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) = \mathbb{E}[(f(\boldsymbol{x}_i) - \boldsymbol{\mu}_i)(f(\boldsymbol{x}_j) - \boldsymbol{\mu}_j)^T], \tag{51}$$

para duas variáveis i e j diferentes [94]. O processo gaussiano é representado por $N(\mu, \Sigma)$ que é uma distribuição gaussiana com média μ e variância Σ.

Para obtermos os valores dos componentes da matriz Σ é preciso calcular seus autovalores e autovetores.

Escrevemos $Av_i = \lambda_i v_i$, em que v é um autovetor de A e λ é o autovalor correspondente. Considerando esta fórmula, mas para a matriz C dos elementos da matriz de covariância Σ, podemos escrever a equação para a matriz Σ,

$$\Sigma V = VL \rightarrow \Sigma = VLV^{-1} \quad . \tag{52}$$

Acima, colocamos os autovetores de Σ nas colunas de uma matriz V e todos os autovalores como as entradas da diagonal da matriz L. Este é o procedimento usual de diagonalização de uma matriz.

Os autovetores são vetores unitários que representam as direções da maior variação em cada dado enquanto os autovalores representam a relevância dessa variação nas direções correspondentes. Isto significa que V é uma matriz ortogonal, $V^{-1} = V^T$ e L uma matriz diagonal, podemos dizer que $L = L^T$. Chamando $D = V\sqrt{L}$ podemos escrever

$$C = V\sqrt{L}\sqrt{L}V^{-1} = D(D)^T \tag{53}$$

Dessa forma, conseguimos calcular a matriz de covariância através de uma transformação linear utilizando os autovetores e os autovalores. O programa terá que determinar as matrizes citadas nas equações acima para cada elemento da matriz de covariância. Po-

rém, todo esse cálculo tem um custo operacional e de armazenamento alto, principalmente por conta da inversão de matrizes.

Para cada treinamento, é preciso determinar os valores das matrizes diversas vezes, em especial na fase de otimização, para ter o melhor desempenho do aprendizado de máquina. Além disso, estamos lidando com amostras de muitas galáxias, o que implica em matrizes de dimensões na casa dos milhares. Para tentar diminuir esse custo de funcionamento, o código utiliza algumas aproximações. Este processo pode fazer com que o parâmetro calculado na otimização seja instável, causando incerteza no valor do dado de saída. Em Almosallam et al [14] usam-se modelos para melhorar a acurácia da variância sem comprometer o custo computacional dos processos gaussianos [95].

Para ilustrar como funciona o processo do `GPz`, vamos considerar um exemplo em que utilizamos os termos definidos anteriormente para um conjunto de dados mais simples. Consideremos os conjuntos $\mathbf{X}$ e $\mathbf{y}$ definidos como

$$\begin{aligned} \boldsymbol{X} &= (2.220, 8.707, 2.067, 9.186, 4.884, 6.117, 7.659, 5.184, 2.968, 1.877) \quad , \\ \boldsymbol{y} &= (0.078, 0.739, 0.186, 0.661, -0.564, 0.496, 0.996, 0.797, 0.217, 0.378) \quad , \end{aligned}$$

em que $\boldsymbol{X}$ é o conjunto de dados de entrada do `GPz`, os dados fotométricos das galáxias. A descrição dos dados utilizados está na seção 5.1.1.

O cálculo do conjunto de dados m_Y é dado pela função $b(x) = sen\big((x-2,5)^2\big)$. A escolha da função foi feita de forma arbitrária para que os pontos ficassem visíveis no gráfico.

O processo gaussiano aplicado no aprendizado de máquina parte do pressuposto de que há uma distribuição gaussiana de probabilidade a priori (*prior*) denominada como $m(x)$ para cada dado do conjunto de treinamento. Considerando a equação (47), queremos achar a função $g(x) = \phi(x)\omega$. Para isso é preciso definir o *prior*, ou seja, a função do valor esperado $m(x)$ e a função de covariância k do processo gaussiano.

Para Σ vamos usar a função de covariância exponencial quadrática definida por

$$\delta(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2} \right) \quad , \tag{54}$$

em que $d(x_i, x_j)^2$ é a distância euclidiana entre os pontos x_i e x_j do dado de entrada $\boldsymbol{X}$. O termo l é um valor real de escalada da função $\delta(x_i, x_j)$. O padrão de l é ser igual a 1, porém é possível modificar o seu valor de acordo com a necessidade do usuário. Se pensarmos em um dado de entrada com 4 parâmetros (como as cores), $d(x_i, x_j)$ é dado

pela equação (33). No caso do exemplo, como estamos lidando com dados de entrada de 1 dimensão, teremos

$$\delta(x_i, x_j) = \exp\left(- \frac{(x_i - x_j)^2}{2l^2} \right) \quad . \tag{55}$$

Em nosso trabalho, lidamos com conjuntos de dados contendo 5 dimensões. Aqui escolhemos apenas uma dimensão para melhor visualização na figura. Então a matriz de covariância será dada por

$$\Sigma = \begin{bmatrix} \delta(x_1, x_1) & \delta(x_1, x_2) & \cdots & \delta(x_1, x_{10}) \\ \delta(x_2, x_1) & \delta(x_2, x_2) & \cdots & \delta(x_2, x_{10}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(x_{10}, x_1) & \delta(x_{10}, x_2) & \cdots & \delta(x_{10}, x_{10}) \end{bmatrix} \tag{56}$$

O cálculo da *priori* é definido em Almosallam et al. [14] como $m(x) = k(x)^T \Sigma \boldsymbol{y}$. Em que $k(x_i) = \delta(x_i, x)$ [96] pode ser calculada como

$$k = \begin{bmatrix} \delta(x_1, x) \\ \delta(x_2, x) \\ \vdots \\ \delta(x_{10}, x) \end{bmatrix} = \begin{bmatrix} \exp\left(- \frac{(x_1 - x)^2}{2l^2} \right) \\ \exp\left(- \frac{(x_2 - x)^2}{2l^2} \right) \\ \vdots \\ \exp\left(- \frac{(x_{10} - x)^2}{2l^2} \right) \end{bmatrix} \quad . \tag{57}$$

Por fim, temos que a função do valor esperado será dada por

$$m(x) = \begin{bmatrix} \exp\left(- \frac{(x_1 - x)^2}{2l^2} \right) & \exp\left(- \frac{(x_2 - x)^2}{2l^2} \right) & \cdots & \exp\left(- \frac{(x_{10} - x)^2}{2l^2} \right) \end{bmatrix} \times \tag{58}$$

$$\begin{bmatrix} \delta(x_1, x_1) & \delta(x_1, x_2) & \cdots & \delta(x_1, x_{10}) \\ \delta(x_2, x_1) & \delta(x_2, x_2) & \cdots & \delta(x_2, x_{10}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(x_{10}, x_1) & \delta(x_{10}, x_2) & \cdots & \delta(x_{10}, x_{10}) \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{10} \end{bmatrix} \quad . \tag{59}$$

Na figura 27 podemos ver 5 possíveis funções (linhas tracejadas) de $m(x)$ e o desvio padrão na distribuição *prior* do conjunto de dados. A diferença entre as funções é o valor real do termo l fixado da equação (54). Isto significa que os dados de treinamento ainda não foram usados para calcular os parâmetros destas funções. A linha preta representa distribuição final se usássemos as funções das linhas coloridas como base para a distribuição. A área cinza é a incerteza. Os pontos vermelhos são os 10 dados originais fornecidos à máquina, mas que ainda não foram utilizados.

Os dados de treinamento fornecem à máquina o valor dos dados de entradas e as suas incertezas. No caso deste trabalho, são fornecidas as informações fotométricas e as incertezas das medições fotométricas disponibilizados pelos catálogos. Com estas informações, podemos montar uma função gaussiana da probabilidade [97].

Com os dados de entrada, a máquina otimiza, então, os valores dos parâmetros da função $m(x)$ e calcula todas as probabilidades a serem consideradas. Após treinarmos a máquina, teremos a figura 28, em que cada uma das funções está melhor ajustada aos pontos fornecidos. A área da incerteza diminuiu consideravelmente, sendo maior em pontos em que não havia dado de entrada para fixar os valores exatos para x. A linha preta é a função final, no caso da equação (54) seria $g(x)$.

Com a nossa máquina treinada e otimizada, é possível calcular o *output* para qualquer *input*. Para cada novo dado de entrada (ponto verde na figura 28) x_t fornecido à máquina, será dado como resultado o valor de y_t e a sua incerteza neste ponto.

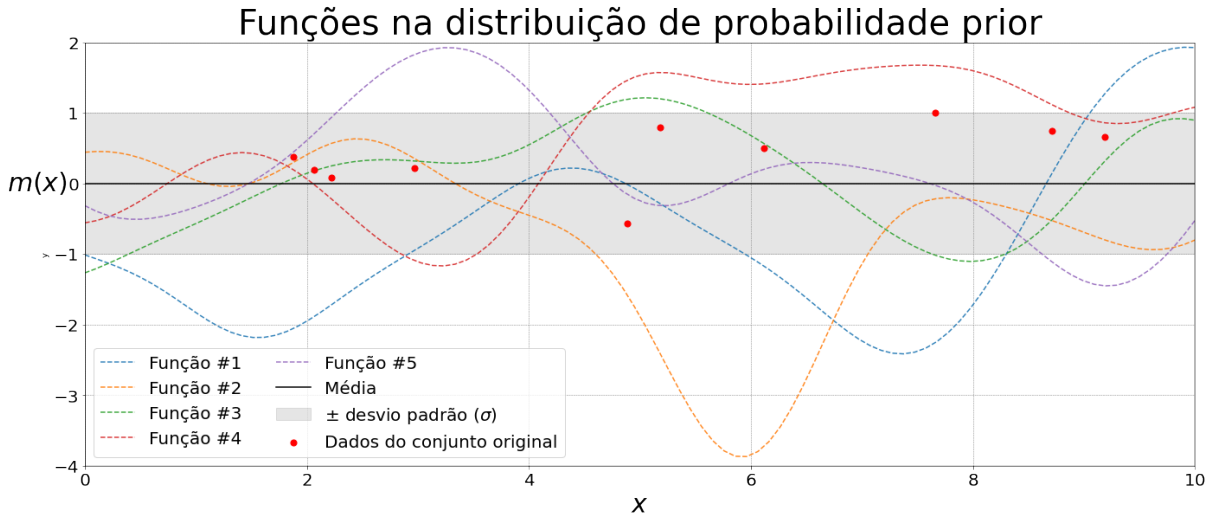


Figura 27: Figura ilustrativa das funções da distribuição de probabilidade *prior*. As curvas tracejadas são as diferentes funções $m(x)$ que podem ser utilizadas para este conjunto de dados e catalogadas na legenda. A diferença entre as funções é o valor de l fixado na função $\delta(x_i, x_j)$. A linha preta é a média de todas estas funções e a área cinza é o desvio padrão desta linha preta. Os pontos em vermelho são os pontos do conjunto de dados ($\mathbf{X}$ e $\mathbf{y}$).

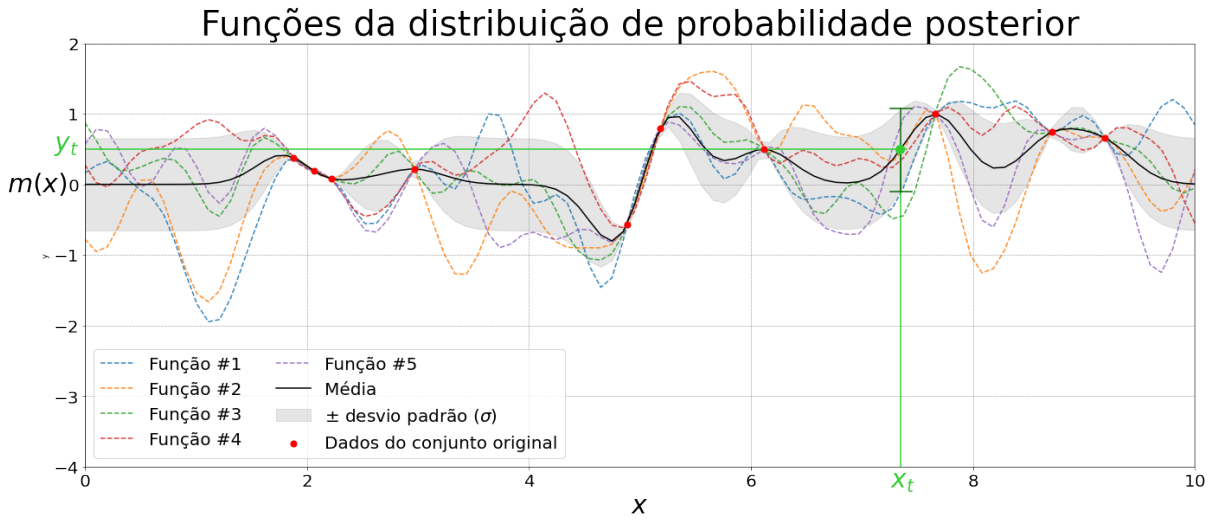


Figura 28: Figura ilustrativa das funções da distribuição de probabilidade posterior. As curvas tracejadas são as diferentes funções $m(x)$ que podem ser utilizadas para este conjunto de dados e catalogadas na legenda. A linha preta é a média de todas estas funções e a área cinza é o desvio padrão desta linha preta. Os pontos em vermelho são os pontos do conjunto de dados ($\mathbf{X}$ e $\mathbf{y}$). O ponto verde é o dado de entrada x_t e o y_t resultado fornecido pela máquina. A linha verde mais escura é a incerteza para o ponto m_t.

5.1.1. Dois tipos de dados de entrada

Foram treinados dois aprendizados de máquina utilizando o código do `GPz`. Para cada uma das máquinas foram utilizados tipos de dados de entrada diferentes. Na primeira máquina, intitulada `GPz- Magnitude`, os dados de entrada são as 5 magnitudes nos filtros m_g, m_r, m_i, m_z e m_Y. Na segunda máquina, denominada `GPz - Color`, o tipo de dado de entrada são as 4 cores $m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$ e a magnitude no filtro `i`.

A escolha de se usar dois tipos de dados de entrada deve-se à limitação da magnitude do catálogo do `VIPERS`. Este catálogo limitou os seus objetos a galáxias com magnitude menor que 22,5 no filtro `i` ($m_i \leq 22,5$). Esta limitação da magnitude pode acarretar na perda de algumas informações fotométricas na banda `i`. O catálogo do `DES` não tem esta limitação, havendo neste catálogo galáxias com magnitudes no filtro `i` maiores que 22,5. A disparidade entre as magnitudes do filtro `i` em cada catálogos é representada na figura 29.

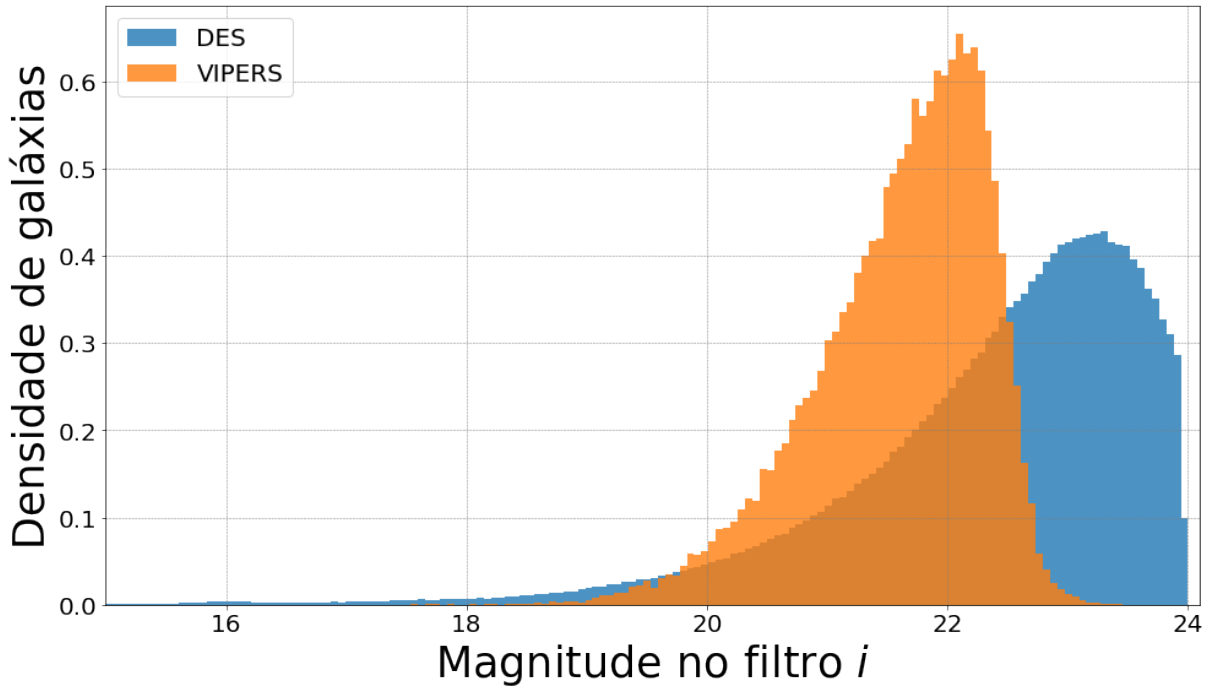


Figura 29: Histogramas da concentração de galáxias de cada catálogo em função da magnitude no filtro `i`. Foi utilizada a densidade da concentração de galáxias, pois o catálogo do `DES` é significativamente maior que o catálogo do `VIPERS`, como descrito no capítulo 3. A curva azul representa as galáxias do `DES` e a curva laranja representa as galáxias do `VIPERS`.

Para que possamos calcular o redshift fotométrico com o menor erro possível, é necessário utilizar um dado de entrada que contenha as informações dessas galáxias para todo o espectro de magnitudes do filtro `i` possíveis. Isto significa que, para galáxias do `DES` que tem um alcance da magnitude no filtro `i` entre 15 e 24, é preciso um conjunto de

treinamento que tenha este mesmo alcance. Ao observarmos a figura 29, podemos ver que há um número considerável de galáxias do DES cuja magnitude é maior do que $22,5$ (cerca de 55%). Portanto, mais da metade do catálogo do DES tem uma magnitude diferente do conjunto de treinamento (galáxias do VIPERS). A consequência desta desproporção será uma máquina que não terá sido bem treinada para metade das galáxias, já que o conjunto de treinamento não será representativo.

Para evitar o problema da representatividade, utilizamos as cores ($m_g - m_r$, $m_r - m$, $m_i - m_z$ e $m_z - m_Y$) no lugar das magnitudes. A razão desta substituição é representada na figura 30, ao contrário da imagem 29, em que vemos um deslocamento da concentração das galáxias dos dois catálogos; ao utilizarmos as cores, temos que o conjunto de treinamento está no mesmo intervalo que as galáxias do DES. Isto indica que as galáxias do VIPERS são representativas daquelas do DES.

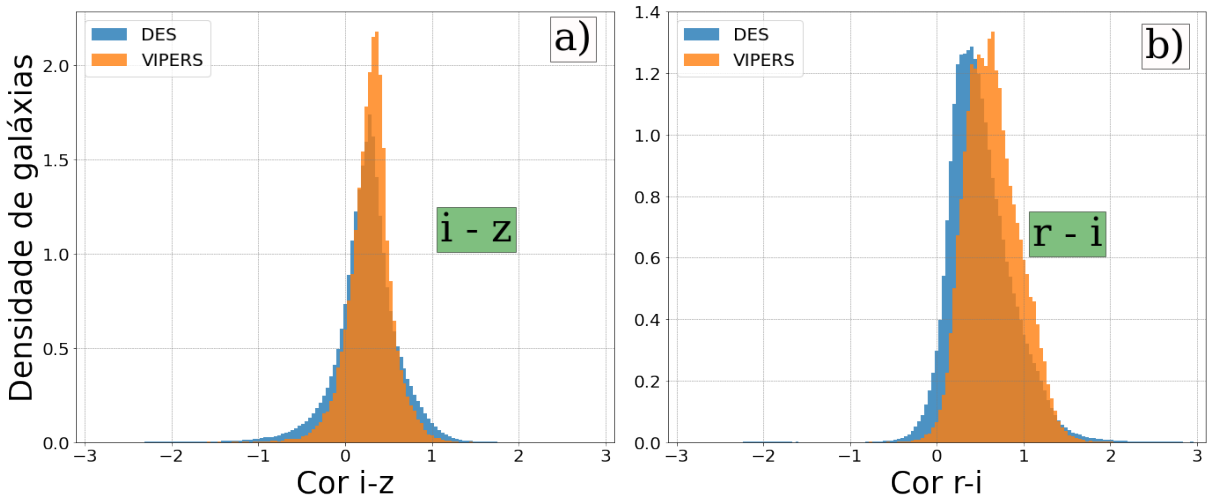


Figura 30: Histogramas da concentração de galáxias de cada catálogo em função das cores $m_i - m_z$ e $m_r - m_i$. Foi utilizada a densidade da concentração de galáxias pois o catálogo do DES é significativamente maior que o catálogo do VIPERS, como descrito no capítulo 3. A curva azul representa as galáxias do DES e a curva laranja representa as galáxias do VIPERS. Na imagem 30a, temos a concentração de galáxias em função da cor $m_i - m_z$ e, na imagem 30b, temos a concentração de galáxias em função da cor $m_r - m_i$.

Como o catálogo do VIPERS é limitado apenas na magnitude do filtro i, as outras magnitudes deste catálogo estão no mesmo intervalo que as galáxias do DES. Isto significa que há informações das magnitudes m_g, m_r, m_z e m_Y do catálogo do VIPERS que podem ser utilizadas no cálculo do redshift fotométrico. Ao utilizarmos apenas as cores, estas informações não são contempladas no treinamento do aprendizado de máquina. Por isso, foram utilizados os dois tipos de dados de entrada, para que possamos comparar o resultado das máquinas treinadas com as 5 magnitudes e as 4 cores e a magnitude m_i.

Quando são usadas apenas as 4 cores e a magnitude m_i, há uma perda de informação das magnitudes dos filtros m_g, m_r, m_z e m_Y. Por outro lado, ao usarmos 5 magnitudes, apesar de termos mais informação das outras magnitudes para o aprendizado de máquina,

um dos parâmetros tem menos informação (magnitude do filtro `i`). Uma ideia seria utilizar as 4 cores e as 5 magnitudes, mas isso acarretaria em um ganho de complexidade à máquina (com a adição de um parâmetro a máquina torna-se mais complexa, demorando mais tempo para treinar). O ganho qualitativo é muito baixo.

Ao usarmos as 4 cores, colocamos também a magnitude no filtro `i`, justamente porque essa magnitude é mais profunda, ou seja, consegue captar informações do espectro de galáxias mais distantes. Isso pode ser visto na figura 1, em que o filtro `i` é o que tem a transmissão relativa mais alta, chegando perto de 1. O filtro `z` também apresenta uma transmissão alta e comparável com a do filtro `i`, porém a sua transmissão varia bastante, indo até abaixo de 0,8, enquanto a transmissão do filtro `i` é mais estável.

O erro na `GPz- Magnitude` foi a incerteza na magnitude fornecida pelo catálogo do `VIPERS`, enquanto que, para o `GPz - Color`, foi feita uma conta para calcular o erro da cor considerando a incerteza da magnitude. Por exemplo, para a cor $m_g - m_r$ temos que considerar o erro na magnitude m_g (err_{m_g}) e, na magnitude m_r (err_{m_r}), o erro desta cor ($err_{m_g - m_r}$) será dado por

$$err_{m_g - m_r} = \sqrt{err_{m_g}^2 + err_{m_r}^2} \quad . \tag{60}$$

A implementação do `GPz` foi a primeira que fizemos das 3 máquinas deste trabalho. Por isso, decidimos testar os dois tipos de dados de entrada neste código. Após o treinamento das duas máquinas, sendo cada máquina com um *input* diferente, medimos os resultados a partir do erro quadrático médio. A figura 31 mostra o gráfico de dispersão do resultado de ambas as máquinas.

Observando a figura 31, é possível ver que ambas apresentam o padrão de se sobreporem à reta ideal quando as galáxias têm redshifts dentro do intervalo $0,4 < z < 1,0$. Para valores fora deste intervalo, os pontos se distanciam da reta ideal. Isto acontece pois o conjunto de treinamento (galáxias do `VIPERS`) tem uma maior concentração de galáxias dentro do intervalo, e isto significa que as máquinas têm dados o suficientes para criar um padrão para o cálculo do redshift.

Pela figura 31, o gráfico do `GPz - Magnitude` parece ter um menor erro dentro do intervalo $0,4 < z < 1,0$, porém esta é uma visão baseada apenas na qualidade das curvas. Para quantificar a diferença entre os resultados, calculamos a média do erro quadrático médio (EQM) para cada *bin* e colocamos na figura 32 em função do redshift espectroscópico.

A figura 32 mostra que a diferença entre os resultados foi pequena, porém quantitativamente a máquina `GPz - Color` apresentou um menor erro quadrático médio. Esta diferença é maior dentro do intervalo $0,4 < z < 1,0$, em que o erro se manteve perto de 0. Já para a máquina `GPz - Magnitude`, o erro se afastou do 0 mais rapidamente do que a máquina do `GPz - Color`.

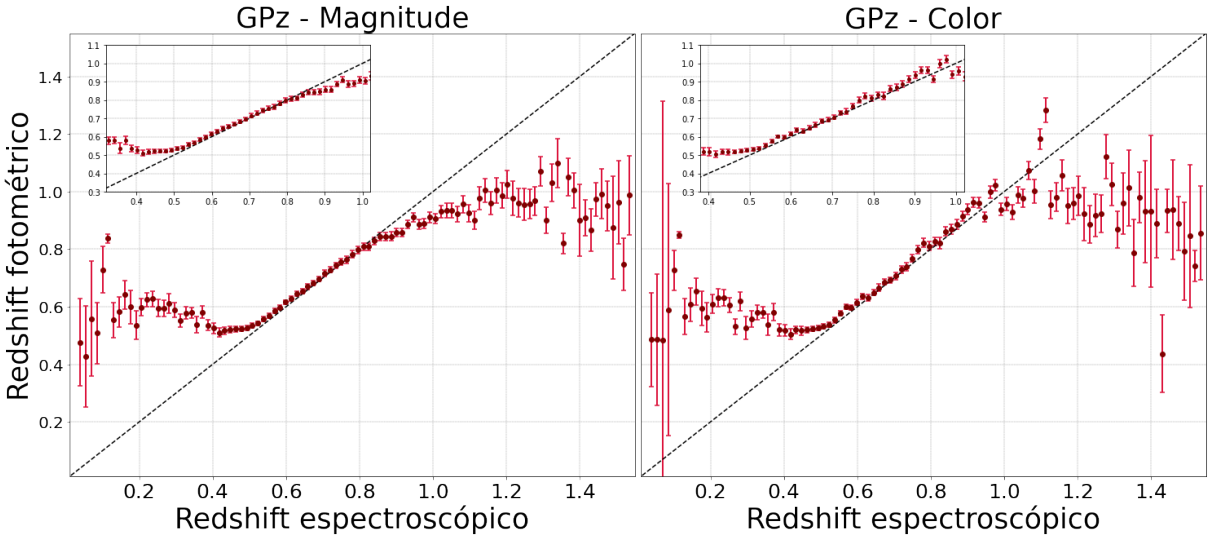


Figura 31: Gráficos dos resultados do `GPz` por conta dos *inputs* em função do redshift espectroscópico. A primeira imagem contém os redshifts fotométricos calculados pelo `GPz` treinado com as 5 magnitudes e a segunda imagem os z_{phot} calculados pelo `GPz` treinado com as 4 cores e a magnitude m_i. A linha pontilhada preta é a reta ideal do redshift, se as máquinas fossem perfeitas, ou seja, se o resultado fosse $z_{phot} = z_{spec}$ para todas as galáxias, este seria o produto da máquina. As galáxias foram divididas em 100 segmentos (*bins*) em função do redshift espectroscópico e foi calculada a média dos redshifts fotométricos em cada seção. A média dos z_{phot} é o que está sendo plotado no eixo y da figura. As barras verticais são as médias dos erros dos redshifts fotométricos em cada segmento.

Com base neste estudo, concluímos que utilizar as 4 cores e a magnitude no filtro `i` ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$ e m_i) resultou em um erro menor dos redshifts fotométricos. Então, para as próximas duas máquinas (`Keras` e `ANNz`), utilizamos este mesmo *input*.

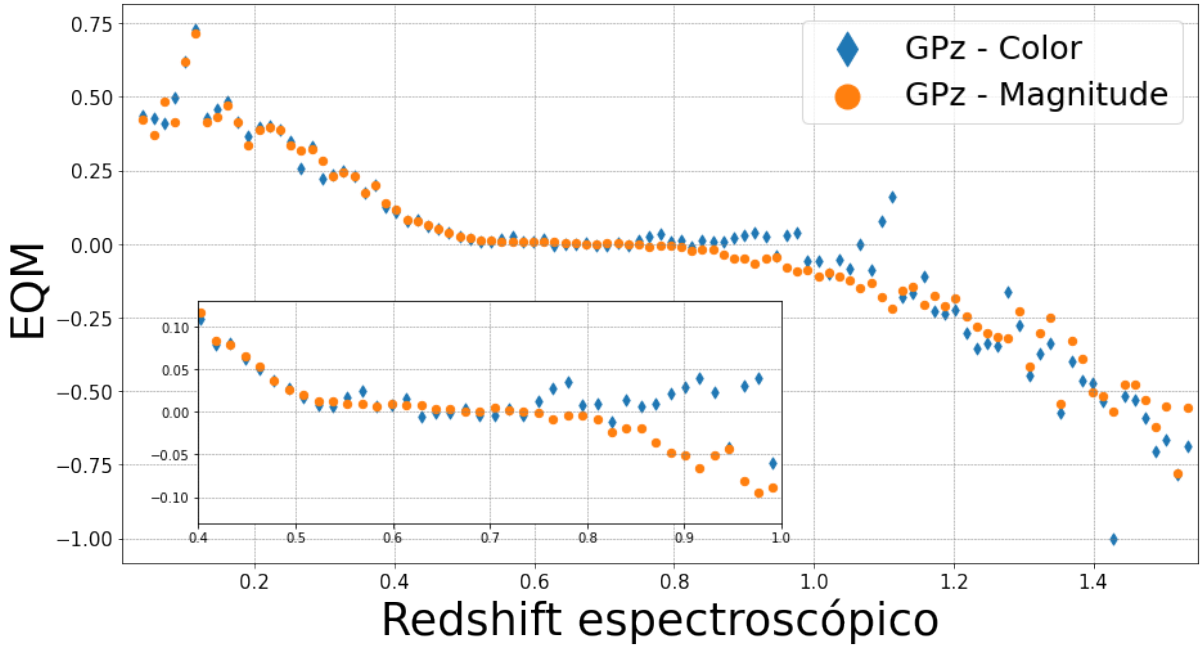


Figura 32: Gráfico do erro quadrático médio (`EQM`) dos resultados do `GPz` em função do redshift espectroscópico. As pontos azuis são os resultados do `GPz - Color` e os pontos laranjas são os produtos do `GPz - Magnitude`. As galáxias foram divididas em 100 *bins* em função do redshift espectroscópico e foi calculada a média dos `EQM` em cada seção. Este resultado é o que foi plotado no eixo y da figura.

5.1.2. Testes de métodos

A fim de saber qual o melhor método a utilizar para o cálculo do redshift fotométrico, foi feito um estudo, dos diferentes modelos disponíveis no código do GPz. Para este estudo, o aprendizado de máquina foi treinado com os seis modelos à disposição pelo código e depois foi calculado o erro deles de forma a verificar qual algoritmo nos dava um *output* com menor erro em relação ao dado de saída do conjunto de treinamento fornecido à máquina.

O diferencial entre os modelos é a estrutura da matriz de precisão Γ_j (equação 45) em cada uma das funções de base. Abaixo colocamos uma explicação da diferença entre estes modelos.

O primeiro modelo é denominado Processos Gaussianos com covariâncias variáveis (*Gaussian processes with variable covariances* - `GPVC`), em que Γ_j é a matriz de precisão associada a cada função de base. Neste caso as matrizes de covariância não são iguais, e isto significa que, para cada função de base radial $\phi_J(\boldsymbol{x}_i)$, haverá uma matriz de precisão Γ_j diferente.

O segundo modelo se chama Processos Gaussianos com covariâncias globais (*Gaussian processes with a global covariance* - `GPGC`), no qual $\Gamma_j = \Gamma_k = \Gamma$, ou seja, a matriz Γ de precisão é igual para todas as funções de base. Todas as funções de base radiais $\phi_J(\boldsymbol{x}_i)$ terão a mesma matriz de precisão Γ_j.

O terceiro modelo se chama Processos Gaussianos com diagonais de covariâncias variáveis (*Gaussian processes with variable diagonal covariances* - `GPVD`), em que Γ_j é uma matriz diagonal para cada função de base ϕ. Os valores fora da diagonal serão nulos.

O quarto modelo se chama Processos Gaussianos com a diagonal das covariâncias global (*as Gaussian processes with global diagonal covariance* - `GPGD`). Γ tem a mesma diagonal para todas as funções de base. Isso implica em $\phi_J(\boldsymbol{x}_i)$ ter a mesma diagonal em Γ para todos os j. Os valores fora da diagonal são nulos.

O quinto modelo se chama Processos Gaussianos com escala variável (*Gaussian processes with a variable length-scale* - `GPVL`) ou Processos Gaussianos com covariância isotrópica para cada função de base j. Neste modelo, a matriz de precisão é proporcional à matriz identidade, $\Gamma_j = \boldsymbol{I}\gamma_j$, em que γ_j é a variável de escala e $\boldsymbol{I}$ é a matriz identidade. Portanto, Γ_j terá os valores fora da diagonal nulos e os que estão na diagonal serão iguais a γ_j para cada função de base ϕ_j.

O sexto modelo se chama Processos Gaussianos com escala global (*Gaussian processes with a global length-scale* - `GPGL`) ou Processos Gaussianos com covariância isotrópica para todas as funções de base. Este modelo é análogo ao modelo `GPVL`, porém com a variável de escala γ_j sendo igual em todas as funções de base. Neste caso, $\Gamma_j = \boldsymbol{I}\gamma$, ou seja, todas as matrizes de precisão serão proporcionais à matriz identidade multiplicada pela variável de escala γ.

Testando todos os modelos citados conseguimos obter os gráficos das figuras 33 e 34.

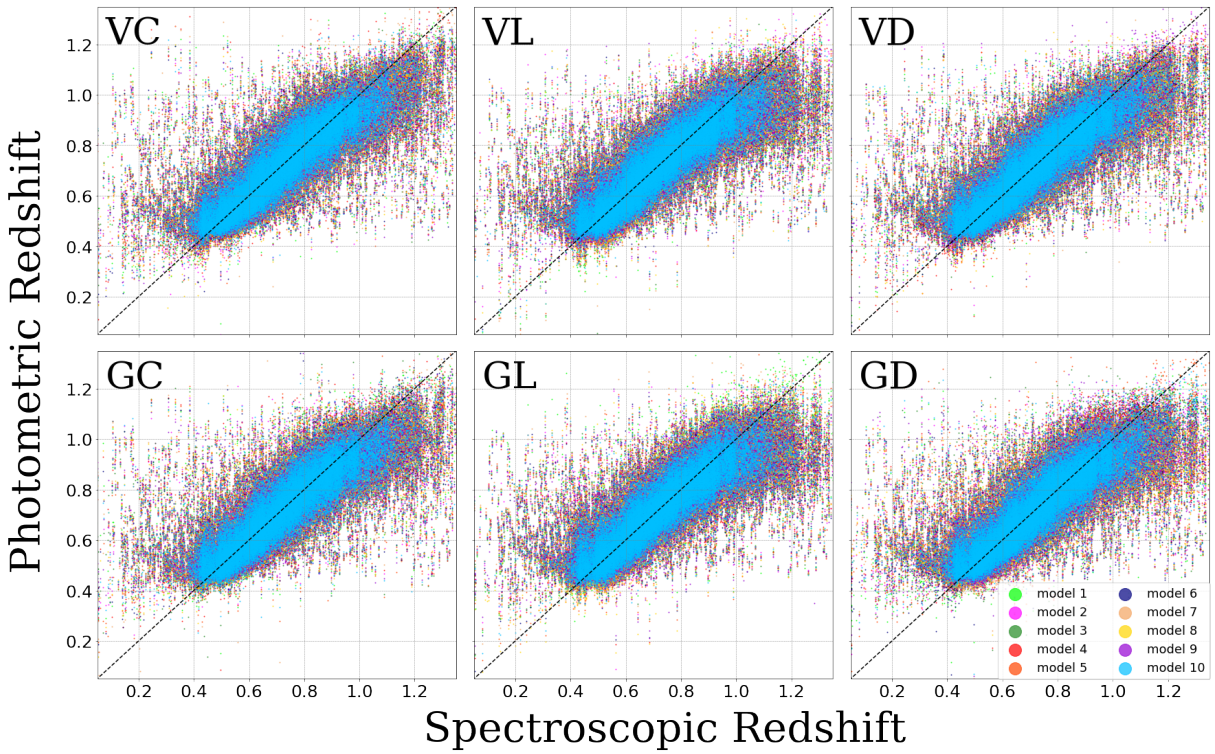


Figura 33: Gráficos de dispersão dos 10 treinamentos de cada método do GPz. Cada máquina foi treinada com um método 10 vezes, pois para cada método temos 10 modelos de aprendizados de máquina distintos. Os gráficos de dispersão de cada cor correspondem a um modelo indicado na legenda da figura.

Na figura 33 podemos ver os gráficos de dispersão de cada um dos 6 métodos. O programa `GPz` foi treinado 10 vezes em cada caso. A diferença entre os vários treinamentos são os valores iniciais dos parâmetros atribuídos às amostras. O código define de modo aleatório os valores das matrizes do programa e, com cada treinamento e otimização, ajusta estes valores, de forma que a máquina tenha um melhor desempenho. Esses valores iniciais aleatórios são gerados através de uma função do python.

A linha preta da figura 33 mostra a máquina ideal, a situação em que a máquina conseguiu calcular os valores do redshift fotométrico corretamente. É possível ver que há pouca diferença entre os métodos. Para facilitar a visualização, também fizemos um histograma dos modelos junto com o redshift espectroscópico.

Nos gráficos da figura 34, é possível ver uma diferença maior entre os métodos. Por exemplo, o histograma do `VIPERS` mostra que há um pico de galáxias com redshift entre 0,5 e 1, sendo um pico mais acentuado perto de $z = 0.7$. Alguns métodos não tiveram esse pico mais acentuado, mostrando uma distribuição mais uniforme do redshifts. Apesar disso, nenhum método apresentou uma diferença significativa dos dados calculados. Por fim, decidimos utilizar o método `GPVC` para o nosso catálogo de redshifts fotométricos por ser o padrão do código.

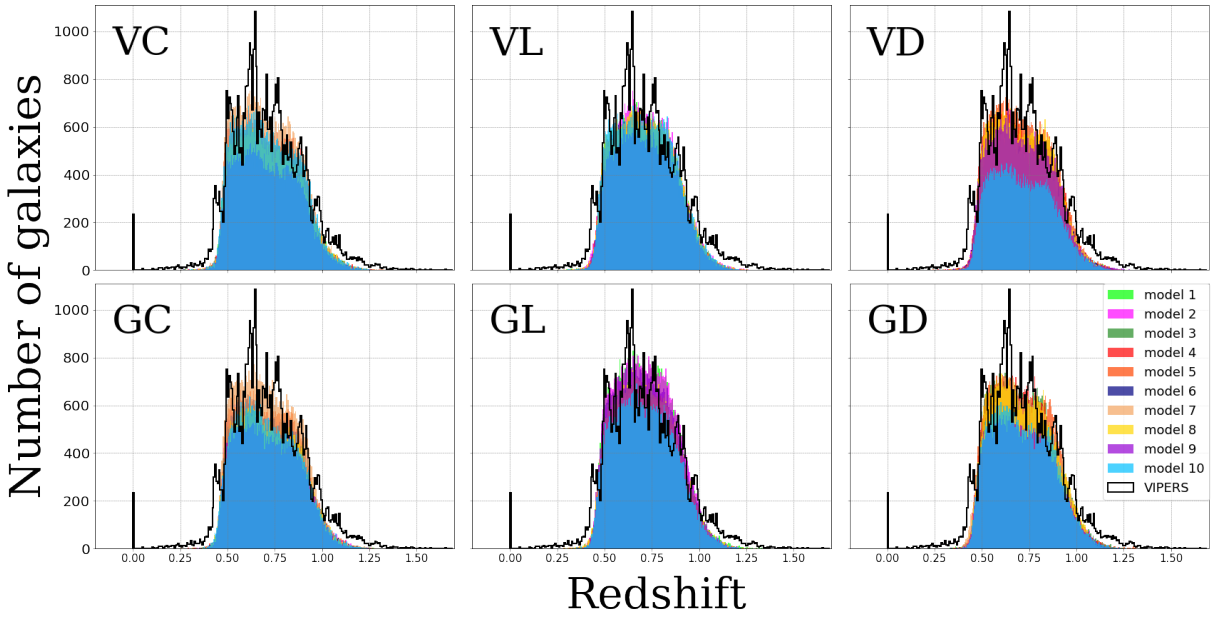


Figura 34: Histograma dos treinamentos do GPz para os seis métodos testados. Cada máquina foi treinada com um método 10 vezes, e, para cada método, temos 10 modelos de aprendizados de máquina distintos. Os histogramas de cada cor correspondem a um modelo indicado na legenda da figura.

5.1.3. O código

A figura 35 é um fluxograma de como funciona o código para o cálculo do redshift através do `ANNz`. Uma explicação mais detalhada pode ser vista no apêndice B.

Na primeira etapa do código, são definidos os parâmetros que serão considerados no aprendizado de máquina. Um deles é o número de iterações. Este valor é o número máximo de vezes em que a máquina irá rodar para calibrar os seus pesos; no nosso caso, escolhemos esse número como 200. Após esse número, a máquina será considerada pronta para ser testada. Outra definição dada é o método a ser utilizado na máquina. As opções de métodos são detalhadas na seção 5.1.2.

A próxima fase é o momento em que os dados são separados em três conjuntos para serem usados na máquina. O primeiro, de treinamento para a máquina aprender com os dados. O segundo, de validação que serve para a máquina aperfeiçoar os valores do seu peso. O último conjunto é o de teste, para verificarmos se os resultados do código estão parecidos com os reais.

Na fase de treinamento, o programa é posto para treinar 10 máquinas diferentes. O que difere entre as máquinas são os valores iniciais dos parâmetros. A máquina usará esses valores para aprender os padrões dos dados e refinar os parâmetros até retornar redshifts fotométricos parecidos com o valor real. Em tese, apesar de todas as máquinas começarem com valores de parâmetros diferentes, a ideia é que estes números convirjam para valores específicos e iguais em todas as máquinas. Isso aconteceria porque deveria haver uma máquina perfeita que devolveria um resultado igual ao real. Como tal mecanismo não

existe, os valores dos pesos de cada máquina continuam a se diferenciar, porém com valores bem menores do que os do começo.

A etapa seguinte é a que calcula os redshifts fotométricos das 10 máquinas e a sua média. Esta média é o resultado que colocaremos no catálogo. A média foi utilizada porque, dessa forma, evitamos ter valores muito discrepantes de z_{phot}, ou seja, se uma máquina não tiver conseguido aprender os padrões dos dados fornecidos de forma eficaz, a média consegue diminuir a importância dessa máquina com valores fora da curva. Assim, o resultado é a combinação do dado de saída de todas as máquinas. Além disso, nessa etapa, os 10 modelos de aprendizado de máquina são salvos em nosso sistema; assim, quando formos rodar para o catálogo do `DES`, poderemos usar as máquinas já treinadas e com os seus parâmetros já definidos.

Por fim, para criarmos o nosso catálogo de galáxias do `DES` com o redshift fotométrico calculado pelo `GPz`, iniciamos a última fase. Nesta etapa, os catálogos das galáxias do `DES` são baixados. As 10 máquinas rodam estas galáxias e calculam os redshifts fotométricos. O programa calcula a média e a incerteza destes resultados. Com tudo isso finalizado, o programa cria um catálogo das galáxias do `DES` em cada pixel com o resultado de cada máquina e a média desse resultado. Este catálogo é salvo com o nome do pixel. Uma demonstração deste catálogo pode ser visto na tabela 8 para o *input* com as 5 magnitudes e, na tabela 7, para o dado de entrada com as 4 cores e a magnitude m_i.

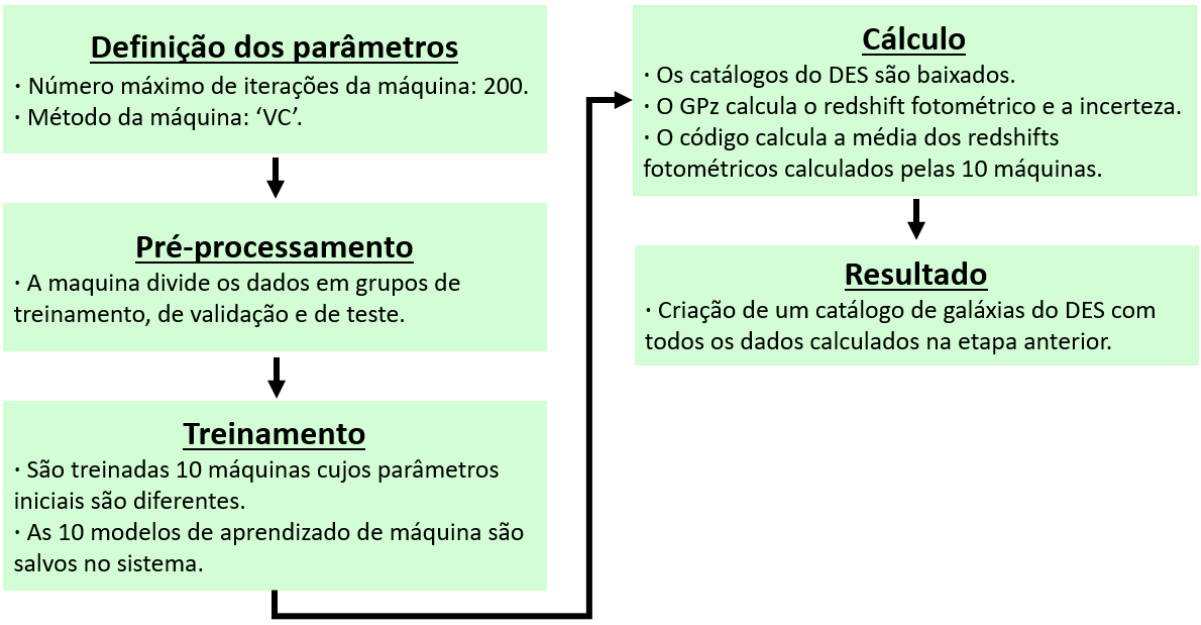


Figura 35: Diagrama das etapas do código do GPz. As setas pretas indicam o caminho percorrido para o cálculo dos redshift fotométrico e as suas incertezas para as galáxias do `DES`.

5.2. ANNz

O programa `ANNz`[98] visa calcular o redshift fotométrico utilizando redes neurais artificiais (*Artificial Neural Networks*). Este código foi introduzido ao público em 2004 e atualizado em 2016, com o lançamento do segundo programa chamado `ANNz2` [15]. Neste estudo, foi utilizada a versão mais recente, que inclui a distribuição de funções de probabilidade `PDF`.

As Redes Neurais artificiais são descritas em mais detalhes na seção 2.1.3 desta tese. Neste código, foram utilizadas redes neurais de multicamadas, ou seja, redes neurais com diversas camadas na seção intermediária.

Para o cálculo do erro, foi utilizada uma função que utiliza o algoritmo do menor erro quadrático médio. Este erro é calculado da forma mostrada na equação

$$EQM(\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)^2 \quad , \tag{61}$$

em que $\hat{y}$ é o valor do redshift calculado pela rede neural e y é o valor do redshift espectroscópico que já temos. O programa tem como objetivo achar uma rede neural que tenha o menor valor $EQM(\hat{y})$. Este processo é feito no momento em que é usado o conjunto de dados de validação, ou seja, a rede já foi criada e treinada e agora usa os valores do conjunto de validação para regularizar e otimizar a sua configuração.

O uso do EQM também ajuda a identificar se a rede neural está fazendo sobre-ajuste. Para isso, compara o valor do erro nas amostras de treinamento e nas de validação. Se o erro do conjunto de treinamento for muito menor que o de validação, é um indicativo de que a rede foi sobre-treinada, ou seja, a rede apenas decorou os valores do redshift e não aprendeu e criou um padrão a partir deles. A fim de evitar o excesso de treinamento, há também testes de convergência que consistem em checar se a máquina continua a melhorar com os ciclos de treinamento ou se está estável. No caso de estar constante, o programa para de treinar e aceita que esse será o melhor resultado para o treinamento. Por fim, há a regularização bayesiana que penaliza a rede neural se a máquina tiver uma estrutura muito complicada, ou seja, se tiver muitos graus de liberdade. O termo "graus de liberdade"engloba o número de camadas, os valores dos pesos e as funções de ativação da máquina.

O programa foi criado com o uso das amostras de galáxias do décimo lançamento (DR10) de catálogos do SDSS [11], incluindo as medidas espectroscópicas do BOSS [99]. Como dados de entrada foram utilizadas as magnitudes nos filtros m_u, m_g, m_r, m_i e m_z, cujos valores foram calculados a partir das medidas de fluxo e utilizando a equação (1). Este conjunto de dados tem cerca de 180 mil objetos.

Para medir a performance do `ANNz`, foram utilizadas diversas métricas. Estes cálculos consideram os pesos dos parâmetros por objeto. A máquina divide os dados de entrada em

subconjuntos e pode optar por colocar pesos em cada subconjunto de galáxias com base em um grau de confiança determinado pela máquina. Isso implica em certos subgrupos terem mais peso no desenvolvimento do aprendizado de máquina na fase de treinamento do que outros subgrupos.

As métricas utilizadas são:

- `Viés fotométrico` (*photometric bias*): $\delta_{gal} = z_{phot} - z_{spec}$ para cada galáxia.
- σ - `Dispersão fotométrica` (*photometric scatter*): $\sigma = \left(\sqrt{\frac{\sum_{i=0}^{n}(\delta_{gal_i} - \bar{\delta})^2}{n-1}}\right)$ em que $\bar{\delta}$ é o valor médio do viés de todo o conjunto das galáxias, n é o número total de galáxias e δ_{gal_i} é o viés de cada uma das galáxias do conjunto.
- σ_{68}: Consiste no percentil dos 68% dos valores de viés δ_{gal} do pico da sua distribuição.
- `Fração fora do padrão` (*outlier fraction*): $f(\alpha\sigma)$, consiste na porcentagem de objetos que tem um viés maior do que um fator α, que é um valor real e definido pelo usuário.

Estas métricas são calculadas para segmentos (*bins*) de redshift. A divisão é feita tanto para redshifts fotométricos quanto para redshifts espectroscópicos. Em cada *bin* foram calculadas as métricas das galáxias que estavam dentro do intervalo de redshift. Estes cálculos servem para quantificar a acurácia do cálculo do redshift fotométrico entre as galáxias. No artigo de Sadeh et al. [15], são mostrados os gráficos destas métricas para o redshift fotométrico. Nas figuras 36 e 37, podemos ver como ficam as métricas do viés fotométrico e da dispersão fotométrica para o conjunto de galáxias de teste do `ANNz`. Nestas figuras, os *bins* têm largura 0,1 dentro do intervalo de 0 até 1,5 dos redshifts.

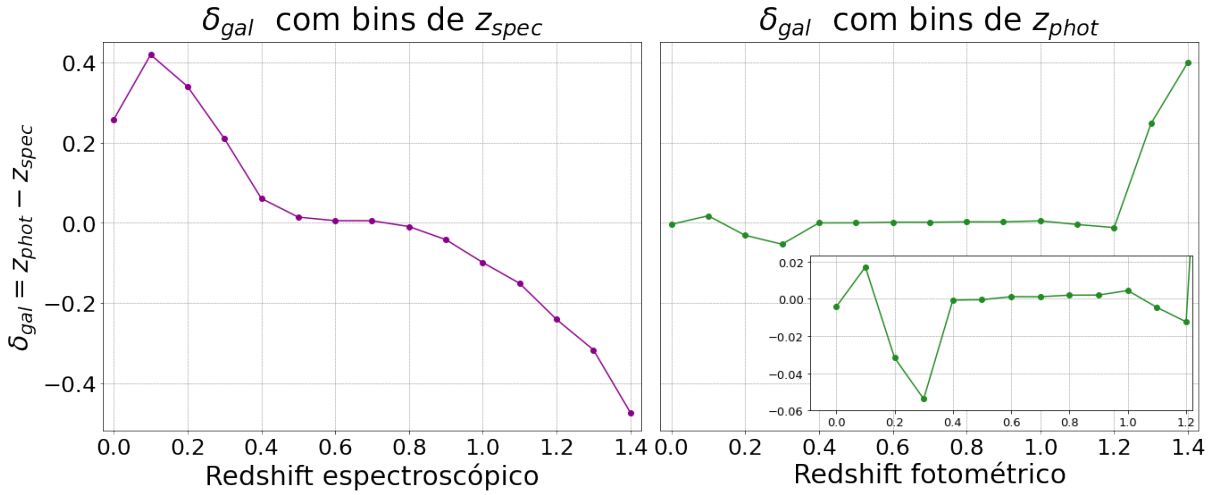


Figura 36: δ_{gal} dos redshifts calculados pelo ANNz. Os redshifts foram divididos em *bins* tanto para o redshift espectroscópico (primeira imagem) quanto para o redshift fotométrico (segunda imagem).

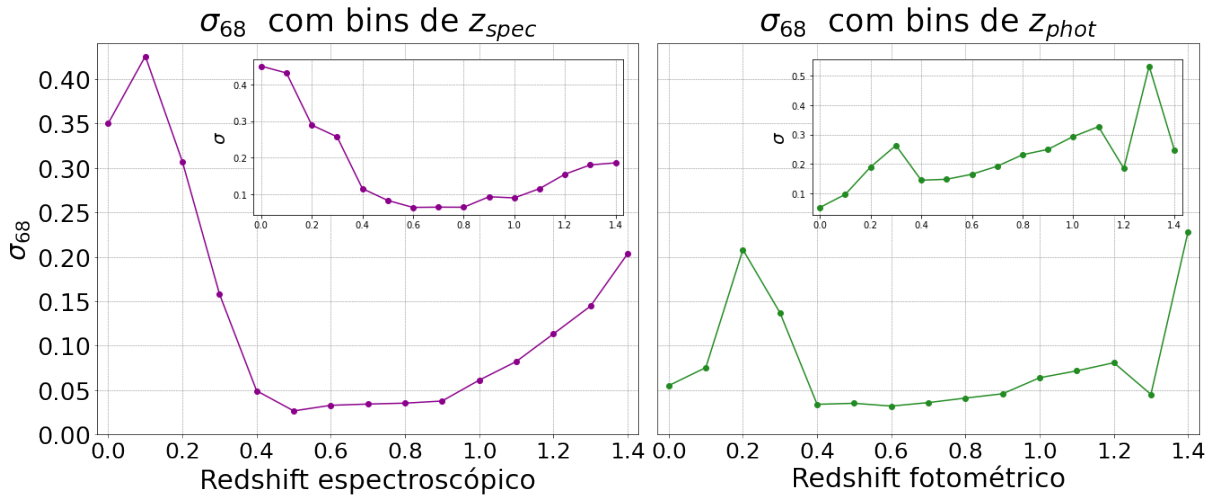


Figura 37: σ dos redshifts calculados pelo ANNz. Os redshifts foram divididos em *bins* tanto para o redshift espectroscópico (primeira imagem) quanto para o redshift fotométrico (segunda imagem).

Para a construção do algoritmo do ANNz, foram considerados alguns tipos de incertezas no cálculo do redshift fotométrico.

`Incerteza dos dados de treinamento.` O código não usa apenas as magnitudes dos 5 filtros no aprendizado de máquina, usa também a incerteza nas medições destes valores. Estas incertezas são disponibilizadas pelo próprio catálogo astronômico (tanto o `VIPERS` quanto o `DES`). Ao considerar estas incertezas, a máquina adiciona um outro resultado ao dado de saída, a incerteza do redshift fotométrico.

`Incerteza nos aprendizado de máquina.` Os aprendizados de máquina têm uma incerteza intrínseca em sua configuração. Esta incerteza se deve tanto à escolha do algoritmo a ser utilizado quanto à aleatoriedade com que as amostras são testadas, e isso resulta em uma variação da performance da máquina.

`Conjunto de dados de treino não representativo.` O conjunto de dados fornecidos à amostra pode não ser um grupo representativo das amostras fotométricas avaliadas. Por exemplo, o nosso conjunto de dados espectroscópicos (`VIPERS`) tem uma limitação de magnitude no filtro `i` até 22,5 e o grupo de amostras fotométricas (`DES`) tem objetos com magnitude de 17 até 24. Apesar de terem uma pequena interpolação nos valores dos dados fotométricos, grande parte dos itens que terão o redshift calculado não farão parte do conjunto representativo com o qual a máquina foi treinada, causando uma incerteza nos dados de saída da máquina, pois o mecanismo não foi treinado para o intervalo de valores fornecidos à máquina. A fim de contornar esse problema, o código tem um fator de correção que coloca pesos nas amostras em função dos dados de entrada. Isso faz com que tanto as amostras de treinamento quanto as fotométricas estejam no mesmo espaço de parâmetros, o que evita que a máquina calcule redshifts com incertezas muito altas. Estes pesos são definidos com base nos intervalos dos valores dos parâmetros do *input*.

`Conjunto de dados de treino incompleto.` O conjunto de dados carece de quantidade de *inputs*, o que implica em uma máquina treinada com uma performance inferior ao que o seu algoritmo permite. Para evitar isso, o código tem um filtro de qualidade que sinaliza ao usuário quando um conjunto com pouca quantidade de objetos está sendo usado. Primeiro, o código calcula $R_{NN}^{a/b}$ que é a distância euclidiana entre o objeto a e o objeto b correspondente mais perto dele na amostra de treinamento. Para dois objetos a e b e os dados de entrada como as 4 cores $m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$, a distância $R_{NN}^{a/b}$ será dada por

$$R_{NN}^{a/b} = \sqrt{\big(c_{g-r}(a) - c_{g-r}(b)\big)^2 + \big(c_{r-i}(a) - c_{r-i}(b)\big)^2 + \big(c_{i-z}(a) - c_{i-z}(b)\big)^2 + \big(c_{z-Y}(a) - c_{z-Y}(b)\big)^2} \tag{62}$$

Na equação (62), o termo $c_{g-r}(a)$ se refere à cor $m_g - m_r$. Então $R_{NN}^{y/n}$ é a distância do objeto y do qual os n_{NN} objetos do conjunto de treinamento são encontrados.

Com isso, calculamos

$$Q_{NN} = max\big\{0, \frac{R_{NN}^{y/n} - R_{NN}^{y/x}}{R_{NN}^{y/n}}\big\} \quad , \tag{63}$$

sendo que o parâmetro Q_{NN} representa a distância entre o objeto calculado e objetos de treinamento similares. Com base na equação (63) podemos ver que para regiões densas do conjunto de treinamento, $R_{NN}^{y/x} \ll R_{NN}^{y/n}$. Isto significa que há pouca distância entre as galáxias,

$$Q_{NN} = max\big\{0, \frac{R_{NN}^{y/n} - R_{NN}^{y/x}}{R_{NN}^{y/n}}\big\} \tag{64}$$

$$\sim \frac{R_{NN}^{y/n} - R_{NN}^{y/x}}{R_{NN}^{y/n}} \xrightarrow{R_{NN}^{y/x} \ll R_{NN}^{y/n}} \frac{R_{NN}^{y/n}}{R_{NN}^{y/n}} = 1 \quad . \tag{65}$$

No código do `ANNz` foi escolhido $n_{NN} = 100$. A propriedade Q_{NN} indica o quão confiável o cálculo do redshift fotométrico pode ser, o valor do corte de Q_{NN} varia a cada caso considerando a fração de objetos excluídos e a relativa melhora na performance da máquina.

Para o cálculo do redshift fotométrico, o código do `ANNz` utiliza um pacote de ferramentas de análise de dados multivariados chamado `TMVA` [100](*Toolkit for Multivariate Data Analysis*). Este pacote contém diversos algoritmos de aprendizados de máquina. O programa treina múltiplas máquinas com diferentes algoritmos com o conjunto de treinamento. O número de máquinas treinadas é definido pelo usuário.

O código do `ANNz` permite também que o usuário tenha acesso a um conjunto de valores que a máquina associa à função de distribuição de probabilidade (*probability distribution*

function) (`PDF`). Após produzir um único valor para o redshift fotométrico, o programa combina este resultado com as incertezas das magnitudes das diferentes aprendizados de máquina treinados. Os principais algoritmos para o cálculo do redshift fotométrico são as redes neurais e as árvores de regressão, detalhadas na seção 4.2. A lista completa dos algoritmos disponíveis no `TMVA`[10] está no manual deste pacote publicado em 2009 [100].

Para cada algoritmo, o pacote `TMVA` modifica as configurações da máquina de forma a diversificar os mecanismos. No caso da rede neural, são treinadas várias máquinas com diferentes estruturas, ou seja, máquinas que diferem entre si no número de camadas intermediárias, funções de ativação e número de nós em cada camada. Além disso, podem diferenciar na escolha e na ordem das amostras usadas no treinamento. Algumas máquinas podem fazer um pré-processamento dos dados antes de usá-los para treinar, como normalizá-los.

O pacote `TMVA` utiliza a linguagem de programação `c++`, porém o `ANNz` tem a interface na linguagem de programação python para o usuário.

Em geral, a variação entre as diferentes configurações é pequena, porém a combinação de diversas variações torna possível construir uma `PDF` (*probability distribution function*) de forma a mostrar as degenerescências do resultado. Ao combinar o resultado de diferentes configurações de aprendizado de máquina, o `ANNz` permite diminuir a arbitrariedade do cálculo do redshift.

O código permite restringir as configurações das máquinas, de forma que a rede não fique muito simples ou muito complexa para o nosso objetivo de calcular o redshift fotométrico. As possibilidades das redes neurais do código `ANNz` foram definidas da seguinte forma: Na seção intermediária o número de camadas pode variar entre 2 e 4 camadas. O número de neurônios em uma seção intermediária pode variar entre N e N+10, sendo que N é o número de parâmetros de um dado de entrada (no caso das magnitudes dos 5 filtros, teríamos N = 5). A função de ativação poderia ser somente `sigmoid` ou `tanh`. E o número de passos entre os testes de convergência foi limitado a um intervalo entre 100 e 500.

Para o cálculo da incerteza σ_{gal} é utilizado o método do K-ésimo vizinho mais próximo. Considerando a definição da distância $R_{NN}^{a/b}$ mostrada na equação (62), procuramos as n_{NN} galáxias b mais próximas (menor $R_{NN}^{a/b}$) da galáxia a de todo o conjunto de amostras. Para cada vizinho l haverá o viés definido como $\gamma_{NN}^{l} = z_{phot}^{l} - z_{spec}^{l}$. Deste valor a largura do percentil de 68% da distribuição de γ_{NN}^{l} é considerada como a incerteza do redshift fotométrico da galáxia a, σ_{gal} [101]. Esta técnica é boa para determinar incertezas fotométricas enquanto for utilizado um conjunto de treinamento que seja um grupo representativo do fotométrico. O valor de n_{NN} é discutido em Oyaizu et al. (2008) [101], onde os autores defendem que n_{NN} tem que ser alto o suficiente para que o cálculo da incerteza não seja misturado com os valores fora do padrão, porém não pode ser tão grande a ponto

[10] https://root.cern.ch/download/doc/tmva/TMVAUsersGuide.pdf

de não fazer diferença usar as n_{NN} galáxias ou todo o conjunto de dados iniciais. Para o ANNz o valor escolhido foi $n_{NN} = 100$.

Para o cálculo da PDF, o ANNz combina diversos métodos de aprendizado de máquina e as suas incertezas. Após o treinamento dos aprendizados de máquina, é gerada uma distribuição de z_{phot} para cada galáxia. Um processo de seleção é aplicado com o intuito de descartar resultados fora do padrão que tenham valores altos do viés γ e da incerteza σ_{68}. Desta seleção, os aprendizados de máquina são classificados pelas suas performances. A máquina que tiver a melhor desempenho será escolhida como a máquina que irá calcular um único valor do redshift fotométrico. Este redshift será chamado de z_{best} pelo código.

Após este processo, são calculadas as incertezas de cada um dos aprendizados de máquina que não foram descartados. São combinados métodos diferentes, de forma a criar candidatas de funções densidade de probabilidade para cada galáxia. A combinação mais trivial é aquela em que todas as máquinas têm o mesmo peso. Isso não necessariamente significa o melhor resultado. Estas combinações são escolhidas de forma aleatória, porém considerando a classificação das máquinas em relação às suas performances.

O desempenho das candidatas é comparado usando um parâmetro C definido como

$$C(z_{spec}) = \int_{z_0=0}^{z_{spec}} p_{reg}(z)\, dz. \tag{66}$$

O resultado que melhor descrever a distribuição do redshift espectroscópico, será selecionado como a PDF final.

A equação (66) define C, a função de densidade cumulativa (*cumulative distribution function*) CDF. Na equação (66) o diferencial da PDF para um determinado redshift é $p_{reg}(z)$ e z_0 corresponde ao limite inferior da PDF. O valor C é o que permite ao código escolher a melhor candidata de PDF, pois o ideal seria o valor que se assemelhasse mais a uma distribuição uniforme.

5.2.1. Código

A figura 38 é um fluxograma dos pontos principais de cada etapa do código do ANNz. Uma descrição mais detalhada do programa está no apêndice F.

Antes do programa começar a funcionar, é preciso fixar o valor dos parâmetros utilizados pelo código, como o número de máquinas a serem utilizadas e o intervalo de valores permitidos de z_{phot}. O código do ANNz treina várias máquinas com diferentes parâmetros e depois escolhe a que teve o melhor desempenho. Nessa etapa, o usuário define o número de máquinas a serem testadas e determina a variação permitida do redshift. Ainda haverá alguns redshift fotométricos calculados fora deste intervalo, mas a incidência deles será significativamente menor do que de z_{phot} dentro da restrição. Os valores para o intervalo do z_{phot} foram fixados baseado no conjunto de treinamento do VIPERS. Cerca de $99,5\%$ das galáxias estão dentro do intervalo $0 < z < 2.7467$. Se não fosse imposta esta restrição,

a máquina iria trabalhar com uma faixa de valores de redshift maior do que a realidade, o que implicaria em adicionar complexidades às máquinas que este trabalho não precisa e só iria custar mais tempo de treinamento para o programa.

A etapa do pré-processamento é a que mostra o caminho onde estão as tabelas a serem utilizadas no aprendizado de máquina. Com estas tabelas, há a definição de quais serão os dados utilizados no programa ao indicar as colunas que o código deve rodar; no nosso caso, as cores nos filtros definidos e as suas incertezas. Por fim, faz o *download* dos dados já detalhados no capítulo 3.

A fase de treinamento é a que coleta todos os dados e treina 100 aprendizados de máquina. As configurações das máquinas são feitas de forma aleatória para obter uma maior variedade do desempenho desse modelos. Além disso, definem quais informações serão usadas como dados de entrada (valor das cores) e quais serão usadas para o cálculo das `PDFs` (a incerteza fotométrica).

A otimização é a etapa em que o código avalia o desempenho de todas os aprendizados de máquina treinados e testa qual retornou o valor do redshift fotométrico mais perto do real. A partir disso, o programa cria diversos conjuntos para construir a `PDF` destes redshifts para cada galáxia. A criação destes grupos é feita de modo a obter a melhor `PDF` considerando as incertezas das magnitudes e a classificação dos aprendizados de máquina. Nesta fase, definimos o número de divisões (*bins*) que queremos da `PDF`. Para o nosso programa, fixamos o número de *bins* como 200.

A última etapa consiste em calcular o redshift fotométrico do catálogo do `DES`. Nesta fase o código baixa estas tabelas, calcula o redshift fotométrico, a incerteza desse valor e as `PDFs` de cada galáxia do catálogo. Com tudo isso calculado, cria-se uma nova tabela de galáxias do `DES` com estes resultados. Um exemplo desta tabela pode ser vista na tabela 9.

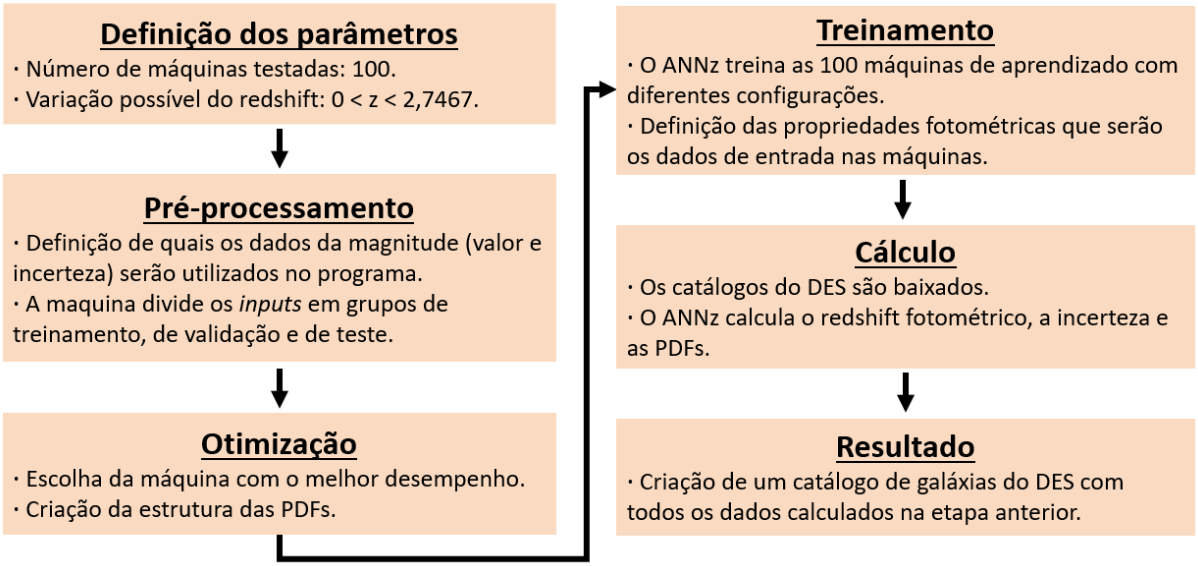


Figura 38: A figura mostra um diagrama dos procedimentos de cada etapa do código do ANNz. Uma visão mais detalhada pode ser vista no apêndice 38.

5.2.2. Resultado

Com a configuração do programa definida, foi feito o treinamento das máquinas com o catálogo das galáxias do `VIPERS` representativas das galáxias do `DES`. O resultado pode ser visto na figura 39.

É possível ver na primeira imagem da figura 39 o comportamento do redshift fotométrico calculado pelo `ANNz` em relação ao redshift espectroscópico. A reta de cor preta mostra os valores de redshift fotométrico para uma máquina ideal, ou seja, se o `ANNz` fosse um aprendizado de máquina perfeito, os pontos estariam sobrepostos a essa reta. Podemos ver que isso acontece entre os redshifts 0,5 até 0,9. Antes e depois deste intervalo de z os pontos e as suas incertezas estão fora da reta ideal.

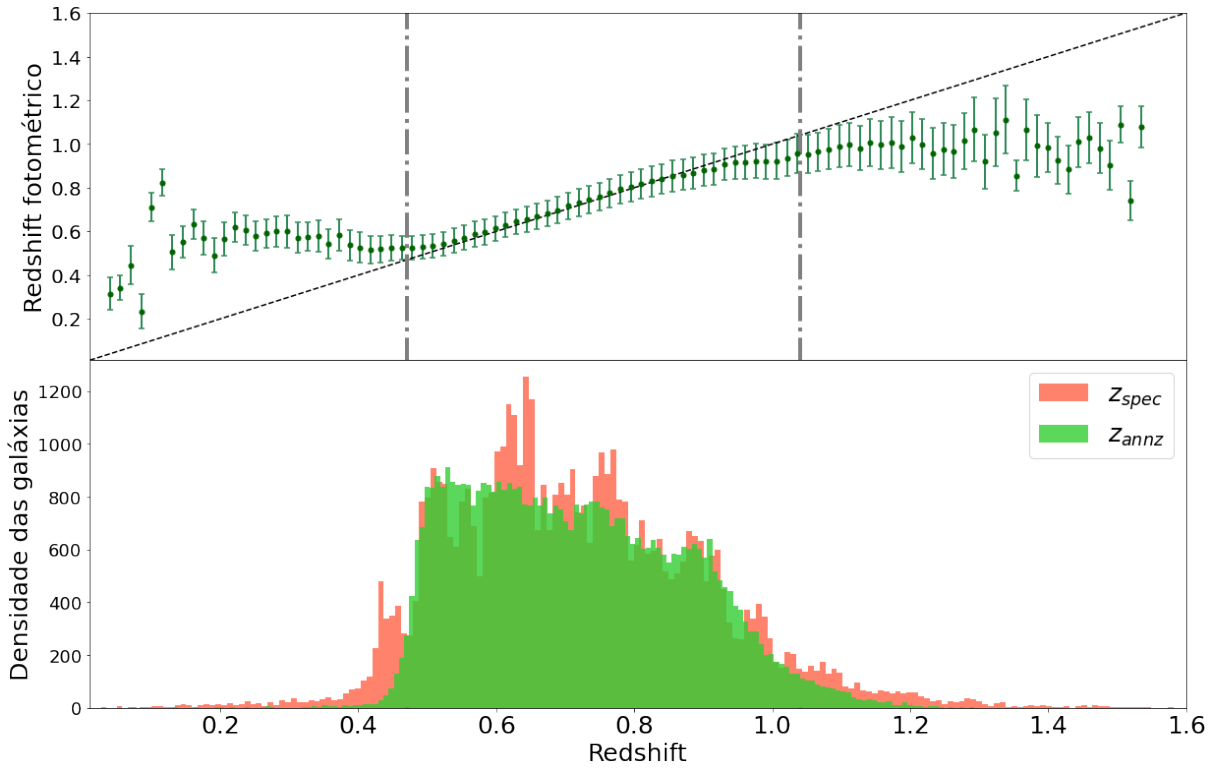


Figura 39: A primeira imagem é um gráfico dos redshifts fotométricos calculados pelo `ANNz` em função do redshift espectroscópico. O z_{spec} foi dividido em *bins* de redshift de largura 0.015. Foi calculada a média dos redshift fotométrico das galáxias em cada um destes segmentos. O resultado é a figura com as 100 divisões e a média de z_{phot} em cada uma delas. Os traços acima e abaixo de cada ponto são as incerteza das médias dos redshifts fotométricos calculados da mesma forma que os redshifts. A segundo imagem é um histograma dos redshifts espectroscópicos (rosa) e fotométricos (verde). A linha cinza tracejada, que percorre ambas as imagens, representa a área do redshift com maior concentração de galáxias, aqui as linhas cercam a área no intervalo $0{,}47 < z < 1{,}04$.

Para explicar este comportamento, podemos observar a segunda imagem da figura 39, que combina o histograma dos redshifts fotométricos e espectroscópicos. É possível ver que a maior quantidade de galáxias está localizada no intervalo de redshift entre 0,5

e 1. Esta informação é confirmada em Scodeggio et al. [13] que dizem que o VIPERS tem como função mapear com detalhes galáxias com redshift entre $0,5 < z < 1,2$. Isto significa que a máquina tem mais amostras de galáxias dentro do intervalo de redshift do que em outros. A consequência disso é que, para z fora deste intervalo, as máquinas não têm amostras o suficiente para treinar, logo o seu desempenho é inferior fora do intervalo $0,5 < z < 1,2$.

5.3. Keras

O terceiro método de cálculo do redshift fotométrico utiliza uma interface de programação de aplicações de aprendizado profundo (*deep learning application programming interface*) escrito na linguagem de programação Python chamada `keras` [102]. Este código usa o aprendizado de máquina do pacote *TensorFlow*. Essa biblioteca foi escolhida por conta da simplicidade e da flexibilidade de seu código. O programa nos deixa definir o número de camadas intermediárias junto com as funções de ativação e o número de neurônios em cada uma delas.

A interface de programação de aplicações (*API*) é um software que possibilita ao usuário acessar um outro sistema operacional e fazer a comunicação entre diferentes sistemas. Esta conexão é definida no código e fornece uma comunicação entre o aplicativo e o usuário, de forma a garantir a portabilidade do código. Uma *API* possibilita a conversão de uma linguagem complexa para uma de nível inferior em outras linguagens de programação [103]. Uma *API* é um tradutor de parâmetros complexos para um mais simples, utilizando um software de alto nível para executar o que o usuário quer. O `keras` facilita ao usuário utilizar os pacotes de *TensorFlow* de forma a focar apenas nas configurações da estrutura da rede neural.

O pacote *TensorFlow* é uma biblioteca de código aberto para computação numérica de alta performance. Foi desenvolvido pelo Google e tornou-se público em 2015, tendo sido atualizada em 2019 [104] [105]. Este pacote pode ser utilizado em diversas linguagens de programação como python, JavaScript, C++ e Java. O pacote permite a construção de algoritmos de aprendizado de máquina e a implementação destes algoritmos em diversos programas. O *TensorFlow* é uma biblioteca com inúmeros parâmetros para a construção de redes neurais. O `keras` permite facilitar o uso deste pacote de maneira que o foco seja apenas na configuração da estrutura do aprendizado de máquina.

Para medir a acurácia do aprendizado de máquina escrito com o `Keras`, foi preciso usar uma métrica para calcular o erro entre o resultado calculado e o real. Esta métrica fica conhecida como função de perda (*loss function*) no programa no código do *TensorFlow* e definida na seção 2.1.3. A métrica fixada para este trabalho foi a que calcula o erro quadrático médio (equação 61). O cálculo desta métrica é feito no momento de treinamento e de validação, ou seja, a rede já foi criada e treinada e agora usa os valores do conjunto de validação para regularizar a sua configuração. O uso da métrica com função do EQM ajuda o usuário a identificar se está ocorrendo sobre-ajuste ao compararmos os valores dos erros para o conjunto de treinamento e de validação.

5.3.1. Testes dos nós

A configuração desta rede neural foi definida através de diversos testes da estrutura e verificação do valor do erro. Isto significa que, a cada mudança na estrutura, foi calcu-

lado o valor do erro, comparado com o anterior e assim decidida a configuração final do aprendizado de máquina.

Por exemplo, no início foi criada uma rede neural com três camadas intermediárias, cada uma com uma função de ativação diferente. A métrica do erro quadrático médio para o conjunto de treinamento deu bons resultados com acurácia de até 98%, porém quando a rede neural foi testada com o conjunto de teste, a acurácia diminuiu drasticamente (56%). Isto é indício de que a rede neural não aprendeu o padrão da fotometria das galáxias para o cálculo do redshift. Uma razão para isto é que a rede neural é muito complexa para o problema apresentado. Após novos testes, foi decidido que a rede neural teria apenas uma camada intermediária. Esta estrutura foi a que teve menor erro no conjunto de teste.

Para a camada intermediária era necessário definir o número de nós desta seção. No nosso exemplo, na seção 2.1.3, nós utilizamos uma rede neural com camada intermediária com 4 nós. Para o programa do `keras`, foi feito um estudo do número de nós (dimensões) necessário para esta camada.

Uma camada com muitos nós pode tornar a rede neural muito complexa, o que acarreta o sobre-ajuste. Uma camada com poucos nós, pode ter dificuldade de aprender o padrão fotométrico pela falta de termos na equação da rede neural, o que acarreta em um sub-ajuste. Foram testados 5 valores de nós na camada intermediária: 10, 15, 20, 25 e 30. Os resultados deste estudo podem ser visto nas figuras 40 e 41.

A figura 40 permite a comparação visual das diferentes configurações do código do `keras`. É possível ver que, em geral, todos os modelos mantêm o mesmo padrão, as galáxias com redshift espectroscópico entre 0,4 e 1,0 conseguem se manter perto da reta ideal. Já galáxias com redshift espectroscópico abaixo de 0,4 e acima de 1,0, têm o seu redshift calculado pelo `Keras` longe do ideal.

É possível ver que na figura 40 das redes neurais com 10, 15 e 20 dimensões é possível ver que mesmo para galáxias com $z_{spec} > 0,4$, ainda há uma quantidade considerável de galáxias cujos valores do redshift fotométricos estão abaixo de 0,4. Isto implica que apesar de a rede neural ter uma concentração de galáxias entre 0,4 e 1,0, há alguns modelos que não conseguiram aprender corretamente o padrão fotométrico para estas galáxias, isto causa o sub-ajuste na rede neural. Pela figura 40, podemos ver que quanto menor o número de dimensões da camada intermediária, maior a quantidade de galáxias com $z_{spec} > 0,4$ cujo z_{phot} ficou abaixo de 0,4.

Além disso, para galáxias com redshift espectroscópico acima de 1,0, é possível ver que as redes neurais consideraram que o seu redshift fotométrico está localizado abaixo de 1,0. Isto é esperado dado que a concentração maior de galáxias do `VIPERS` está situada abaixo de 1,0. Porém, podemos ver que, com o aumento das dimensões, o número de galáxias acima de 1,0 com um erro elevado entre o resultado real e o predito aumenta consideravelmente.

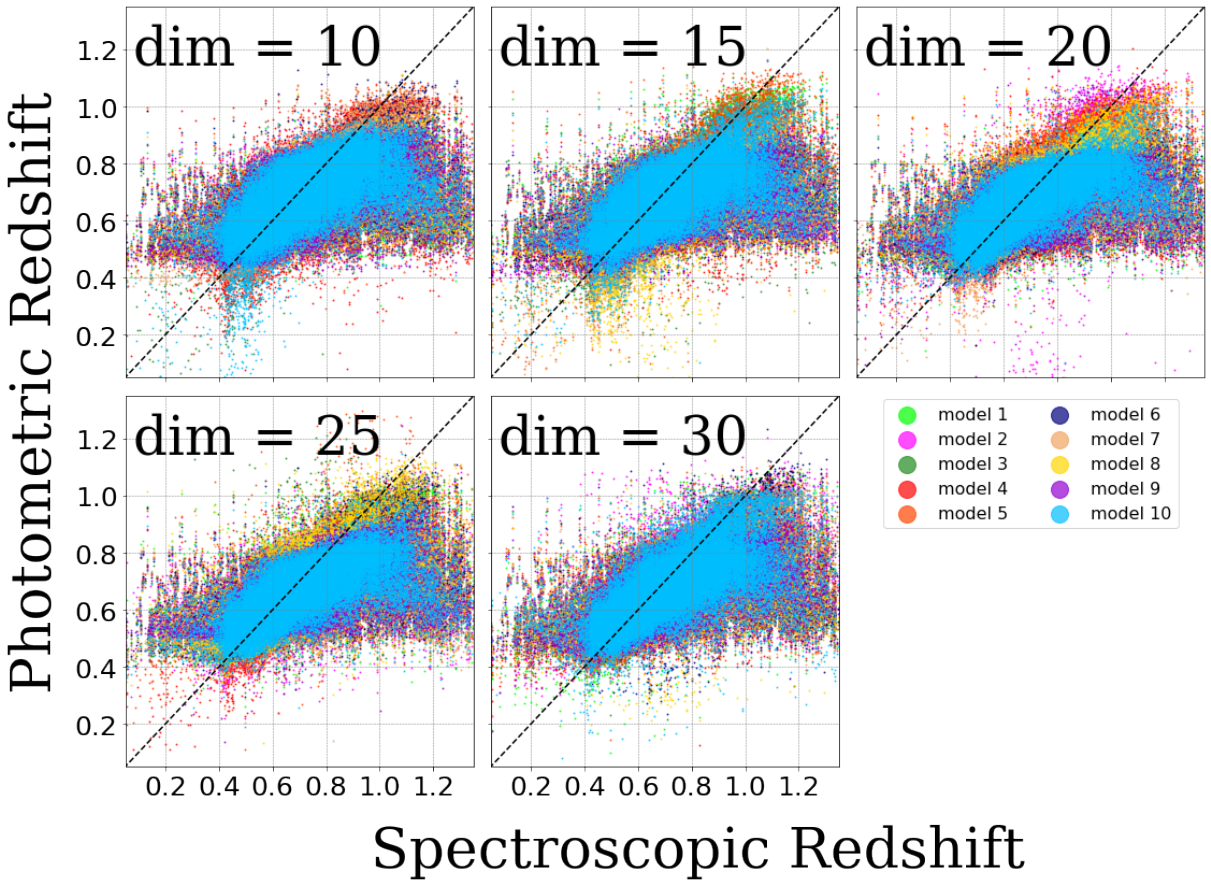


Figura 40: Gráficos de dispersão do estudo dos nós no `keras` para 10 modelos. Para cada configuração (número de dimensões) da camada intermediária foram treinados 10 modelos com diferentes valores iniciais dos parâmetros da rede neural. Cada cor representa um modelo, a legenda na figura mostra qual a cor de cada modelo. Os gráficos mostram o redshift fotométrico do conjunto de teste do `VIPERS` em função do redshift espectroscópico. A linha tracejada mostra a reta ideal dos pontos em que o redshift fotométrico é igual ao redshift espectroscópico.

A figura 41 reafirma que a maioria das galáxias teve um comportamento do redshift fotométrico parecido com o comportamento do redshift espectroscópico. A concentração de galáxias do z_{spec} está localizada no intervalo de $0,4$ a $1,0$, enquanto que a concentração de galáxias do z_{phot} está localizada entre $0,5$ e $0,9$. Como este comportamento é igual ao conjunto de treinamento, podemos confirmar novamente que, quanto mais galáxias disponíveis no treinamento, melhor a rede neural consegue aprender o padrão fotométrico das galáxias.

A figura 41 corrobora as suposições levantadas pelos gráfico de dispersão da figura 40. Podemos ver que para redes neurais com menor dimensão na camada intermediária há picos no histograma em diversos segmentos perto de 0,5 e 0,8. Estes picos indicam que houve um problema no ajuste da rede neural. Os picos sinalizam que a rede neural apenas decorou os pontos e não aprendeu o padrão fotométrico das galáxias, ou seja,

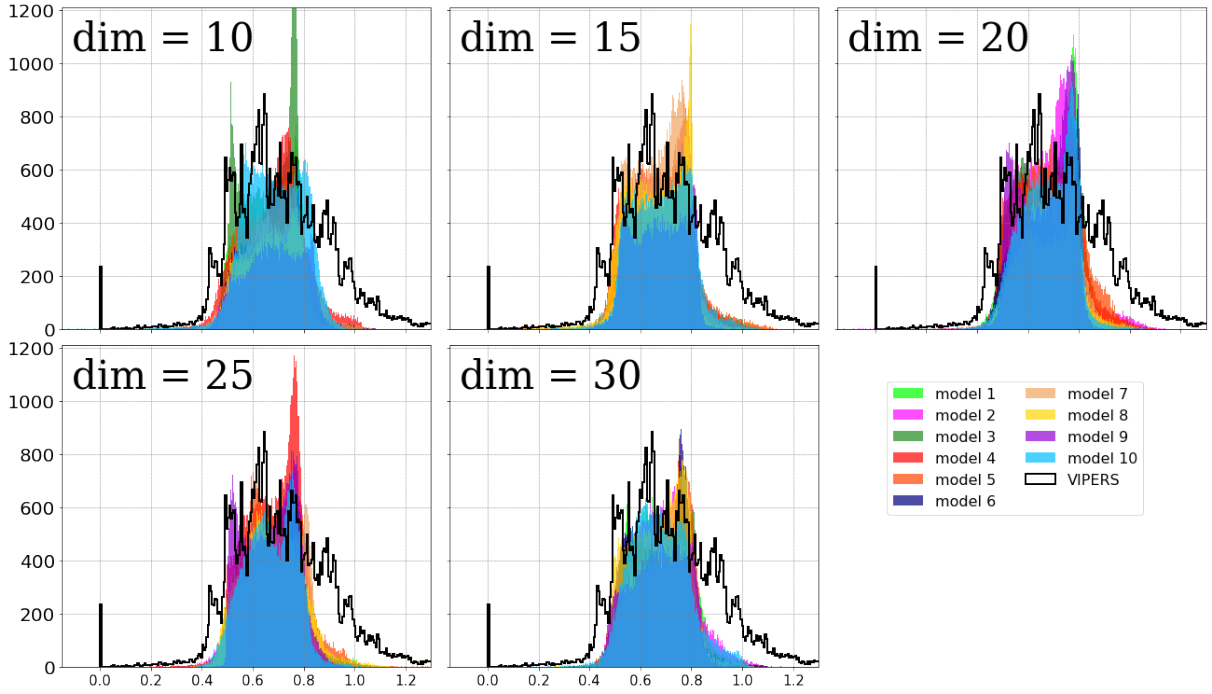


Figura 41: Histogramas do estudo dos nós no `keras` para 10 modelos. Para cada configuração (número de dimensões) da camada intermediária foram treinados 10 modelos com valores iniciais do peso da rede neural diferentes. Os histogramas mostram o número de galáxias separadas por *bins* em função do redshift. As curvas coloridas são os redshifts fotométricos do conjunto de teste do `VIPERS` e a curva preta é o histograma dos redshift espectroscópicos deste mesmo conjunto. Para estes histogramas, as galáxias foram divididas em 100 segmentos entre 0 e 1,3 de acordo com o valor dos seus redshifts. Cada histograma representa um modelo treinado pela máquina, as cores correspondentes a cada modelo estão especificadas na legenda da imagem.

houve um sobre-ajuste da máquina. Podemos ver que o tamanho destes picos diminui com o aumento da dimensão.

As figuras 40 e 41 são bons indicativos do comportamento do redshift fotométrico para as galáxias de teste do `VIPERS`, porém foi preciso usar a métrica (EQM) definida pela rede neural para quantificarmos a diferença entre as 5 configurações da camada intermediária. Podemos ver a representação dos valores desta métrica na figura 42 e na tabela 6.

Pela figura 42 podemos ver que, para as 5 configurações testadas, o comportamento das curvas é similar. A variação entre as configurações é de cerca de 0,03 para os pontos com maior variação (fora do intervalo de maior concentração das galáxias - $0,4 < z < 0,9$). A configuração com maior variação do erro é a de 25 nós na camada intermediária, a segunda que mais varia é a de dimensão 20. Esta variação indica que a rede não conseguiu se ajustar tão bem ao padrão fotométrico das galáxias quanto as outras configurações. Verificando os valores do erro em cada *bin* na tabela 6 e a média de cada dimensão, chegamos à conclusão de que a melhor estrutura para a rede neural do `Keras` é uma

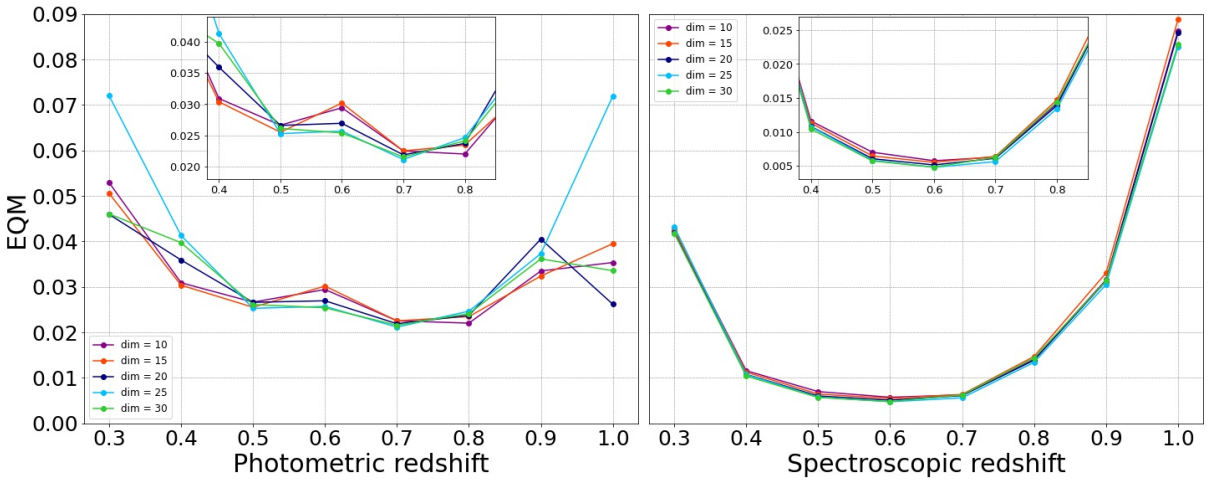


Figura 42: Gráficos do erro quadrático médio z_{phot} calculados pelo `Keras` divididos em segmentos (*bins*) de redshift. Para a construção destes gráficos, foram calculadas as médias dos 10 redshift fotométricos de cada configuração da rede neural. Em cada uma das configurações, as galáxias foram divididas em oito grupos de acordo com o valor dos seus redshifts. Na primeira imagem, as galáxias foram divididas com base nos valores do redshift fotométrico e, na segunda imagem, pelo redshift espectroscópico. Em ambas imagens, o eixo y representa a média do erro quadrático médio (EQM) em cada *bin*. Os gráficos menores são a amplificação dos valores de EQM entre 0,4 e 0,8. A cor da curva de cada dimensão é mostrada da legenda da figura, a curva roxa é para a rede neural com 10 nós na camada intermediária, a curva laranja é para a rede neural com 15 nós na camada intermediária, a curva azul escura é para a rede neural com 20 nós na camada intermediária, a curva azul clara é para a rede neural com 25 nós na camada intermediária e a curva verde é para a rede neural com 30 nós na camada intermediária

camada intermediária com 30 nós. Fixamos este valor em nosso código para predizer o redshift fotométrico com o uso do `Keras`.

Uma quantidade maior de nós na camada intermediária foi testada, porém por adicionar mais complexidade à máquina, a rede neural demorou mais para calcular o redshift fotométrico. Além disso, os valores do erro foram maiores do que quando utilizamos redes com 30 nós na camada intermediária, muito provavelmente por acontecer um sobre-ajuste na máquina. Por isso uma rede neural com 30 nós foi a escolha ideal de máquina para o nosso trabalho.

A figura 43 é uma representação gráfica de como ficou a estrutura do modelo do `Keras` com base na figura 5 do capítulo 2.

5.3.2. Código

Na figura 44, temos o diagrama das etapas do código do aprendizado de máquina que utiliza o `Keras`. O código completo e detalhado pode ser visto no apêndice G.

Na primeira etapa do código são definidas as configurações da rede neural. Neste ponto são fixadas as métricas para avaliar o erro entre o redshift calculado pela rede

bins \ dimensões	10	15	20	25	30
$0,3 < z < 0,4$	0,053	0,051	0,046	0,072	0,045
$0,4 < z < 0,5$	0,031	0,030	0,036	0,041	0,040
$0,5 < z < 0,6$	0,027	0,026	0,027	0,025	0,023
$0,6 < z < 0,7$	0,029	0,030	0,027	0,026	0,023
$0,7 < z < 0,8$	0,023	0,023	0,022	0,021	0,020
$0,8 < z < 0,9$	0,022	0,023	0,024	0,025	0,022
$0,9 < z < 1,0$	0,034	0,032	0,041	0,037	0,031
$1,0 < z < 1,1$	0,035	0,039	0,026	0,072	0,036
média:	**0,032**	**0,032**	**0,031**	**0,040**	**0,030**

Tabela 3: Tabela do erro quadrático médio do redshift fotométrico calculado pelo `Keras` e separado por *bins*.

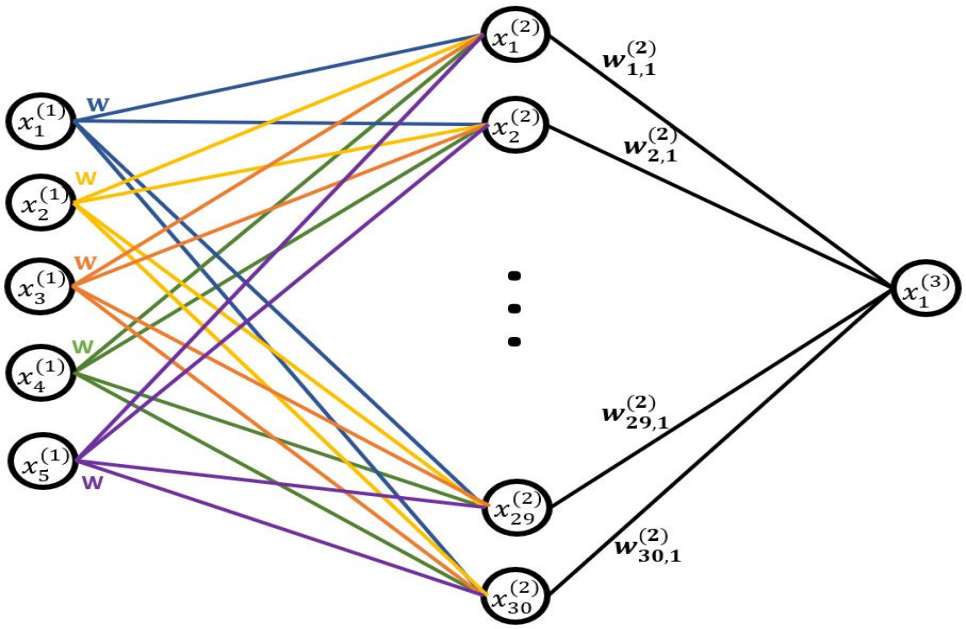


Figura 43: Representação gráfica da estrutura do `Keras` com base na figura 5. No caso do `Keras` temos que os dados de entrada $\{x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}, x_5^{(1)}\}$ são os dados fotométricos das galáxias ($\{m_g - m_r, m_r - m_i, m_i - m_z, m_z - m_Y, m_i\}$). Os valores dos pesos `W` e dos vieses `b` serão definidos na fase de treinamento do aprendizado de máquina. A camada intermediária, como discutido anteriormente, contém 30 nós. E o dado de saída desta rede neural será o redshift fotométrico ($x_1^{(3)} = z_{phot}$).

neural e o redshift real. Também é estabelecida a estrutura da rede neural, como a máquina ter apenas uma camada intermediária. Além disso, é definido o número de vezes que a rede neural será treinada. Assim como o aprendizado de máquina do `GPz`, foi fixado

que seriam treinadas 10 aprendizados de máquina do `Keras` e o resultado final é a média dos resultados destas 10 máquinas.

Na próxima etapa, são trabalhadas as estrutura dos dados de entrada e de saída do código, além da configuração da camada intermediária. Os dados de entrada serão normalizados, esta normalização facilita a máquina a treinar mais rapidamente e convergir em valores dos parâmetros para que a rede neural tenha um bom desempenho. Nesta etapa também é definida a configuração da camada intermediária, como a função de ativação (tangente hiperbólica) e o número de nós (30 dimensões) nesta camada. Por fim, é definida a estrutura do dado de saída da rede; como queremos o redshift fotométrico, a saída da rede neural será um valor real unidimensional.

Na terceira etapa do código, as 10 máquinas são treinadas. A diferença entre as máquinas é que os seus valores iniciais são diferentes (assim como acontece com as redes neurais no `ANNz`) e convergem para valores que tornem o erro (EQM) do aprendizado de máquina menor à medida em que a máquina é treinada. Os 10 modelos da rede neural são então salvos para que possam ser utilizados em catálogos fotométricos.

Por fim, os catálogos fotométricos do `DES` são baixados, assim como os modelos das redes neurais treinadas na etapa anterior. O redshift fotométrico é calculado pelas 10 redes neurais e o resultado final é a média do redshift fotométrico calculado por cada um dos 10 aprendizados de máquina. Com estes resultados, é criado um catálogo com as galáxias do `DES` com os dados de saída das 10 máquinas e a média destes resultados. Um exemplo deste catálogo pode ser visto na tabela 10 ao final desta dissertação.

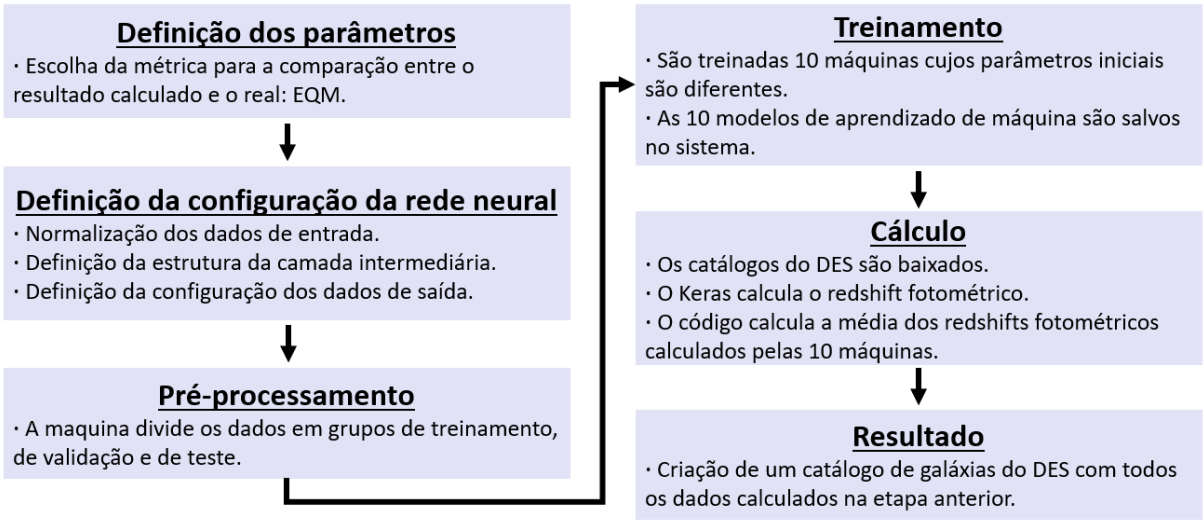


Figura 44: Diagrama do código do `Keras`

6. Comparação dos resultados

Após todo o treinamento das máquinas de aprendizado, o próximo passo foi comparar os resultados. A figura 45 combina os gráficos de dispersão dos redshifts fotométricos em função dos redshifts espectroscópicos das 4 máquinas.

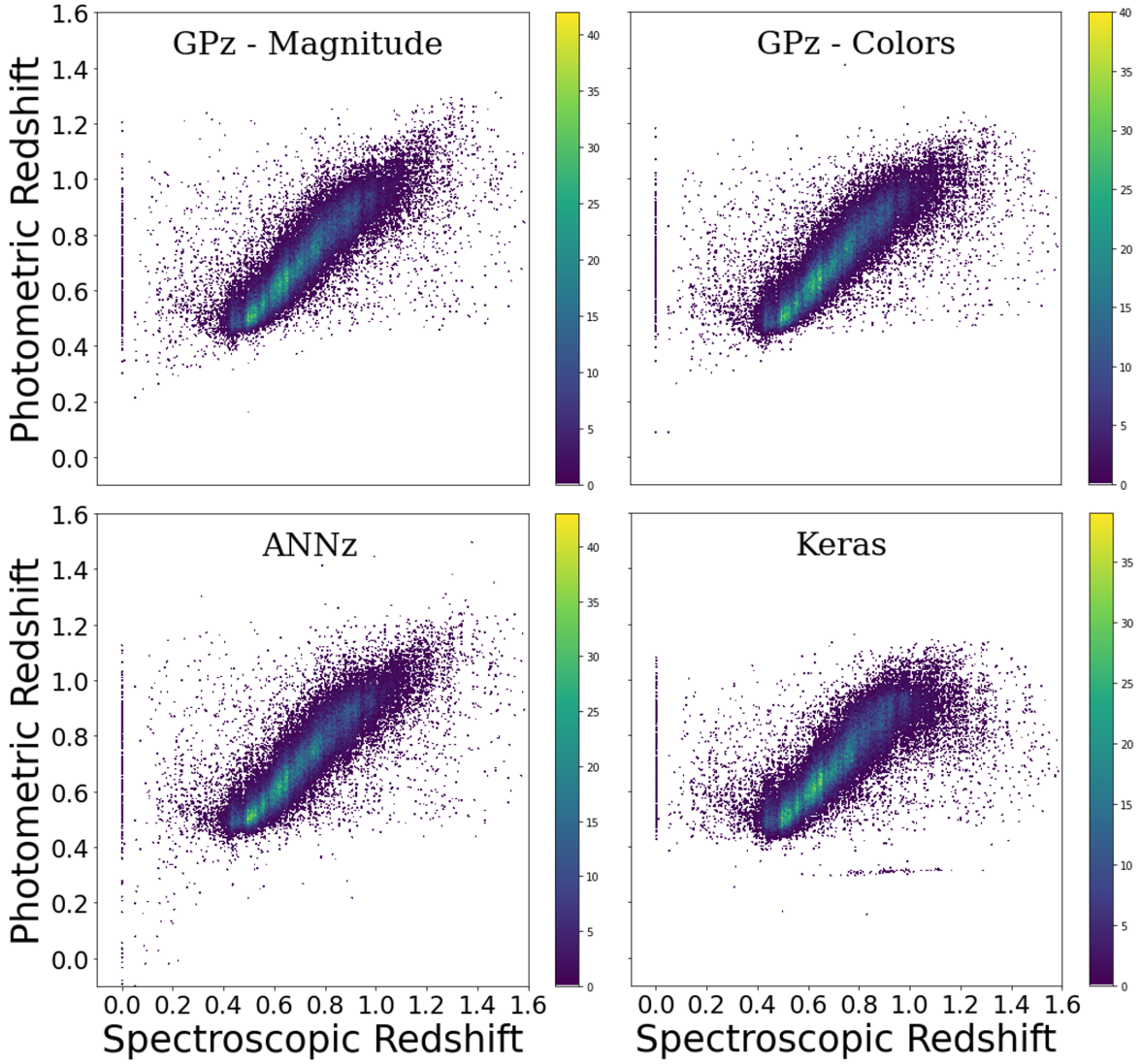


Figura 45: Gráficos de dispersão dos redshifts fotométricos em funções dos redshifts espectroscópico. Os nomes dos programas que calcularam cada z_{phot} estão escritos em cima dos pontos. Todos z apresentam curvas parecidas. É possível ver que a concentração dos redshifts tanto espectroscópicos quanto fotométricos está entre 0,4 e 1,2.

A figura da dispersão dos redshifts fotométricos em função do redshift espectroscópico (figura 45) mostra o quão similares os resultados são. Porém, é possível observar algumas discrepâncias; por exemplo, no redshift calculado pelo `keras`, temos uma fração de galáxias que ficaram com $z_{phot} \sim 0.3$ independente do valor de z_{spec}. Outra coisa é que, para redshifts maiores que 1, os valores calculados do `GPz - Color` e o `keras` se distan-

ciam um pouco da posição ideal, que seria na diagonal do gráfico. Tal efeito é visível nos outros gráficos, mas com menos intensidade. Uma possível causa disso é que os redshifts espectroscópicos se concentram entre 0,4 e 1; logo, as máquinas conseguiram aprender o padrão fotométrico de galáxias com redshift neste limite. Fora deste intervalo, o resultado da máquina pode ter um valor longe do valor real, já que não se sabe o padrão fotométrico dessas galáxias. Podemos dizer então que obtivemos os melhores resultados dentro de um intervalo ótimo, que é $0,4 < z < 1,0$.

Todos os 4 gráficos apresentam uma pequena concentração de galáxias cujo redshift espectroscópico está em 0, porém o redshift calculado está dando um resultado diferente de 0 (a linha vertical em $z_{spec} \sim 0$). O motivo deste fenômeno é o mesmo citado anteriormente, há poucas galáxias com baixo redshift, então a máquina não tem uma quantidade suficiente de galáxias com redshift baixo para criar um padrão e conseguir calcular o redshift fotométrico. Isso adiciona uma incerteza aos resultados, pois o conjunto de treinamento dos aprendizados de máquina teve uma pequena parcela galáxias fora do intervalo ótimo.

Na figura 46, temos o histograma dos redshifts fotométricos (`ANNz`, `GPz - Color`, `GPz - Magnitude` e `keras`) e o redshift espectroscópico (`VIPERS`).

É possível ver, pela figura 46, que os histogramas dos redshifts fotométricos têm características parecidas com as do histograma do redshift espectroscópico. Todos têm uma concentração de galáxias com redshifts no intervalo entre 0,4 e 1. Sendo que z_{spec} possui alguns vales (*bins* com menor número de galáxias) dentro deste intervalo, enquanto os redshifts fotométricos não têm esses vales, ou seja, a quantidade de galáxias em cada *bin* é relativamente constante. Outra discrepância é que o redshift espectroscópico tem um pico de galáxias no primeiro *bin*, o que nenhum dos outros histogramas apresenta.

A figura 47 é baseada na imagem 46, sendo um histograma da diferença entre o redshift fotométrico e o espectroscópico de cada método. Para cada galáxia foi calculada a diferença $z_{phot} - z_{spec}$.

A figura 47 mostra que o pico da diferença está na região de $z_{phot} \sim 0$, o que implica que os redshifts são parecidos e têm um erro pequeno. O formato dos histogramas de todas as 4 máquinas estão parecidos. Uma pequena diferença acontece no `ANNz` em que o pico tem 200 galáxias a menos que as outras. Isto significa que a diferença dos redshifts do `ANNz` está mais dispersa que e das outras, porém é uma porcentagem pequena, se considerado o total de galáxias avaliadas.

Para medir quanto tempo cada aprendizado de máquina demora para calcular o redshift fotométrico, foram escolhidos, de forma aleatória e em coordenadas diferentes, 10 pixels do catálogo do `DES`. O tempo necessário para calcular cada pixel é mostrado na tabela 4. Para as máquinas de aprendizado do `GPz - Color` (média $= 3,695s$) e `GPz - Magnitude` (média $= 2,393s$), as médias dos tempos para o cálculo do redshift ficaram mais ou menos parecidas, sendo que o `GPz - Magnitude` foi o que calculou mais rapidamente. O `Keras` (média $= 6,842s$) demorou em média o dobro dos `GPz`s. Já o aprendizado

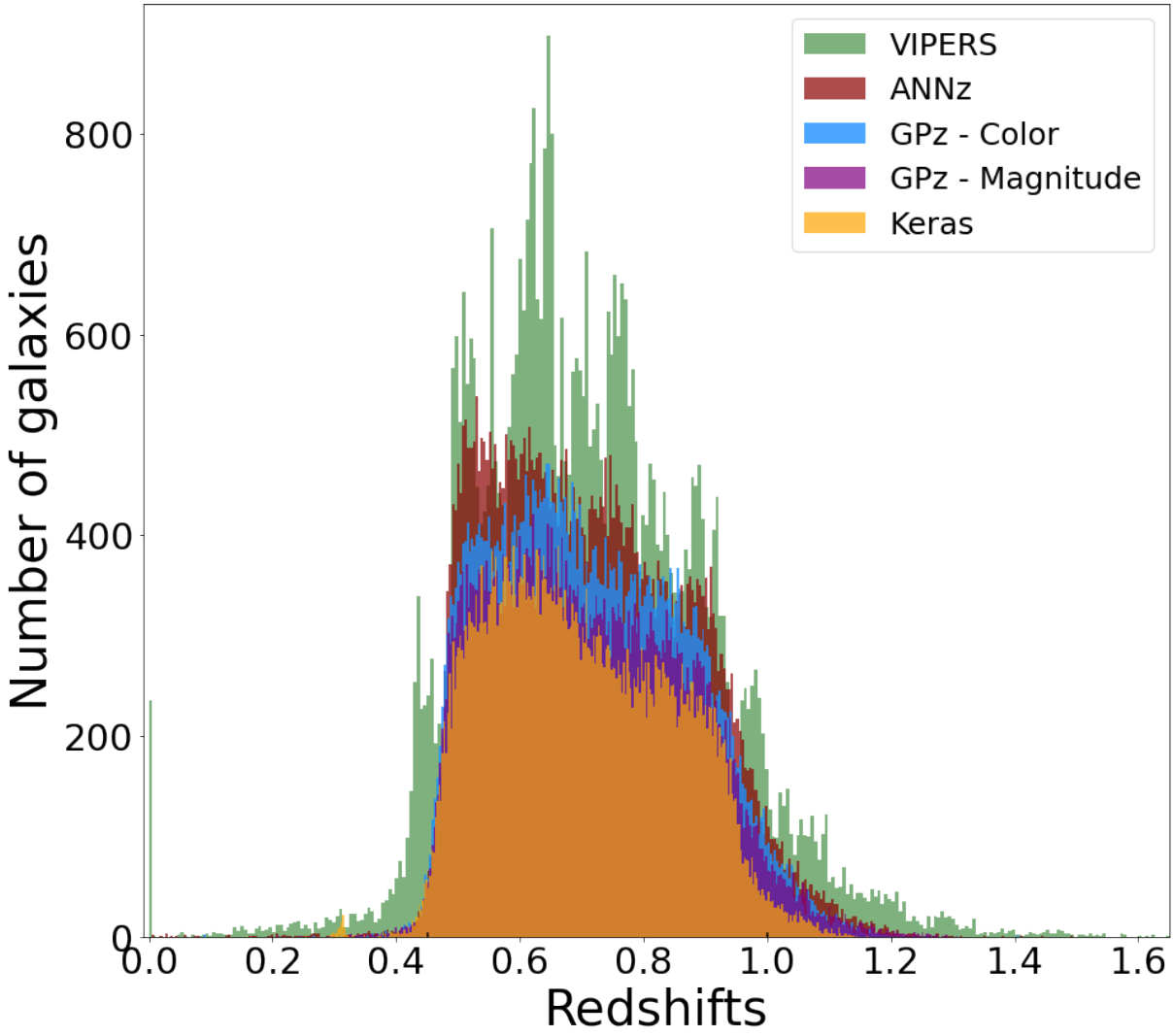


Figura 46: Histograma dos redshifts espectroscópicos e fotométricos de 50115 galáxias do VIPERS. O histograma verde representa os redshifts espectroscópicos observados pelo VIPERS. O histograma azul representa os redshifts fotométricos calculado pelo `GPz - Color`, o histograma roxo são os z_{phot} calculados por `GPz - Magnitude`, a curva laranja são os redshifts fotométricos calculados pelo `keras` e o histograma vinho são os redshifts fotométricos calculados pelo `ANNz`. Algumas máquinas tiveram como resultado galáxias com valor de z_{phot} fora do padrão, ou seja, valores abaixo de 0 ou muito maiores que 1.6 (limite do z_{spec}). Estas galáxias foram desconsideradas para uma melhor visualização do comportamento dos redshifts fotométricos de cada máquina. A porcentagem de galáxias retiradas para a formação deste histograma foi: `GPz - Color:` 0.39%, `GPz - Magnitude:` 0.16%, `Keras:` 0.00% e `ANNz:` 0.02%

de máquina do `ANNz` (média $= 452,786s$) demorou em média quase 100 vezes mais que as outras máquinas de aprendizado. A desigualdade dos tempos se dá pela complexidade das máquinas de aprendizado. A configuração do `ANNz` utiliza diversas máquinas de aprendizado para calcular o redshift, uma destas máquinas calcula o redshift fotométrico que temos como resultado e uma combinação das outras calcula a `PDF` para cada galáxia. A organização das responsabilidades de cada máquina do `ANNz` é definida na fase de treina-

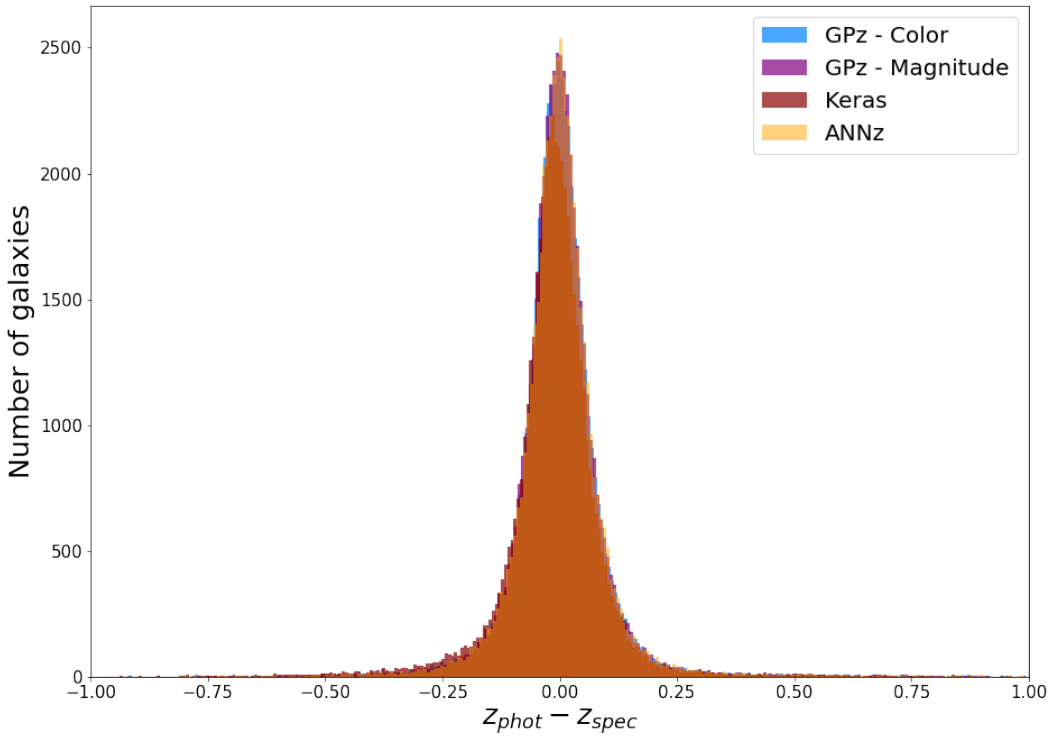


Figura 47: Gráfico das diferenças dos redshifts. Para cada galáxia foi feita a conta $z_{phot} - z_{spec}$, o resultado destas diferenças é o histograma da figura. Alguns valores ficaram fora do intervalo [-1,1] da diferença, são as galáxias cujos redshifts fotométricos fogem do padrão, ou seja, $z_{phot} < 0$ ou $z_{phot} > 2$. A porcentagem de galáxias que estão fora deste padrão é fornecida na legenda da figura 46.

mento. Já as estruturas das outras máquinas são mais simples quando comparadas com o `ANNz`; por exemplo, o `GPz` não utiliza redes neurais, já que o cálculo do seu redshift é feito através de processos gaussianos. A rede neural do `Keras` possui apenas uma camada intermediária, tornando o processo de calcular o z_{phot} mais simples e rápido.

Nº de galáxias	**GPz - C**	**GPz - M**	**Keras**	**ANNz**
6	0,430 (s)	0,437 (s)	0,946 (s)	98,524 (s)
10	0,332 (s)	0,318 (s)	0,919 (s)	115,343 (s)
31	0,344 (s)	0,330 (s)	2,121 (s)	112,147 (s)
13 833	2,413 (s)	1,962 (s)	4,759 (s)	345,851 (s)
14 696	2,626 (s)	1,954 (s)	4,791 (s)	328,713 (s)
33 961	5,583 (s)	4,335 (s)	9,725 (s)	648,284 (s)
34 696	5,590 (s)	4,167 (s)	10,385 (s)	657,858 (s)
39 682	6,389 (s)	4,726 (s)	11,474 (s)	732,799 (s)
37 403	6,414 (s)	4,958 (s)	10,822 (s)	715.779 (s)
42 895	6,828 (s)	5,076 (s)	12,477 (s)	772.560 (s)
média: 21 721	**3,695 (s)**	**2,393 (s)**	**6,842 (s)**	**452,786 (s)**

Tabela 4: Tabela do tempo necessário para as máquinas calcularem o redshift fotométrico de cada pixel. Cada pixel é uma tabela com uma certa quantidade de galáxias, esta quantidade está especificada na primeira coluna da tabela. A última linha é a média do número de galáxias em cada pixel e a média do tempo necessário para cada máquina calcular o redshift fotométrico.

6.1. Métricas

As figuras 45 e 46 mostram que os redshifts fotométricos das 4 máquinas estão com valores parecidos, porém, é preciso estabelecer métricas para uma comparação quantitativa. Para esta comparação, foram escolhidas as 4 métricas definidas em Rivera et al. [106]. A descrição de cada métrica está detalhada nos itens abaixo.

- **Viés**

A primeira métrica é o viés (*bias*), que mede o desvio do redshift fotométrico do redshift espectroscópico. O seu valor é calculado pela diferença entre os dois redshifts e normalizado pelo redshift espectroscópico,

$$Bias = \left(\frac{z_{phot} - z_{spec}}{1 + z_{spec}} \right) \quad . \tag{67}$$

Para analisarmos os valores da métrica, foram feitos dois tipos de gráficos na figura 48. A imagem 48a dividiu as galáxias com base nos valores dos seus redshifts fotométricos. A divisão resultou em 9 *bins* a partir de $zphot = 0,3$. Cada conjunto possui uma largura de 0,1, menos o último conjunto que concentra os z_{phot} de 1,1 até 5. Para valores menores que 0,3, não havia galáxias com redshifts abaixo de $0,3$ o suficiente para que fossem calculadas as métricas. Para a imagem 48b, o gráfico desta divisão foi feito com base nos valores dos redshifts espectroscópicos das galáxias. Em cada *bin* foi calculado o viés de todas as galáxias do grupo e calculada a média destes valores. A média está indicada nas imagens junto ao valor do *bin*. Em ambas as imagens, foi feita a mesma conta da métrica, apenas a propriedade que guiou a divisão dos grupos foi diferente.

A figura 48 mostra como a métrica do viés se comporta com o aumento dos valores dos redshifts. Para a divisão feita pelos redshifts fotométricos (imagem 48a), podemos ver que para redshifts menores que 0,4, a média do viés alcançou números maiores que o esperado. Em particular o`Keras` que obteve valor abaixo de $-0,25$, as outras máquinas também obtiveram valores acima do esperado, menos os resultados do `ANNz` que mantiveram um viés de menos de $0,05$. Para galáxias com redshifts fotométricos entre $0,4$ e $1,1$, os vieses entres os redshifts são menores do que 1% dos valores de redshift fotométrico.

Para valores maior do que $1,1$, havia pouca quantidade de galáxias e por isso os valores ficaram muito fora do padrão. São mostrados apenas os valores de redshift entre $0,3$ e $1,1$, pois, para valores fora deste limite, há uma menor porcentagem de galáxias. Isto faz com que qualquer valor do redshift longe do padrão modifique significativamente a média do resultado da métrica. Então, para melhor comparação, foram colocados nos gráficos apenas os valores dos redshifts fotométricos dentro do intervalo $0,3 < z_{phot} < 1,1$. A porcentagem de galáxias que não estão sendo representadas na imagem por conta disso é: `GPz - Color:` 0.91%, `GPz - Magnitude:` 1.14%, `Keras:` 0.30% e `ANNz:` 1.13%. Estes núme-

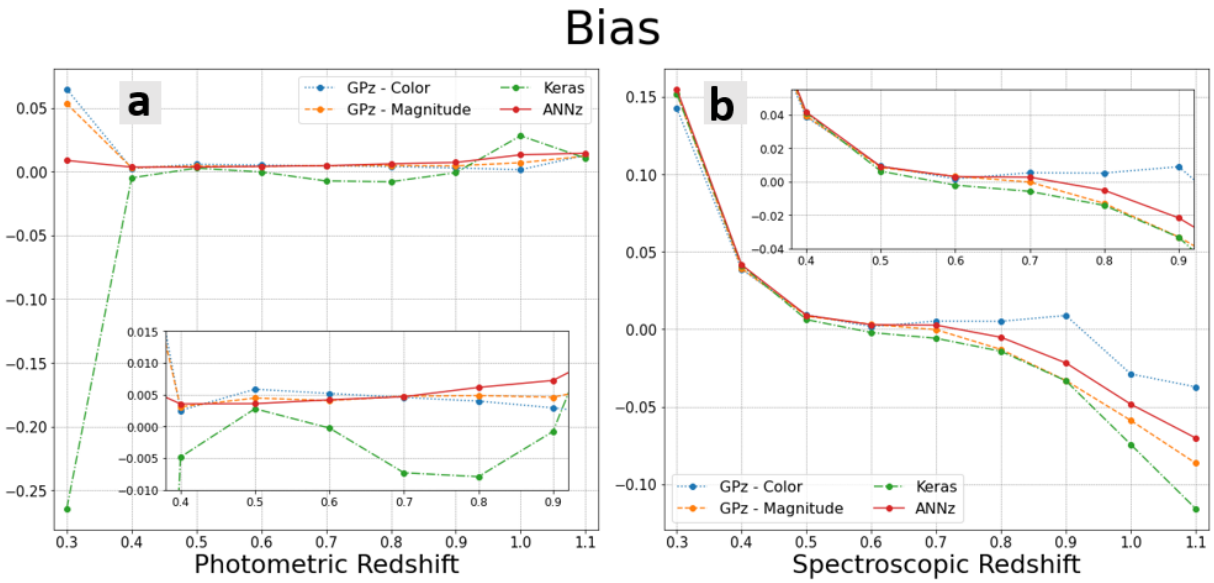


Figura 48: Gráfico do viés do redshifts fotométricos para aprendizado de máquina. A linha azul é o viés dos z_{phot} calculados pelo `GPz - Color`, a linha laranja é a métrica dos redshifts fotométricos dos z_{phot} calculados pelo `GPz - Magnitude`, a linha verde é o resultado do viés para os z_{phot} do `Keras` e a linha vermelha para os z_{phot} do `ANNz`. Na primeira imagem as galáxias foram divididas com base nos seus valores do redshift fotométrico em segmentos de largura 0,1 dentro do limite de $0,3 < z_{phot} < 1,1$. Na imagem 48b as galáxias foram divididas com base nos valores dos redshifts espectroscópicos em segmentos de largura 0,1 do limite de $0,3 < z_{spec} < 1,1$.

ros são baixos e por isso podemos dizer que estes gráficos representam o comportamento geral dos redshifts fotométricos das galáxias.

Como o conjunto de redshifts espectroscópicos utilizado para o treinamento da máquina tem a sua concentração de galáxias entre $0,4$ e $1,1$, a comparação das métricas para valores maiores que $1,1$ não acrescenta em nada no nosso estudo. Como há poucas galáxias fora deste intervalo, as máquinas não conseguem aprender as propriedades fotométricas das galáxias para calcular um redshift fotométrico perto do valor real.

Na imagem 48b, foi feito o mesmo processo de cálculo do viés, porém as galáxias foram divididas de acordo com o valor do redshift espectroscópico. É possível ver que para os resultados de todas as máquinas houve uma diminuição dos valores do viés com o aumento do redshift. A curva é a mesma para todos os resultados, porém esta curva é mais acentuada para o `Keras` e menos acentuada para o `GPz - Color`. Apesar das desigualdades nas curvas, os valores destas diferenças não passam de $0,10$.

A figura 48 mostra que os vieses das quatro máquinas dentro do intervalo de concentração de galáxias do `VIPERS` $(0,4 < z < 1,0)$ são relativamente parecidos, apresentando pequenas diferenças que não afetam substancialmente os valores de z_{phot}.

- **Sigma** ($\sigma_{z_{phot}}$)

A segunda métrica é o desvio padrão do redshift fotométrico. Esta métrica é definida pelo desvio padrão σ das galáxias,

$$\sigma_{z_{phot}} = \left\langle \left(\frac{z_{phot} - z_{spec}}{1 + z_{spec}} \right)^2 \right\rangle^{1/2} \quad . \tag{68}$$

Assim como na figura do 48, foram feitos dois gráficos para verificarmos a evolução das métricas em função dos redshifts. Na figura 49, as galáxias foram divididas de acordo com os seus valores de redshift fotométrico e espectroscópico em segmentos de largura 0,1 para podermos avaliar a evolução desta métrica com o aumento de z.

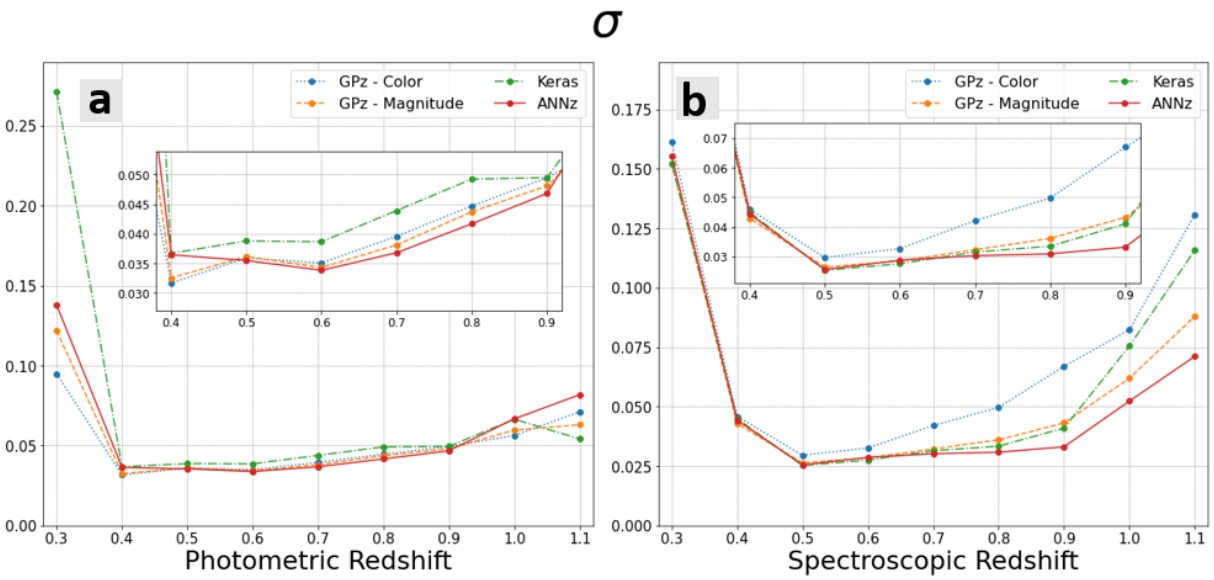


Figura 49: Gráfico do sigma (σ) para os redshifts fotométricos. A linha azul é o $\sigma_{z_{phot}}$ dos z_{phot} calculados pelo `GPz - Color`, a curva laranja é a métrica dos redshifts fotométricos dos z_{phot} calculados pelo `GPz - Magnitude`, a linha verde é o resultado do $\sigma_{z_{phot}}$ para os z_{phot} do `Keras` e a linha vermelha, para os z_{phot} do `ANNz`. Na primeira imagem, temos o $\sigma_{z_{phot}}$ em função do redshift fotométrico, ou seja, as galáxias foram divididas com base nos seus valores do redshift fotométrico em *bins* de largura 0,1 do limite de $0,3 < z_{phot} < 1,1$. Na segunda imagem, as galáxias foram divididas com base nos valores dos redshifts espectroscópicos em segmentos de largura 0,1 do limite de $0,3 < z_{spec} < 1,1$.

Para a divisão baseada no redshift fotométrico, o sigma está entre 0 e 0,30. Na imagem 49a, podemos ver que, tal qual aconteceu nas imagens da figura 48, os menores valores de σ ocorrem para galáxias cujos redshift estão dentro de um intervalo, neste caso, entre 0,4 e 1,0. A explicação é a mesma do viés, este intervalo é onde tem a maior porcentagem de galáxias do `VIPERS`, o que permite as máquinas serem treinadas com bastante dados e, com isso, criar um padrão para calcular o redshift fotométrico.

A imagem 49b traz as galáxias separadas pelo valor do seu redshift espectroscópico. Nesta figura podemos ver uma diferença maior entre as máquinas do que no σ em função

do redshift fotométrico. Para valores de redshifts mais altos ($z > 0,7$), os resultados do `GPz - Color` começam a se separar do valores das outras máquinas, e em $z > 0,9$, o mesmo acontece com os resultados do `Keras` e do `GPz - Magnitude`.

Em ambas as imagens, o σ dos resultados do `ANNz` são os que permaneceram constantes com o aumento dos redshifts.

- **Sigma 68 ($\sigma_{z_{phot},68}$)**

Para o desvio $\sigma_{z_{phot}}$ há uma fórmula para as galáxias com menor erro. Esta fórmula concentra o percentil 68 de galáxias com a menor diferença normalizada. Isto significa que a métrica calcula o valor de $z^i_{phot} - z^i_{spec}/1 + z^i_{spec}$ para cada galáxia i do catálogo e seleciona 68% das galáxias com os menor valores desta equação. A partir disso, o máximo valor calculado do módulo desta equação é escolhido como $\sigma_{z_{phot},68}$. A fórmula pode ser vista na equação

$$\sigma_{68} = max_{i \in U} \left\{ \left| \frac{z^i_{phot} - z^i_{spec}}{1 + z^i_{spec}} \right| \right\} \tag{69}$$

É importante que $\sigma_{z_{phot},68}$ tenha valores pequenos para cada galáxia, já que precisamos deste indicativo para utilizarmos o redshift fotométrico na criação do mapa de distribuição de redshift do catálogo de galáxias do `DES`. Se esta métrica fosse muito grande, implicaria que a informação em cada pixel da divisão do mapa dos redshifts estaria misturada com a informação da porção adjacente, o que causaria a perda da precisão em nossos cálculos.

O mesmo procedimento de divisão das galáxias em *bins* para os gráficos das métrica anteriores, foi feito com o $\sigma_{z_{phot},68}$.

Como a figura 50 concentra 68% das galáxias com melhor valor da métrica σ, temos valores menores para todos os redshifts calculados. Na primeira imagem, as galáxias tinham $\sigma_{z_{phot},68}$ entre $0,1$ e $0,4$, agora os valores estão entre $0,02$ e $20,07$, ou seja, menos de $1/3$ dos valores de σ. O mesmo efeito acontece para galáxias divididas pelo redshift espectroscópico, os valores de σ chegavam até $1,6$ por conta do `GPz - Color`, agora o máximo que esta métrica consegue chegar é em $0,12$ e os resultados estão com os valores da métrica mais parecidos. Isto pode ser visto através das linhas que estão quase se sobrepondo, diferente do gráfico de σ.

- **Taxa de galáxias dentro do padrão (FR_e)**

Por fim, temos a taxa de galáxias dentro do padrão FR_e (equação 70) que calcula a porcentagem de galáxias que têm o redshift fotométrico dentro do erro esperado. Para um número n de galáxias esta métrica calcula a quantidade em porcentagem de objetos

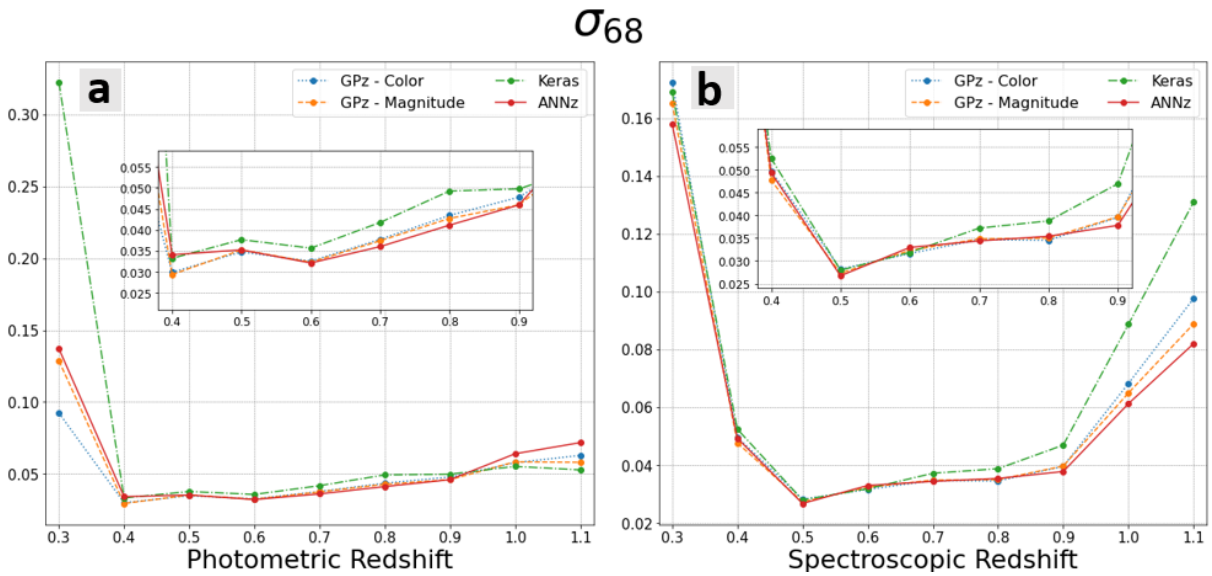


Figura 50: Gráfico do $\sigma_{z_{phot},68}$ para os redshifts fotométricos. A linha azul é o $\sigma z_{phot}, 68$ dos z_{phot} calculados pelo `GPz - Color`, a linha laranja é a métrica dos redshifts fotométricos dos z_{phot} calculados pelo `GPz - Magnitude` a linha verde é o resultado do viés para os z_{phot} do `Keras` e a linha vermelha para os z_{phot} do `ANNz`. Na primeira imagem 50 as galáxias foram divididas com base nos seus valores do redshift fotométrico em segmentos de largura 0,1 do limite de $0,3 < z_{phot} < 1,1$. Na segunda imagem 50 as galáxias foram divididas com base nos valores dos redshifts espectroscópicos em segmentos de largura 0,1 do limite de $0,3 < z_{spec} < 1,1$.

que têm o valor $z^i_{phot} - z^i_{spec}/1 + z^i_{spec}$ acima da tolerância e aceita do erro e definida pelo usuário. Em nosso trabalho escolhemos utilizar o o valor da tolerância igual ao do Rivera et al. [106], então fixamos $e = 0,15$ e temos que

$$FR_e = \frac{100}{n}\left\{\left|\frac{z^i_{phot} - z^i_{spec}}{1 + z^i_{spec}}\right| < e\right\}. \tag{70}$$

A figura 51 traz as galáxias separadas de acordo com os seus redshifts. A máquina ideal conseguiria obter a taxa a 100%. Podemos ver que, em ambas as figuras, a taxa fica acima de 95% no intervalo de redshifts $(0,4 < z < 0,9)$ que concentra mais galáxias. Fora deste intervalo, a taxa tem valores menores, o que significa que há muitas galáxias com valor de redshift fotométrico longe do padrão esperado. Apesar de algumas discrepâncias, todas as máquinas têm um resultado parecido para esta métrica.

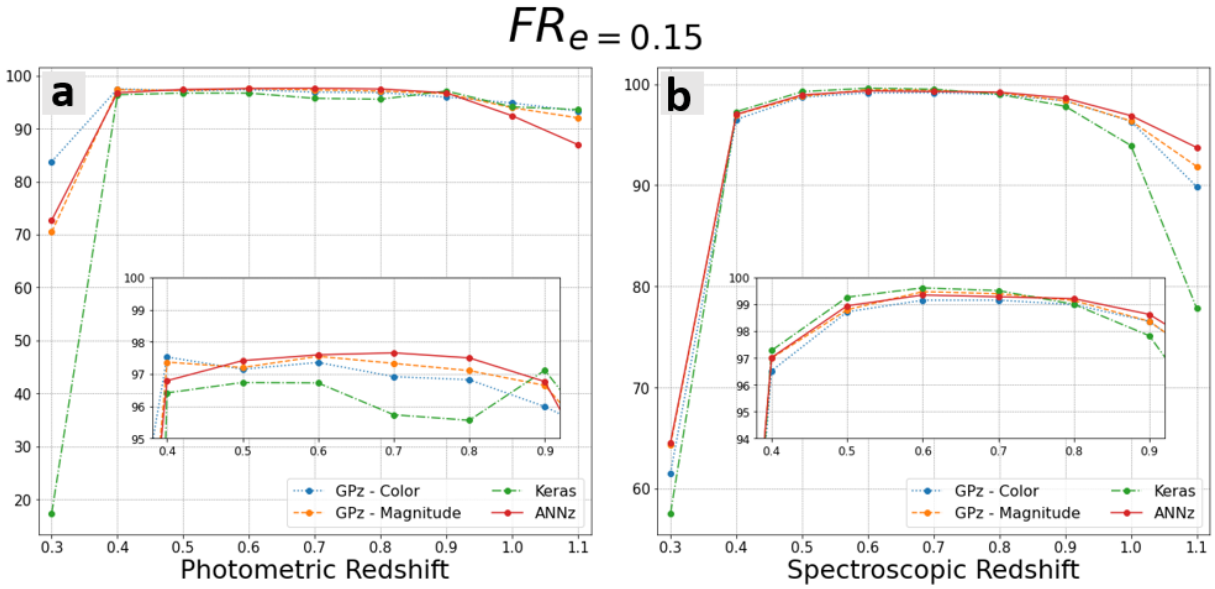


Figura 51: Gráfico do FR_e para os redshifts fotométricos. A linha azul é a taxa de galáxias típicas dos z_{phot} calculados pelo `GPz - Color`, a linha laranja é a métrica dos redshifts fotométricos dos z_{phot} calculados pelo `GPz - Magnitude`, a linha verde é o resultado do viés para os z_{phot} do `Keras` e a linha vermelha para os z_{phot} do `ANNz`. Na imagem 51a as galáxias foram divididas com base nos seus valores do redshift fotométrico em segmentos de largura 0,1 do limite de $0,3 < z_{phot} < 1,1$. Na imagem 51b as galáxias foram divididas com base nos valores dos redshifts espectroscópicos em segmentos de largura 0,1 do limite de $0,3 < z_{spec} < 1,1$.

6.2. Estrutura de dados com redshift fotométrico

Com todos os redshifts fotométricos calculados e divididos nos 4096 conjuntos da K-d Tree, foi possível analisar os resultados. Baseado nas figuras 9 a 12 do artigo *The DEEP2 galaxy redshift survey: design, observations, data reduction, and redshifts* de Newman et al. [107], fizemos uma análise dos dados a partir dos diagramas cor-cor com as cores $m_g - m_r$ em função de $m_r - m_i$ para cada galáxia.

Newman et al.	**Santos et al.**
$m_g - m_r < 0.389$; and/or $m_r - m_i > 1.211$; and/or $m_g - m_r < 2.45$ x $m_r - m_i$ - 0.311	$m_g - m_r < 0.691$; and/or $m_r - m_i > 0.95$; and/or: $m_g - m_r < 1.90$ x $m_r - m_i$ - 0.298
$z_{corte} = 0,75$	$z_{corte} = 0,59$

Tabela 5: Tabela dos limites das linhas da divisão no diagrama cor-cor. Na primeira coluna estão os limites definidos no artigo no Newman et al. [107] enquanto na segunda coluna estão os limites que definimos para o nosso trabalho com base no catálogo de galáxias do VIPERS.

No artigo está dito que é possível selecionar as galáxias com redshift fotométrico alto com base em sua fotometria. Para isso, o artigo fixa uma linha no diagrama cor-cor para separar as galáxias em dois grupos de acordo com um redshift de corte, que no caso do artigo foi definido como $z_{corte} = 0,75$. Um conjunto possui galáxias com baixo redshift ($z_{corte} < 0,75$), e o outro conjunto possui galáxias com alto redshift fotométrico ($z_{corte} > 0,75$). Para fazer esta divisão, foram utilizados os dados do catálogo do Deep Extragalactic Evolutionary Probe 2 (DEEP2) Redshift Survey. Este catálogo tem como objetivo observar galáxias com $z \sim 1$ e as suas propriedades.

Como a concentração de galáxias do catálogo do DEEP2 ($z \sim 1$) é deslocada em relação ao catálogo do VIPERS ($z \sim 0,7$), a linha utilizada para fazer a divisão das galáxias no artigo não pode ser a mesma. Para fixar a linha que cria os dois segmentos, o artigo testou possíveis curvas e a escolha da melhor divisória foi feita pela porcentagem de galáxias que foram corretamente rotuladas. Esta porcentagem consiste em contar o número de galáxias com redshift abaixo do z_{corte} que foram categorizadas como objetos com baixo redshift e contar o número de galáxias com redshift acima de z_{corte} que foram classificadas como objetos com alto redshift, ou seja, a porcentagem de galáxias que foram corretamente classificadas. Com isso, os autores conseguiram obter a linha que melhor dividia as galáxias, ou seja, com a melhor taxa de galáxias classificadas corretamente. Os limites desta linha podem ser vistos na tabela 5.

Para fixarmos os limites da linha que separa as galáxias em dois segmentos, em um grupo serão classificadas como galáxias com baixo redshift em outro serão classificadas

como galáxias com alto redshift; começamos com os limites definidos por Newman et al. [107] e testamos diversas configurações até que pudéssemos obter a maior taxa de galáxias classificadas corretamente. Este estudo foi feito com as galáxias do VIPERS e os seus redshifts espectroscópicos. Como as galáxias do VIPERS têm um pico de concentração em um redshift mais baixo que as galáxias do DEEP2, tivemos que deslocar o z_{corte} a fim de conseguir fazer a melhor divisão possível. Após testes, concluímos que a melhor divisão do redshift seria com $z_{corte} = 0,59$.

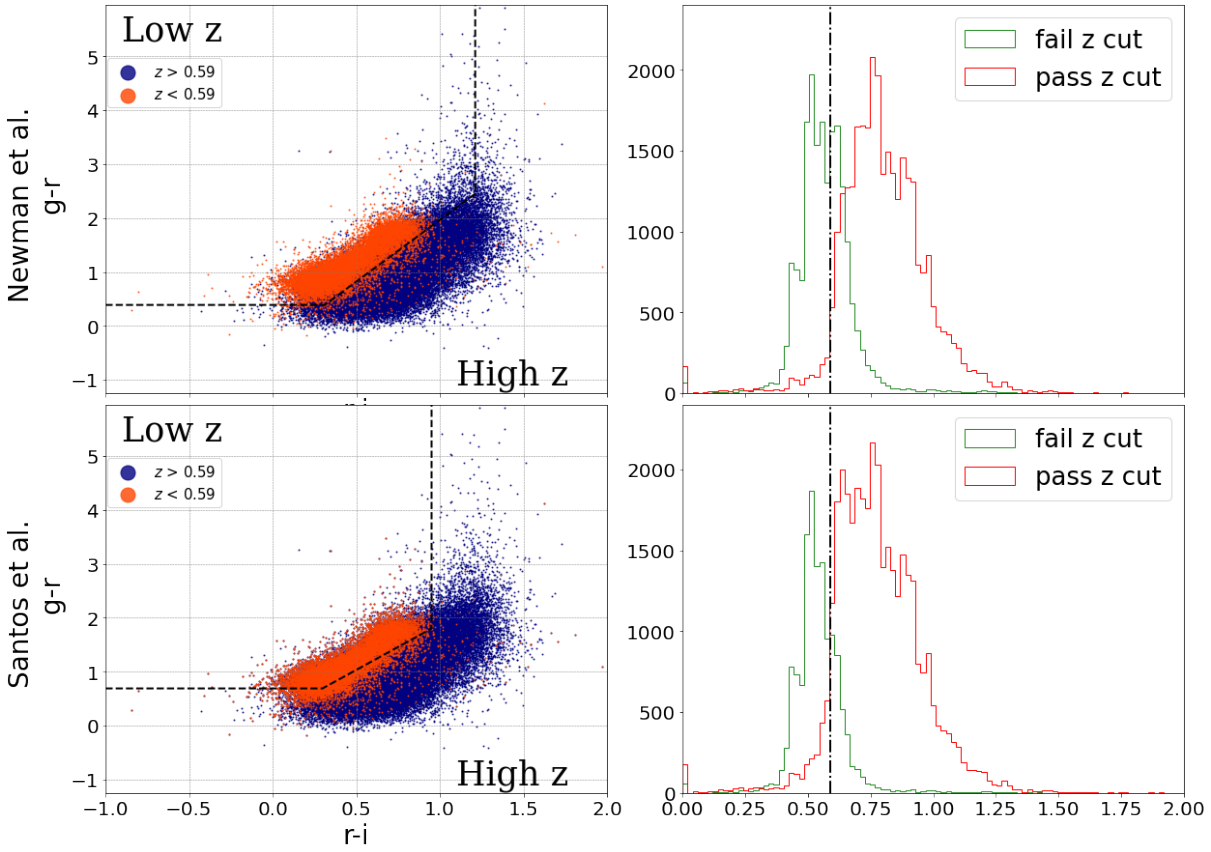


Figura 52: Imagem do estudo da seleção de galáxias. As galáxias utilizadas para esse gráfico são as do VIPERS que estão na mesma área do céu que as do DES (a descrição de como foi feita essa tabela pode ser visto na seção 3.2). O redshift utilizado na comparação é o redshift espectroscópico. O redshift de corte considerado para este estudo foi $z_{corte} = 0.59$. As imagens da primeira coluna são diagramas cor-cor das galáxias, são gráficos de dispersão da cor $m_g - m_r$ em função da cor $m_r - m_i$. As galáxias que estão em laranja têm $z_{spec} < 0,59$ e as galáxias em azul tem $z_{spec} > 0,59$. A curva tracejada preta é a linha que delimita os dois conjuntos de galáxias, acima desta linha são as galáxias classificadas com baixo redshift e as galáxias abaixo desta linha são as galáxias classificadas com alto redshift. As imagens da segunda coluna são os histogramas em função dos redshifts espectroscópicos do VIPERS de acordo com a categoria em que foram empregadas. A curva verde representa as galáxias que ficaram acima da linha tracejada ("*fail z cut*"), ou seja, classificadas com baixo redshift. A curva vermelha são as galáxias que ficaram abaixo da linha tracejada ("*pass z cut*"), ou seja, classificadas com alto redshift. A linha tracejada corresponde ao redshift igual a $0,59$. A primeira linha representa a delimitação feita por Newman et al. e a segunda linha a divisão feita em nosso trabalho.

Na figura 52, podemos comparar os limites definidos por Newman et al. com os limites que foram considerados neste trabalho com base nos limites fixados para as galáxias do VIPERS (tabela 5). Os limites da tabela foram utilizados para dividir as galáxias em dois conjuntos. As taxas de galáxias do VIPERS classificadas corretamente para os limites da tabela 5: Newman et al.: 82,39% e Santos et al.: 87,83%. Este resultado indica que a nossa linha para a divisão da galáxias consegue classificar as galáxias do VIPERS de uma forma melhor que as do Newman et al. A figura 52 traz os diagramas cor-cor de $m_g - m_r$ em função de $m_r - m_i$ de todos os redshifts fotométricos calculados pelas quatro máquinas de aprendizado trabalhadas (GPz, Keras e ANNz).

Observando os histogramas da figura 52, podemos ver como ficou a divisão de galáxias com base no redshift espectroscópico de cada divisão. Em uma divisão ideal, todas as galáxias com $z_{spec} < 0,59$ (curva verde) ficariam à esquerda da linha tracejada preta e todas galáxias com $z_{spec} > 0,59$ (curva vermelha) ficariam à direita desta mesma linha. É possível ver na imagem que a divisão proposta por Newman et al. [107] tem uma maior sobreposição destas curvas, enquanto a nossa divisão consegue separar melhor as galáxias.

Após este estudo, foram escolhidos 10 pixels não consecutivos, ou seja, pixels geograficamente espaçados pelo catálogo do DES para testar os resultados das máquinas de aprendizado. Foram feitos diagramas de cor-cor da mesma forma como mostrado na figura 52 com as galáxias destes 10 pixels e usando os valores de redshift fotométricos calculados pelas máquinas. O resultado é a figura 53.

Na primeira coluna da figura 53, temos os diagramas apenas com os redshifts sem fazer qualquer corte em relação ao λ. Desta coluna, podemos ver que há um número considerável de galáxias classificadas erroneamente, é possível ver isto na quantidade de pontos laranjas abaixo da linha (deveriam estar no conjunto de baixo redshift) e galáxias azuis na parte superior do gráfico. O resultado disto é que a acurácia (*score*) desta divisão está em torno de 80% para todas as máquinas, sendo que o keras tem a melhor porcentagem. Nas colunas seguintes é aplicado um corte de λ que constitui em descartar todas as galáxias cujo λ é menor que $0,45$. Ter um λ significa que há uma grande chance de a máquina não ter conseguido calcular o redshift fotométrico de forma satisfatória, por conta da galáxia ter valores fotométricos muito diferentes das galáxias do VIPERS.

A segunda coluna da figura 53 apresenta os diagramas cor-cor com as galáxias com $\lambda < 0,45$ desconsideradas. Este λ foi calculado através da árvore que recebe apenas as 4 cores como *input* ($\mathtt{k} = 4$). Para esta coluna, podemos ver que a acurácia de todos os diagramas aumentou, o que implica em uma divisão com menos galáxias classificadas de forma errada. A visualização disto na figura 53 é o fato de haver menos sobreposição dos pontos de cores diferentes.

A terceira coluna da figura 53 apresenta os diagramas cor-cor com as galáxias com $\lambda < 0,45$ desconsideradas. Este λ foi calculado através da árvore que recebe as 4 cores e a magnitude m_i como *input* ($\mathtt{k} = 5$). Para esta coluna, podemos ver que a acurácia de todos

os diagramas é a maior das três colunas da figura, apesar de não haver tanta diferença em relação a segunda coluna. Há menos sobreposição dos pontos de cores diferentes, novamente a diferença para a segunda coluna não é muito grande.

A figura 53 mostra como o corte das probabilidades λ aumenta a taxa de galáxias classificadas corretamente. Em todos os casos, podemos ver que as galáxias com redshift menor que o redshift de corte (laranja), apesar de estarem concentradas acima da linha, apresentam uma presença considerável de galáxias com baixo redshift abaixo da linha. Isso implica em uma parcela de galáxias classificadas de forma errônea. Já para as outras colunas, a quantidade de galáxias com $z_{phot} < 0.59$ que estão no conjunto de galáxias com alto redshift diminuiu bastante. A comprovação desse resultado é a taxa de galáxias classificadas corretamente que aumentou em relação à primeira coluna. Além disto, podemos ver que, entre as estruturas de dados, a K-d Tree com 5 dimensões foi a que conseguiu melhor separar as galáxias com baixa probabilidade de acurácia do redshift fotométrico, isto é visto na diminuição de galáxias laranjas abaixo da linha fixada em relação à mesma imagem para a estruturas de dados com 4 dimensões.

Observando apenas os valores de z_{phot} de cada máquina, podemos ver que o `ANNz` é o que tem as galáxias com baixo redshift mais dispersas embaixo da linha fixada. Porém, ao consultarmos o *score* de cada máquina, a acurácia do `ANNz` está dentro do padrão e maior que a acurácia do `GPz`. Isto acontece pois apesar de ter mais galáxias com $z_{phot} < 0,59$ no grupo de alto redshift da divisão, o `ANNz` conseguiu ter uma menor porcentagem de galáxias com $z_{phot} > 0,59$ classificados como baixo redshift. Por conta disso, o que o `ANNz` perdeu em acurácia no grupo de baixo redshift, ganhou no grupo de alto redshift. Este resultado ajuda a balancear a taxa.

Comparando os valores da taxa de galáxias classificadas corretamente da primeira e da última coluna (foi a coluna com a maior taxa em qualquer um dos aprendizados de máquina), podemos ver que em todas houve uma melhoria de no mínimo 6% em relação ao diagrama sem o corte. Isso demonstra que a probabilidade λ calculada para cada galáxia consegue fazer a distinção entre galáxias do `DES` que são fotometricamente similares ou não às galáxias do `VIPERS`.

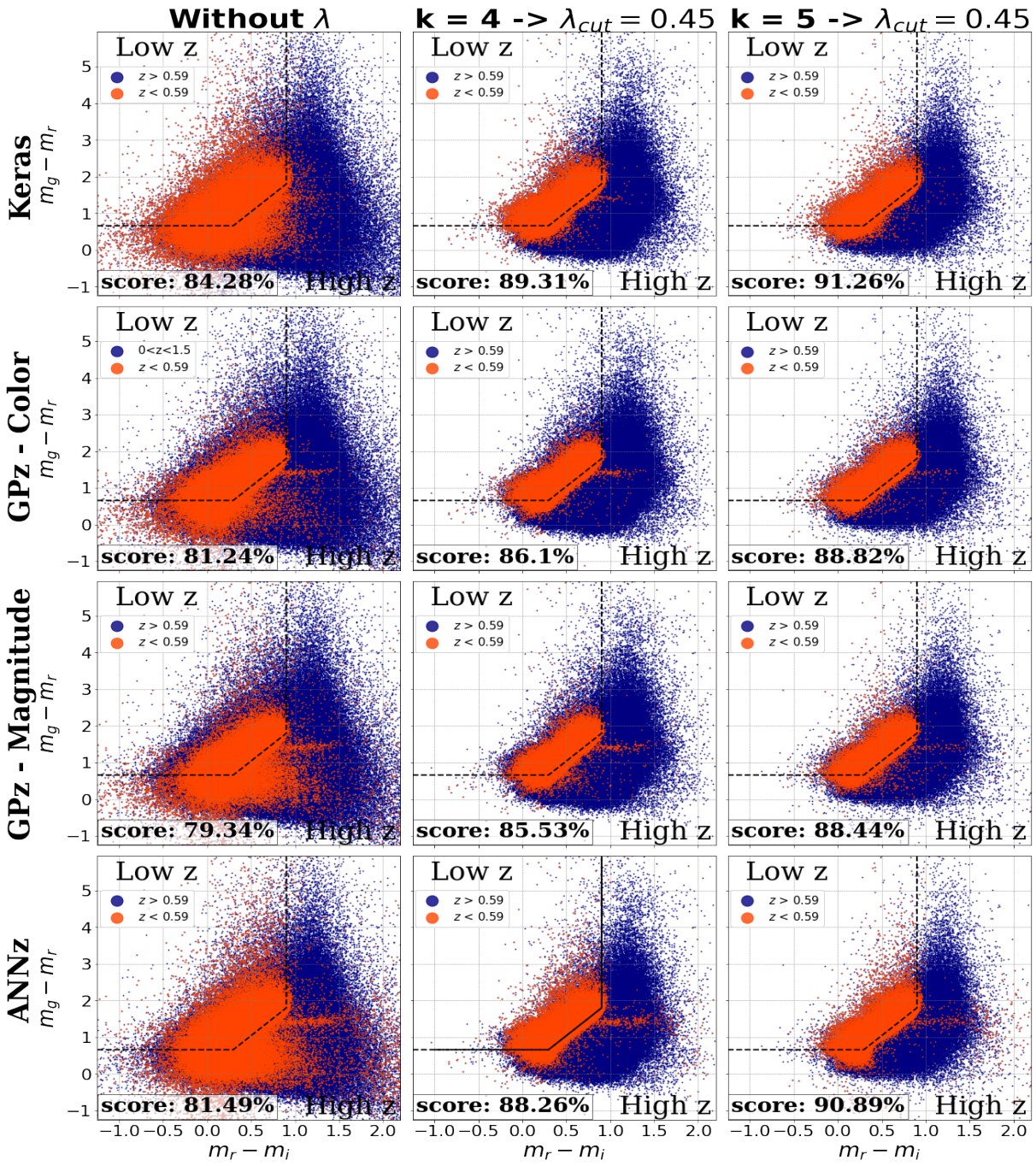


Figura 53: Diagrama cor-cor dos redshifts fotométricos com λ aplicado nas galáxias para os redshifts fotométricos calculados pelas quatro máquinas de aprendizado trabalhadas (`GPz`, `Keras` e `ANNz`). Na primeira coluna temos os redshifts fotométricos para todas as galáxias dos 10 pixels. As galáxias em laranja são que têm $z_{phot} < 0.59$ e as galáxias em azul as que têm $z_{phot} > 0.59$. O *score* indicado em cada gráfico é a porcentagem de galáxias que foram classificadas corretamente, ou seja, galáxias com redshift fotométrico baixo que foram classificadas como galáxias com redshift abaixo do redshift de corte ($z_{corte} = 0.59$) e galáxias cujo redshift fotométrico é alto e foram classificadas como galáxias com redshift acima do redshift de corte. Nas outras duas colunas foi aplicado o corte através da probabilidade λ calculada na estrutura de dados. Em ambas as colunas todas as galáxias com λ abaixo de 0,45 foram desconsideradas no diagrama. Na segunda coluna o corte foi aplicado através da probabilidade calculada pela árvore de dados de 4 dimensões e na terceira coluna o corte foi aplicado através da probabilidade calculada pela árvore de dados de 5 dimensões.

6.3. Próximos passos

Neste trabalho conseguimos criar um catálogo de galáxias do `DES` com 4 redshifts fotométricos calculados pelas máquinas de aprendizado do `GPz`, `ANNz` e `keras`. Além disto, criamos uma estrutura de dados que divide as galáxias em grupo de acordo com a sua fotometria e calcula a acurácia dos redshifts fotométricos.

Com este catálogo, o próximo passo será gerar mapas da distribuição dos redshifts fotométricos. Esta distribuição é dividida em *bins* de redshift. Em cima destes mapas serão aplicadas máscaras do `DES`, de forma que seja possível calcular o espectro de potência angular, como discutido no artigo *Cosmological Measurements from Angular Power Spectra Analysis of BOSS DR12 Tomography* de Loureiro et al. Os mapas gerados da distribuição dos redshifts fotométricos podem ser vistos na figura 54.

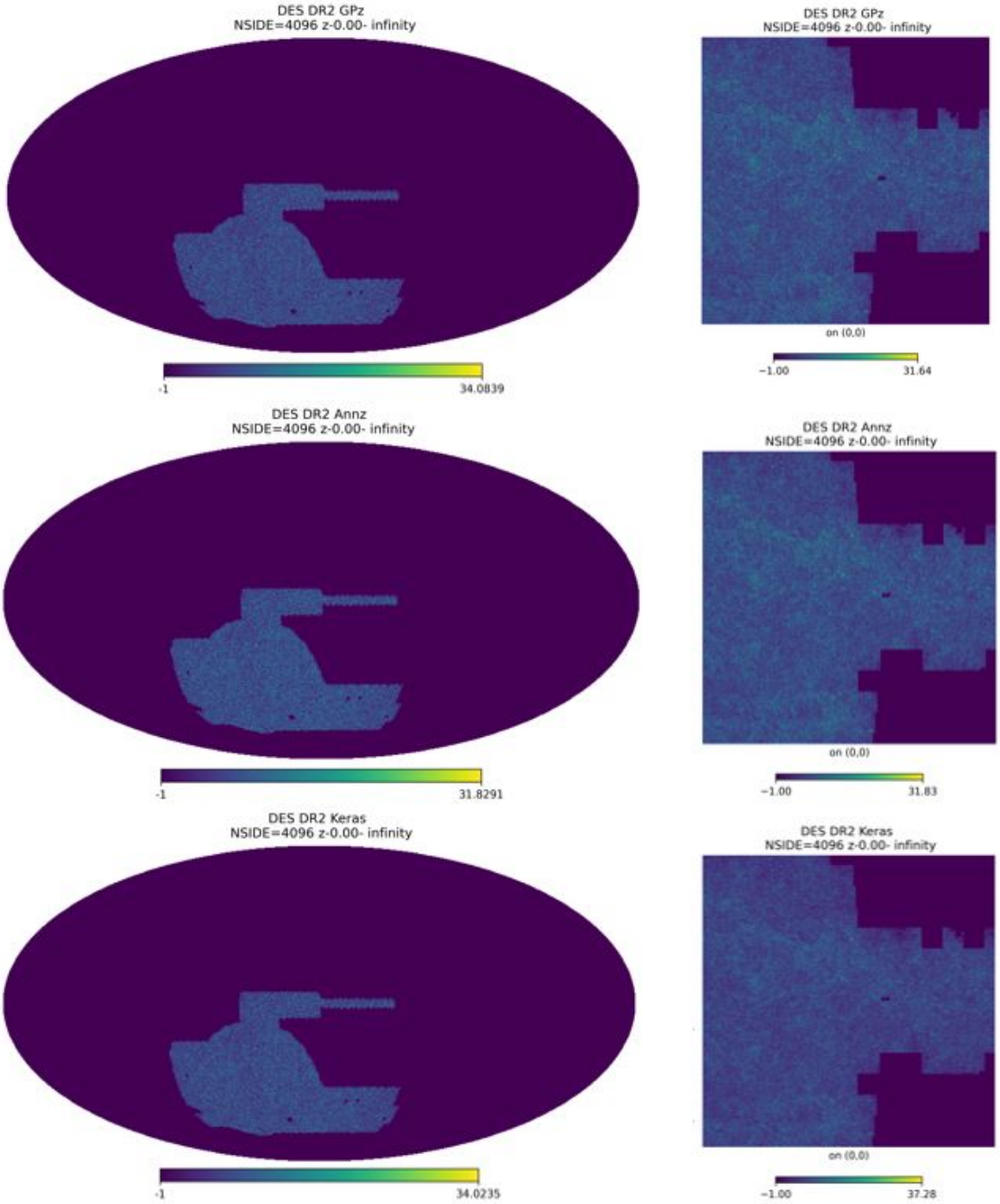


Figura 54: Mapas das galáxias do `DES` em função dos redshifts fotométricos. Os mapas de cada linha foram feitos com os redshifts fotométricos calculados neste trabalho. A primeira linha utilizou o `GPz - Color`, a segunda linha, o `ANNz` e a terceira linha, o `keras`. A primeira coluna é o mapa de todo o céu com as galáxias do `DES` e a segunda coluna é um pedação do mapa na primeira coluna ampliado.

7. Conclusão

Neste trabalho utilizamos três modelos de aprendizados de máquina para o cálculo do redshift fotométrico: `GPz`, `ANNz2` e `Keras`. Nos três modelos, foram utilizadas galáxias do catálogo astronômico do `VIPERS` como conjunto de treinamento dos aprendizados de máquina. Na fase de desenvolvimento dos aprendizados de máquina, foram utilizados os dados fotométricos das galáxias do `VIPERS` para definir os valores dos seus parâmetros. Cada uma destas máquinas utiliza um algoritmo diferente para o cálculo do redshift, tais algoritmos foram detalhados no capítulo 5. É possível ver a diferença dos resultados de cada máquina no capítulo 6.

Todas os 4 aprendizados de máquina (`GPz - Color`, `GPz - Magnitude`, `ANNz2` e `Keras`) conseguiram predizer com acurácia os redshifts fotométricos, incluindo o aprendizado de máquina do `keras` que nós construímos. Conseguimos afirmar isso através das métricas descritas no capítulo 6, o viés, o desvio padrão e a taxa de galáxias dentro do padrão nas 4 máquinas, que tiveram valores parecidos.

Conseguimos provar o quanto ter um conjunto de galáxias representativas afeta a precisão da máquina. Ao utilizarmos informações fotométricas de um catálogo cujo redshift das galáxias está concentrado entre $0,4$ e $1,0$, podemos ver que, para galáxias dentro deste intervalo, as máquinas conseguiram predizer com acurácia os redshifts fotométricos. Entretanto, para galáxias fora deste intervalo, os valores das métricas ficaram aquém dos valores considerados bons. Este resultado foi universal para as 4 máquinas. Além disso, para contornar o limite da magnitude no filtro `i` imposto pelo catálogo do `VIPERS`, utilizamos as cores (diferença entre duas magnitudes) como informação fotométrica para o treinamento dos aprendizados de máquina. Considerando tudo isso, conseguimos criar um catálogo com as galáxias do `DES` que contêm o redshift fotométrico calculado por cada um dos 4 aprendizado de máquina trabalhados nesta dissertação.

Com o intuito de verificar a acurácia dos redshifts fotométricos, foi desenvolvida uma estrutura de dados chamadas K-d Tree. Esta estrutura separa as galáxias em subconjuntos de acordo com os seus dados fotométricos. Em nosso trabalho construímos uma árvore de 12 camadas, o que resultou em 4096 subgrupos separados com base na fotometria. Além disso, foi criada um aprendizado de máquina que utiliza o algoritmo do `k`-ésimo vizinho mais próximo para o cálculo da probabilidade λ. Esta probabilidade mostra o quão preciso é o cálculo do redshift fotométrico feito pelos aprendizados de máquina `GPz`, `ANNz` e `keras`. Este valor indica se o conjunto de treinamento do `VIPERS` é representativo de cada uma das galáxias do `DES`. Se for representativo, o valor da probabilidade será mais próximo de 1, e se não for representativo, teremos $\lambda \sim 0$.

Utilizando os parâmetros definidos por Newman et al. [107], conseguimos provar que esta probabilidade está bem ajustada através da figura 53. Podemos ver que, ao utilizarmos o corte de galáxias, ou seja, apenas considerar os valores dos redshifts fotométricos

de galáxias com $\lambda \geq 0.45$, a divisão entre as áreas dos diagrama cor-cor com base nos valores do redshift ficou bem mais nítida e parecida com a do `VIPERS`.

Por fim, o objetivo desta dissertação de desenvolver e aprimorar as técnicas de cálculo do redshift fotométrico foi cumprido. Conseguimos desenvolver um aprendizado de máquina (`keras`) comparável com os aprendizados de máquina disponíveis ao público (`GPz` e `ANNz`). Além disso, conseguimos criar uma estrutura de dados que consegue nos dizer a acurácia dos redshifts fotométricos calculados por estes aprendizados de máquina. Os próximos passos são utilizar estes dados para gerar mapas em função dos redshifts fotométricos das galáxias do `DES`. A partir destes mapas, é possível fazer o estudo do espectro de potência angular.

Apêndice

A. História do aprendizado de máquina

O conceito de uma máquina que poderia aprender e ter pensamentos semelhantes aos humanos inicialmente veio do filme alemão "Metropolis", de 1937 [108], que introduziu, pela primeira vez, o conceito de robôs com pensamento de humanos [109]. Após isso, Alan Turing (23/06/1912 - 07/06/1954) criou o "Turing Test", que tinha como objetivo testar a capacidade da máquina de exibir um desempenho similar ao de um ser humano. Alguns anos depois, Arthur Samuel (05/12/1901 - 29/07/1990) escreveu o primeiro programa de aprendizado de máquina a partir de um jogo de damas. O programa examinava os movimentos vencedores e incorporava estes movimentos em seus subsequentes jogos.

Na década de 50, o neurofisiologista Warren McCulloch (16/11/1898 - 24/09/1969) e o matemático Walter Pitts (23/04/1923 - 14/05/1969) escreveram um artigo sobre como os neurônios funcionam relacionando-os com um circuito elétrico ([110]). Algum tempo depois, Donald Hebb (22/07/1904 - 20/08/1985) escreveu o livro "The Organization of Behavior" em que argumenta que o aprendizado do ser humano é feito pelas vias neurais, que são reforçadas a cada vez que são usadas, o que implica justamente no aprendizado do ser humano [111].

Com a junção dessas duas frentes sobre aprendizado de máquina e um melhor conhecimento das conexões neurais, na década de 50, foi possível uma primeira tentativa de rede neural por Nathanial Rochester (14/01/1919 - 08/06/2001) na IBM [112]. Esta experiência falhou, mas abriu caminho para novos estudos, como o do projeto de pesquisa de verão de Darthmouth, que focou em inteligência artificial [113].

E, mais tarde, na invenção de Frank Rosenblatt (11/07/1928 - 11/07/1971) do Perceptron, que é a rede neural mais antiga em uso até os dias de hoje. Esta rede é eficiente na classificação binária dos *inputs* [114, 115, 116]. Esse algoritmo tem uma função linear que calcula o resultado a partir dos dados de entrada com coeficientes dos pesos predeterminados. Dado o resultado desta função para cada dado de entrada, é definida a sua classe. Se for positivo, a função retorna 1 e, se for negativo, retorna 0. Esse algoritmo é bem primitivo e limitado a redes com poucas variáveis.

Após este significativo avanço dos mecanismos de aprendizado, houve uma supervalorização do que esses algoritmos poderiam fazer. Porém, tais perspectivas foram limitadas pela capacidade da tecnologia da época. Ao mesmo tempo, a literatura começou a questionar o poder de "máquinas pensantes", tornando-as vilãs, já que poderiam se comportar como humanos, o que deu início a pensamentos filosóficos sobre as consequências disso. Esses dois motivos foram os principais causadores da perda de interesse do público e dos pesquisadores em geral neste tema na época.

Apesar disso, nas décadas de 60 e 70, houve alguns avanços nessa área, tais como a criação do algoritmo Vizinho mais próximo (*Nearest Neighbor*), que considera a distância euclidiana dos parâmetros dos dados de entrada. Este algoritmo foi utilizado neste trabalho e detalhado no capítulo 4. Além disso, foram criadas redes que usam matrizes para rodar a parte artificial do aprendizado, o que resultou em múltiplos dados de saída para a rede.

Com o interesse nesse tema renovado, no início dos anos 80, Gerald Dejong introduz o conceito de aprendizado baseado em explicações (*Explanation Based Learning*), que consiste no computador analisar um conjunto de dados antes de rodar [117]. Deste modo, é possível determinar os pesos de cada parte da rede neural antes de serem inseridos os dados de entrada que vão ser utilizados para obter o resultado. Isso significa que Dejong criou a ideia de um conjunto de treinamento que ajuda a máquina a calibrar as funções antes de rodar os dados principais. Ainda nessa época, John Hopfield criou aprendizados de máquina bidirecionais, isto é, a rede neural não tinha apenas uma direção, a própria rede poderia voltar para outros nós. Isso significa que uma mesma função poderia ser repetida diversas vezes ao longo do caminho neural se fosse necessário [118].

Em 1986 grupos de pesquisa concebem a ideia de retro propagação, que tem como objetivo medir a acurácia da rede e ajudá-la a melhorar o seu desempenho ao modificar o valor dos seus pesos [119]. Em suma, este algoritmo tem como intuito diminuir o erro propagado ao longo da máquina, carregando consigo uma forma de reconhecer e diminuir esses erros. Para isso, são necessárias diversas iterações da máquina para aprender esses erros e redistribuir os pesos em cada função. A retro propagação está detalhada na seção 2.1.3.

Entre 1996 e 1997, o supercomputador Deep Blue, criado pela IBM, ganhou do então campeão mundial de Xadrez Garry Kasparov. Essa máquina inicialmente era capaz de analisar 100 milhões de jogadas por minuto e, em seu primeiro combate com Kasparov, conseguiu ganhar dois jogos, terminando a partida com o placar de 4 a 2 para o campeão. No outro ano, houve uma revanche com a equipe da IBM tendo melhorado o programa que agora conseguia analisar 250 milhões de jogadas por segundo, e, nessa partida, o jogo foi mais equilibrado, terminando com um placar de 3 ½ - 2 ½. Esta partida teve uma polêmica, pois, na jogada de número 44 de um dos jogos, a máquina fez um lance contraintuitivo que não tinha nenhuma função estratégica, o que confundiu Kasparov. O campeão até insinuou que havia alguém manipulando a máquina. Essa dúvida permaneceu até 2014, quando o enigma foi revelado: houve um defeito no código e, como a máquina estava programada para não "travar" o jogo, fez qualquer movimento válido apenas para continuar o jogo [120, 121].

Já nos anos 2000, essa tecnologia viu um novo avanço significativo com Geoffrey Hinton criando o termo `aprendizado de máquina de aprendizado profundo` (*deep learning*), que consiste em algoritmos que fazem com que o computador consiga ver e dis-

tinguir objetos e textos em imagens e videos [122, 123, 124]. Em pouco tempo, grandes empresas como Microsoft, Google, IBM, Amazon, Facebook, entre outras, conseguiram escrever códigos em que a máquina consegue reconhecer animais e pessoas em imagens e vídeos do mesmo modo que os humanos conseguem.

Tudo isso faz com que essa área receba muita atenção e investimento, porém, assim como no meio do século passado, enfrenta um problema de incapacidade do sistema. Os processadores hoje têm um limite que faz com que redes neurais levem semanas para aprender. É nessa etapa que se encontra o aprendizado de máquina. Por mais que estejamos rodeados deles e utilizando-os a todo tempo, tanto no nosso cotidiano quanto nas nossas pesquisas, ainda há uma grande parte desta tecnologia que não consegue alcançar toda a sua capacidade por conta de custo operacional e de armazenamento. Isso acontece pela falta de tecnologia capaz de suportar esses algoritmos e consiga executar o aprendizado de máquina de forma eficiente e rápida.

B. Baixar dados do DES

Abaixo o comando para baixar os dados do DES utilizado nesse trabalho.

```
1    SELECT *
2    FROM DR2_MAIN main
3    WHERE
4    main.EXTENDED_CLASS_COADD > 2 and
5    main.MAG_AUTO_I < 24;
```

Na primeira linha do programa, o código seleciona todo o catálogo da tabela principal do DES e extrai os objetos astronômicos com `EXTENDED_CLASS_COADD` maior que 2. Esta propriedades é a classificação morfológica dos objetos com uma acurácia de 99%. Os objetos do catálogo podem ter a seguinte classificação:

- **0**: Grande confiança do objeto ser estrela.
- **1**: Confiança do objeto ser uma estrela.
- **2**: Confiança do objeto ser uma galáxia,
- **3**: Grande confiança do objeto ser galáxia.
- **-9**: Sem dados para classificar.

Sabendo disso, decidimos pegar todas os objetos com valor de classificação maior que 2. O que implica em um catálogo com 99% de confiança dos objetos serem galáxias.

Como o `VIPERS` tem uma restrição da magnitude na banda `i` de até 22,5, fizemos também essa restrição em `main.MAG_AUTO_I < 24` para o catálogo do `DES`. Escolhemos o número 24 para termos um maior número de galáxias para manipular e não ficarmos no limite definido pelo `VIPERS`.

C. Tabela de catálogos astronômicos

Abaixo uma tabela dos principais catálogos astronômicos citados neste trabalho.

Catálogos		Localização do telescópio	Objetivo
Siglas	**Nome**		
BINGO [1, 2, 3, 4, 5, 6, 7]	**B**aryon Acoustic Oscillations from **I**ntegrated **N**eutral **G**as **O**bservations	Paraíba (Brasil)	Observar a distribuição de matéria através do Hidrogênio atômico (HI) por um radiotelescópio
DES [9, 12]	**D**ark **E**nergy **S**urvey	Coquimbo (Chile)	Mapear objetos astronômicos e estudar detalhes da energia escura que está acelerando a expansão do Universo através da fotometria.
VIPERS [13, 65, 125]	**VIMOS P**ublic **E**xtragalactic **R**edshift **S**urvey	Havaí (Estados Unidos)	Mapear a distribuição espacial de galáxias com $z \sim 1$ através da espectroscopia
SDSS [126, 10, 127]	**S**loan **D**igital **S**ky **S**urvey	Novo México (Estado Unidos)	Mapear objetos astronômicos através da fotometria e da espectroscopia

Tabela 6: Tabela com o resumo dos principais catálogos astronômicos citados neste trabalho.

D. Configuração da estrutura de dados

D.1. Estrutura de dados

O código abaixo foi criado para a estrutura de dados para a divisão das galáxias. O programa detalhado é para uma árvore de 4 dimensões, para uma árvore de K = 5, a estrutura é a mesma, porém adicionando mais um parâmetro no *input*.

Na seção 4.3 descrevemos as 4 funções necessárias para a estrutura da K-d Tree. Abaixo iremos detalhar cada uma desta funções.

- 1ª função: A primeira função calcula a distância euclidiana com os parâmetros das galáxias (as cores).

```
def distancia(point1,point2):
    w1, x1, y1, z1  = point1
    w2, x2, y2, z2  = point2

    dw = w1 - w2
    dx = x1 - x2
    dy = y1 - y2
    dz = z1 - z2

    return math.sqrt(dw*dw + dx*dx + dy*dy + dz*dz)
```

A função serve para calcular a distância fotométrica de duas galáxias. A função recebe dois pontos, que são as duas galáxias junto das suas coordenadas fotométricas, ou seja, as cores $m_g - m_r$, $m_r - m_i$, $m_i - m_z$ e $m_z - m_Y$. Cada cor define uma dimensão da estrutura. Então a função separa essas coordenadas e calcula a diferença entre as cores das duas galáxias em cada dimensão. O resultado é a raiz quadrado da soma das cores ao quadrada, como mostrado na equação (62).

- 2ª função: A segunda função calcula a menor distância entre a galáxias avaliada e os dois nós seguintes. A função retorna o ponto central com a menor distância da galáxia examinada.

```
def menor_distancia(pivot, p1, p2):
    if p1 is None:
        return p2

    if p2 is None:
        return p1

```

```
    d1 = distancia(pivot,p1)
    d2 = distancia(pivot,p2)

    if d1<d2:
        return p1
    else:
        return p2
```

Esta função recebe as informações de três galáxias, a que está sendo analisada (`pivot`) e as duas galáxias que são os pontos centrais dos subgrupos a serem avaliados. Esta parte do código tem como objetivo determinar a qual subgrupo a galáxia analisada deve pertencer. Então ao receber as informações fotométricas das galáxias, a função calcula a distância fotométrica entre os objetos através da função `distancia` detalhada anteriormente. E retorna a galáxia que está mais perto da amostra analisada. Esta galáxia será o ponto central do subgrupo do qual a galáxia analisada irá pertencer.

- 3ª função: A terceira função determina para qual subgrupo a galáxia examinada deve ir com base na menor distância fotométrica calculada pela função distância definida pela função `menor_distancia`.

```
def kdtree_ponto_mais_perto(root, ponto, depth=0):
    if root is None:
        return None

    axis = depth % k

    proximo_no = None
    no_oposto = None

    if ponto[axis]< root['point'][axis]:
        proximo_no = root['left']
        no_oposto = root['right']
    else:
        proximo_no = root['right']
        no_oposto = root['left']

    melhor = menor_distancia(ponto,
                             kdtree_ponto_mais_perto(proximo_no, ponto,
    depth+1),
                             root['point'])

    if distancia(ponto,melhor)>abs(ponto[axis] - root['point'][axis]):
        melhor = menor_distancia(ponto,
                             kdtree_ponto_mais_perto(proximo_no, ponto,
    depth+1),
```

```
                                melhor)

    return melhor
```

Este código combina as funções detalhadas acima e usa os valores fotométricos das galáxias para definir para qual subgrupo a galáxia analisada deve pertencer. A equação fez isso ao analisar qual é o ponto central mais perto das galáxias avaliadas. Por fim, o código retorna o subgrupo da melhor posição da galáxia. Esta função é recursiva, então, após a galáxia ser analisada para uma determinada camada, o objeto é examinado através do mesmo processo para dois subgrupos do subconjunto em que já se encontrava. Isso acontece até que o programa posicione a galáxia em um dos 4096 subgrupos finais da árvore.

Por exemplo, se olharmos a figura 24, temos o ponto central `4` com os subgrupos da próxima camada como `6` e `7`. Esta função determina se a galáxia irá para o subgrupo `6` ou o subgrupo `7` com base na distância da galáxia a cada um dos pontos centrais. Há um caso de quando a distância até um subgrupo é menor, porém a galáxia pertence a outro subgrupo. Isto acontece com as galáxias que estão à margem do grupo a que pertencem. Para evirar de que a galáxia seja colocada no subconjunto errado, a função `kdtree_ponto_mais_perto` contorna este problema com uma condição no fim do código.

- 4ª função: A quarta função é responsável por definir a estrutura da árvore.

```
def construir_kdtree (pontos, k, depth=0):
    n = len(pontos)

    if n<=0:
        return None

    axis = depth % k

    pontos_ordenados = sorted(pontos, key=lambda point: point[axis])

#-----------------------------------------------------------------
    if depth==12:
        grupos_mag.append(pontos_ordenados[int(n/2)])
#-----------------------------------------------------------------
    return {
        'point': pontos_ordenados[int(n/2)],
        'left': construir_kdtree(pontos_ordenados[:int(n/2)], k, depth
    +1),
        'right': construir_kdtree(pontos_ordenados[int(n/2)+1:], k,
    depth+1)
    }
```

A função `construir_kdtree` constrói a estrutura de dados K-d Tree. Primeiro, a função escolhe qual será a dimensão a ser utilizada para o corte. Começando com a dimensão da primeira cor $(m_g - m_r)$ e seguindo para a segunda cor e continuando o processo na ordem (primeiro $m_r - m_i$, depois $m_i - m_z$ e por último $m_z - m_Y$). Como o processo de troca é cíclico, depois dessas 4 camadas o programa volta para a primeira cor $(m_g - m_r)$ e assim por diante. Depois de definir a dimensão, o código ordena os dados na dimensão definida. Isso significa organizar as galáxias do conjunto de acordo com o seu valor na dimensão fixada. Este procedimento possibilita o código de escolher a galáxia central desse conjunto na dimensão considerada.

Para a décima segunda camada, o código recolhe as galáxias que foram denominadas como nós nessa fase. Estas 4096 galáxias serão as nossas guias em relação a divisão das galáxias do `DES`.

Para outras camadas, a função retorna a galáxia que é o nó da divisão dos grupos e os dois subconjuntos formados. Este código é recursivo, então ambos os subconjuntos passarão por esse processo novamente para serem divididos em subgrupos, porém com outra dimensão fixada.

D.2. Aprendizado de máquina

O código descrito a seguir tem como objetivo criar a estrutura do aprendizado de máquina para o cálculo da probabilidade λ das galáxias serem parecidas fotometricamente. O programa detalhado é para uma árvore de 4 dimensões. Para uma árvore de `K` $= 5$, o processo é o mesmo, muda apenas a dimensão dos *inputs*.

```
def ML_colors(dataset):
    model = KNeighborsClassifier(n_neighbors=20, weights= 'distance')
    X_train, y_train, X_valid, y_valid, X_test, y_test =
   train_valid_test_split(dataset.iloc[:,0:4], dataset.iloc[:,5],
   train_size=0.65, test_size=0.25, valid_size=0.1, random_state=None)

    model.fit(X_train, y_train, valid_set = {X_valid, y_valid})

    return model
```

A função mostrada acima é responsável por criar o aprendizado de máquina que calculará λ. O primeiro passo é criar o modelo da máquina, como discutido na seção 4.2, o algoritmo utilizado para isso é o K-ésimo vizinho mais próximo. Definimos o número de vizinhos como igual a 20 pois foi o que nos forneceu uma melhor acurácia do valor calculado para o valor real. A variável `weights` define qual o parâmetro que será utilizado para definir os vizinhos da galáxia, nesse caso escolhemos a distância fotométrica definida na equação (62).

Depois disso, o conjunto de galáxias é dividido em três grupos, o de treinamento, o de validação e o de teste. Sendo que a proporção de galáxias em cada grupo é de 65%, 10% e 25% respectivamente.

Do conjunto de treinamento temos que `dataset.iloc[:,0:4]` é o subgrupo das cores das galáxias, ou seja, são as 4 cores ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$) de todas as galáxias da tabela com classificação binária descrita na seção 3.2. E `dataset.iloc[:,0:5]` é a coluna binária da mesma tabela e em nosso caso, será o *output* do aprendizado de máquina, a probabilidade λ.

Por fim, o aprendizado de máquina recebe os conjuntos de treinamento e validação para aprender com as amostras e criar um padrão para o aprendizado de máquina. Com isso, construímos o mecanismo que calculará a probabilidade de uma galáxia do `DES` de ser parecida em relação aos seus dados fotométricos com as galáxias do `VIPERS`.

E. Código do GPz

Este apêndice mostra as etapas mais importantes para o nosso trabalho. O programa utiliza o pacote `GPz` criado por Almosallam et al. [14][11] que está disponível ao público.

```
import GPz
```

O `GPz` calcula o redshift fotométrico utilizando processos gaussianos. A metodologia é descrita na seção 5.1. Após baixar o código do GPz, é preciso definir alguns parâmetros do aprendizado de máquina.

```
maxIter = int(maxIter)
maxAttempts = int(maxAttempts)
m = int(m)
method = 'VC'
```

Nesta etapa são definidas as configurações da maquina `GPz`. A variável `maxIter` define o número de iterações na máquina, isto significa a quantidade máxima de vezes que o `GPz` recebe os dados de entrada e otimiza os valores dos elementos da covariância. Utilizando o erro definido na seção 5.1, a máquina mede a acurácia com que calcula os valores do redshift fotométrico. Estas métricas são testadas com o conjunto de validação. Se os valores das métricas não mudarem ao serem calculadas depois de um número fixo de rodadas da máquina, sendo que este número é definido na variável `maxAttempts`, o programa para de treinar. Então temos dois casos: a máquina não consegue mais otimizar o seu resultado (métrica não muda de valor) e então a máquina para de tentar otimizar após o número definido em `maxAttempts` ou a máquina roda o número definido em `maxIter` e para de treinar independente dos valores das métricas estarem variando. A máquina para de treinar no caso do que acontecer primeiro. Em nosso trabalho fixamos `maxIter` = 150 e `maxAttempts` = 50.

A variável `m` representa o número de funções não lineares do vetor $\boldsymbol{\Theta}(\boldsymbol{x_i})$, foi fixado `m` = 25.

A última linha desta seção do código representa o método utilizado para o cálculo dos elementos da covariância dos processos gaussianos. Após os testes feitos na seção 5.1.2, foi escolhido o modelo `VC` em que as matrizes de covariâncias são diferentes para cada função de base radial $\phi_j(\boldsymbol{x_i})$.

O código do `GPz` tem uma função própria para dividir o conjunto de dados em grupo de treinamento, validação e teste.

[11] https://github.com/OxfordML/GPz

```
def split(n, trainSample, validSample, testSample):

    if(trainSample<=1):
        validSample = math.ceil(n * validSample)
        testSample = math.ceil(n * testSample)
        trainSample = min([math.ceil(n * trainSample), n - testSample -
    validSample])

    r = np.random.permutation(n)

    validSample = int(validSample)
    testSample = int(testSample)
    trainSample = int(trainSample)

    validation = np.zeros(n, dtype=bool)
    testing = np.zeros(n, dtype=bool)
    training = np.zeros(n, dtype=bool)

    validation[r[0:validSample]] = True
    testing[r[validSample:validSample + testSample]] = True
    training[r[validSample + testSample:validSample + testSample +
    trainSample]] = True

    return training, validation, testing

n, d = input.shape
trainSplit = float(train_size)
testSplit = float(test_size)
validSplit = 1 - (trainSplit + testSplit)
training, validation, testing = split(n, trainSplit, validSplit,
    testSplit)
```

Nesta etapa do código, há a divisão do conjunto de dados de entrada em três grupos: o de treinamento, o de validação e o de teste. Primeiramente, o usuário define a porcentagem de informações para cada grupo, no nosso trabalho decidimos por 60% dos dados para o treinamento (`train_size`), 20% para a validação (`1 - (trainSplit + testSplit)`) e 20% para o teste (`test_size`). Na linha `n, d = input.shape` o código calcula o número de galáxias nos dados de entrada (`n`) e o número de parâmetros de cada dado de entrada `d`. Para o `GPz` nós utilizamos dois tipos de *inputs*: 5 magnitudes (m_g, m_r, m_i, m_z e m_Y) e 4 cores e a magnitude m_i ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$) e m_i), como descrito na seção 5.1.1. Em ambos os casos temos `d=5`.

A função `split` recebe a fração de galáxias que devem ir para cada conjunto além da quantidade de galáxias utilizadas no treinamento da máquina. Esta função separa as galáxias entre os três conjuntos. A divisão é feita de forma aleatória através da linha `r`

= np.random.permutation(n). Isto significa que a ordenação das galáxias será feita de forma aleatória.

A próxima etapa do código consiste em treinar as 10 máquinas do GPz.

```
for i in range(int(number_sim)):
    model = GPz.GP(m, method=method, joint=joint,heteroscedastic=
    heteroscedastic, decorrelate=decorrelate)
    random.seed(i)
    omega = GPz.getOmega(redshift_spec, method=csl_method)
    model.train(dataset_phot.copy(), redshift_spec.copy(), omega=omega,
    training=training,
            validation=validation, maxIter=maxIter, maxAttempts=
    maxAttempts)
    pickle.dump(model, open(outpath + 'model_GPz_' + str(i+1) + ".pkl",
    "wb"))
```

Esta etapa é a fase de treinamento, em que serão fornecidas os três conjuntos de galáxias divididos anteriormente e os parâmetros do aprendizado de máquina GPz. A linha "for i in range(int(number_sim))"define o número de aprendizados de máquina do GPz que serão treinadas, este número é dado pela variável number_sim. Para o nosso trabalho definimos number_sim = 10. Isto significa que foram treinados 10 aprendizados de máquina do GPz, cada uma destas máquinas possui uma configuração inicial diferente por conta da linha random.seed(i). Antes de começar a ser treinada, o aprendizado de máquina define de forma aleatória os valores dos elementos da covariância, então os dados de entrada são fornecidos à máquina que aprende com esses dados e otimiza os valores dos elementos. A definição destes valores inciais dos elementos da covariância são gerados de forma aleatória pela função random.seed(i) em que i muda a cada máquina treinada. Isto significa que os números inciais dos elementos da covariância de cada um dos 10 aprendizados de máquina são diferentes.

A linha "model = GPz.GP(m, method=method, joint=joint, heteroscedastic = heteroscedastic, decorrelate = decorrelate)" cria a estrutura da máquina com todos os parâmetros definidos anteriormente como método (VC) e o valor do número de funções não lineares do vetor $\boldsymbol{\Theta}(\boldsymbol{x_i})$ (m = 25). O termo GPz nesta linha é o pacote do código do GPz[12] descrito no artigo do Almosallam et al. [14]. E "model.train(X.copy(), Y.copy(), omega = omega, training = training, validation = validation, maxIter = maxIter, maxAttempts = maxAttempts)" é responsável por criar a estrutura do aprendizado de máquina. O modelo recebe os dados de entrada e de saída dos conjuntos de treinamento e de validação. Estes dados são fornecidos pela tabela dataset_phot das galáxias do VIPERS definida na seção 3.2. Esta tabela contém as magnitudes das galáxias do VIPERS. Enquanto redshift_spec são os redshift espectroscópicos das galáxias do

[12]https://github.com/OxfordML/GPz

VIPERS e que são o *output* desta máquina. A otimização do aprendizado de máquina é feita dentro do código do pacote do GPz.

Por fim o programa salva (pickle.dump(model, open(outpath + 'model_GPz_' + str(i+1) + ".pkl", "wb"))) os 10 modelos de aprendizado de máquina do GPz treinados com as galáxias do VIPERS.

```
filename = os.listdir(DR2_folder)
for i in range(len(filename)):
    data_aux = Table.read(os.path.join(DR2_folder, filename[i])).
    to_pandas()
    data = remove_outliers(data_aux)

    X = data[['MAG_AUTO_G_DERED', 'MAG_AUTO_R_DERED', 'MAG_AUTO_I_DERED'
    , 'MAG_AUTO_Z_DERED', 'MAG_AUTO_Y_DERED']].values
    err = data[["MAGERR_AUTO_G", "MAGERR_AUTO_R", "MAGERR_AUTO_I", "
    MAGERR_AUTO_Z", "MAGERR_AUTO_Y"]].values

    X = np.concatenate((X, err), axis=1)

    mu_GPz = []
    mu_GPz_sigma = []

    for j in range(number_sim):
        mu, sigma, modelV,noiseV, _ = models[j].predict(X[:,:].copy(),
    model='best')
        zphot = mu.flatten()
        dataset_output['GPz_zphot' + str(j+1)] = zphot
        dataset_output['GPz_sigma' + str(j+1)] = sigma.flatten()
        mu_GPz.append(zphot)
        mu_GPz_sigma.append(sigma.flatten())

    mean_GPz = pd.Series(np.round(sum(np.array(mu_GPz)/10),6), name="
    mean_GPz")
    standard_dev_GPz = pd.Series(np.round(np.std(np.array(mu_GPz), axis
    =0),6), name="StanDev_GPz")
    mean_GPz_sigma = pd.Series(np.round(sum(np.array(mu_GPz_sigma)/10)
    ,6), name="mean_GPz_sigma")
    standard_dev_GPz_sigma = pd.Series(np.round(np.std(np.array(
    mu_GPz_sigma), axis=0),6), name="StanDev_GPz_sigma")

    #Criar um cat logo de gal xias do DES com tudo que foi calculado
    anteriormente.
    dataset_output = pd.DataFrame()
    dataset_output["COADD_OBJECT_ID"] = data["COADD_OBJECT_ID"]
    dataset_output["Mean_GPz"] = mean_GPz
    dataset_output["StanDev_GPz"] = standard_dev_GPz
```

```
32      dataset_output["Mean_GPz_sigma"] = mean_GPz_sigma
33      dataset_output["StanDev_GPz_sigma"] = standard_dev_GPz_sigma
34
35      name = filename[i][:17] + "_GPz" + filename[i][17:]
36      dataset_output.to_csv(outpath + str(name),index = False)
```

A primeira linha deste código é responsável por listar todos os pixels do `DES` disponíveis na pasta `DR2_folder`. Depois disso, o código abre um pixel por vez e coleta informações de todas as galáxias deste pixel, como as magnitudes (`X`) e as incertezas destas magnitudes (`err`).

Após criar o dataset com as informações de entrada, o redshift fotométrico (`zphot`) e a sua variância (`sigma`) são calculados em cada um dos 10 aprendizados de máquina treinados. Também é criada uma matriz contendo todos os redshifts e as variâncias calculadas para cada galáxia nos 10 aprendizados de máquina do `GPz`. A partir desta matriz são calculadas as médias (`Mean_GPz` e `Mean_GPz_sigma`) e os desvios padrão (`StanDev_GPz` e `StanDev_GPz_sigma`) dos redshift e das variâncias para cada galáxia.

Por fim, o programa cria um catálogo para o pixel em questão com o identificador das galáxias (`COADD_OBJECT_ID`), os redshifts fotométricos e as variâncias calculadas por cada máquina e a média e o desvio padrão destas propriedades (z_{phot} e variância). Este catálogo é salvo com o nome do pixel. Um exemplo deste catálogo pode ser visto na tabela 8 para o dado de entrada com as 5 magnitudes e na tabela 7 para o *input* com as 4 cores e a magnitude m_i.

F. Código do ANNz

O `ANNz` possui 18 códigos na linguagem de programação python que são utilizados de acordo com a necessidade do usuário. Para o cálculo do redshift fotométrico pelo `ANNz` nós modificamos apenas o código chamado `annz_rndReg_quick.py`. Todos os outros códigos permaneceram iguais aos que foram distribuídos ao público na página do *github*[13].

A primeira etapa do código consiste em definir os parâmetros que serão utilizados pelo programa.

```
glob.annz["outDirName"] = "BINGO_output"
glob.annz["nMLMs"] = 100
glob.annz["zTrg"] = "z"
glob.annz["minValZ"] = 0.0
glob.annz["maxValZ"] = 2.7467
glob.annz["nErrKNN"] = 100
```

Nessa parte, o termo `outDirName` define o nome da pasta em que ficarão os resultados do código. A variável `nMLMs` determina o número de aprendizados de máquina de diferentes configurações que serão utilizadas no código. Na variável `zTrg` é definido o objeto do cálculo, ou seja, o dado de saída da máquina e em `minValZ` e `maxValZ` são determinados os valores máximo e mínimo a serem calculados do redshift fotométrico. E por fim, a variável `nErrKNN` fixa o número n_{NN} de galáxias próximas para o cálculo das incertezas.

O termo `"glob"` antes dos parâmetros significa que é uma variável que o usuário pode definir ao pedir para o código rodar no comando de linha sem mexer no programa em si. Por exemplo, se o usuário quiser definir que o número de máquinas a serem treinadas do código será 50, pode-se escrever `"- -nMLMs 50"` e então, o programa treinará 50 máquinas ao invés de 100, porque o pedido do usuário no comando de linha se sobrepõem ao que está escrito no código.

Com esses valores definidos para o cálculo do redshift, o código passa por quatro etapas: pré-processamento, treinamento, otimização e estimativa.

a) Pré-processamento

Nessa etapa foi feita a separação das colunas em relação ao que seria utilizado como dado de entrada. No caso dos catálogos do `VIPERS` e do `DES` foram utilizadas as magnitudes nos filtros `grizY`. Foi utilizada a tabela das galáxias representativas do `VIPERS`, descrita na seção 3.2, para este código.

Abaixo como ficou a parte de pré-processamento (*doGenInputTrees* no programa) de dados no código:

[13] `https://github.com/IftachSadeh/ANNZ`

```
if glob.annz["doGenInputTrees"]:
    glob.annz["inDirName"] = "examples/data/"
    glob.annz["inAsciiVars"] = "F:MAG_AUTO_G_DERED; F:MAG_AUTO_R_DERED;
   F:MAG_AUTO_I_DERED; F:MAG_AUTO_Z_DERED; F:MAG_AUTO_Y_DERED; F:
   MAGERR_AUTO_G; F:MAGERR_AUTO_R; F:MAGERR_AUTO_I; F:MAGERR_AUTO_Z; F:
   MAGERR_AUTO_Y; D:z"
    glob.annz["splitTypeTrain"] = "ANNzVIPERSTrain_copy.csv"
    glob.annz["splitTypeTest"] = "ANNzVIPERSEval_copy.csv"
    runANNZ()
```

Primeiro, indicamos onde estão os catálogos de treinamento e de teste para o programa. Esta separação foi feita de maneira aleatória considerando 70% do catálogo de galáxias para treino e o restante para o teste. Se não tivéssemos feito esta separação, o próprio código a teria feito. Então em `inDirName` apontamos onde estão os dois catálogos mencionados anteriormente. E em `splitTypeTrain` e `splitTypeTest` indicamos os nomes dos catálogos de treino e de teste respectivamente.

As colunas que serão utilizadas como dados de entrada são definidas em `inAsciiVars`, sendo que todas têm que começar com `"F:"` para indicar que os valores daquelas colunas são números reais (números de ponto flutuante - *float* - na programação). Para o dado de saída, o redshift espectroscópico, temos `"D:"` (dupla precisão - *double* - na programação) no começo do nome da coluna para que possamos ter uma maior precisão no resultado da máquina.

b) Treinamento

```
if glob.annz["doTrain"]:
    for nMLMnow in range(glob.annz["nMLMs"]):
        glob.annz["nMLMnow"] = nMLMnow
        if glob.annz["trainIndex"] >= 0 and glob.annz["trainIndex"] !=
   nMLMnow:
            continue

        glob.annz["rndOptTypes"] = "ANN_BDT"
        glob.annz["inputVariables"] = "MAG_AUTO_G_DERED;
   MAG_AUTO_R_DERED; MAG_AUTO_I_DERED; MAG_AUTO_Z_DERED;
   MAG_AUTO_Y_DERED"
        runANNZ()
```

Nessa parte, para cada umas das 100 máquinas definidas anteriormente em `nMLMs`, o programa irá rodar e treinar a rede neural de acordo com a configuração de cada uma das máquinas. Depois, são definidos os tipos de máquinas a serem treinadas em `rndOptTypes`, nesse caso foram escolhidas redes neurais que utilizam de máquinas de regressão. Por fim,

em `inputVariables` são determinadas as variáveis que irão ser usadas como dados de entrada dos aprendizados de máquina. Isso faz com que todos os outros dados armazenados em `inAsciiVars` serão considerados o dado de saída (por conta de `zTrg`) ou como os dados da incerteza dos *inputs*. Isso significa que as colunas `MAGERR_AUTO_G; MAGERR_AUTO_R; MAGERR_AUTO_I; MAGERR_AUTO_Z; MAGERR_AUTO_Y` serão definidas como incertezas das variáveis do *input* e serão usadas no cálculo das incertezas do redshift fotométrico do código.

É possível fixar uma função para definir as configurações dos aprendizados de máquina como os cortes e os pesos, porém como o tipo de opções de máquinas já foi definido anteriormente em `rndOptTypes`, este passo não é necessário. A configuração para `rndOptTypes` definido como "ANN_BDT" é dada pelo programa. Esta configuração combina diversas arranjos diferentes de aprendizado de máquina no código através de uma função já definida. Isso significa que as opções estruturas são geradas internamente no ANNZ. Mais detalhes sobre os variados tipos pode ser encontrado na pasta "ANNZ/src/ANNZ_train.cpp"na documentação do programa no github [14].

c) Otimização

```
if glob.annz["doOptim"]:

    glob.annz["nPDFs"] = 1
    glob.annz["nPDFbins"] = 200

    if glob.annz["doOptim"]:
        runANNZ()
```

Na etapa da otimização, o desempenho do conjunto de aprendizados de máquina é avaliado e a melhor solução é escolhida (o código retorna como *zbest*). Além disso, são definidos quantas `PDFs` serão calculados em `nPDFs` e a quantidade de *bins* por PDF em `nPDFbins`. Para esse programa, foi escolhido calcular apenas uma `PDF` com 200 *bins*.

É possível definir cortes a partir das métricas ($\delta_{gal}, \sigma, \sigma_{68}$ e $f(\alpha\sigma)$) citadas na seção 5.2, de modo a eliminar os aprendizado de máquina que têm um desempenho inferior ao desejado. Isso é útil quando o conjunto de dados de treino não é representativo em relação ao conjunto que queremos calcular o redshift fotométrico. Como, em nosso caso, foi feito o pré-processamento dos dados de forma a usar um catálogo representativo (seção 3.2), este passo não é necessário.

d) Estimativa

```
if glob.annz["doEval"]:
    import os
```

[14] https://github.com/IftachSadeh/ANNZ

```
3
4     path = "examples/data/inputs_ANNz/"
5     filenames_in = os.listdir(path)
6
7     for i in filenames_in:
8         name = i.split(".")
9         glob.annz["inDirName"] = path
10        glob.annz["inAsciiFiles"] = i
11        glob.annz["inAsciiVars"] = "F:MAG_AUTO_G_DERED; F:
    MAG_AUTO_R_DERED; F:MAG_AUTO_I_DERED; F:MAG_AUTO_Z_DERED; F:
    MAG_AUTO_Y_DERED; F:MAGERR_AUTO_G; F:MAGERR_AUTO_R; F:MAGERR_AUTO_I;
    F:MAGERR_AUTO_Z; F:MAGERR_AUTO_Y"
12        glob.annz["evalDirPostfix"] = name[0]
13        runANNZ()
```

Com o programa já treinado, é definida a localização dos catálogos para o cálculo do redshift fotométrico em `inDirName`. O nome do catálogo a ser utilizado é dado em `inAsciiFiles` e as variáveis do catálogo fotométrico a serem utilizadas são dadas em `inAsciiVars`. Desta última função podemos ver que apesar de terem o mesmo nome, o seu conteúdo é diferente na fase do pré-processamento, justamente porque aqui não temos o valor do redshift (`D:z`). Por fim, a variável `evalDirPostfix` define o nome do catálogo final que inclui, em suas colunas, o valor do redshift fotométrico calculado pelo melhor aprendizado de máquina definido na etapa anterior, a sua incerteza e a `PDF` dividida entre os 200 *bins* fixados anteriormente. Uma demonstração de como fica o catálogo pode ser visto na tabela 9.

G. Código do Keras

O `keras` é uma interface que age como uma ponte entre o pacote do python *TensorFlow* e o nosso programa, para utilizá-lo foi preciso baixar as seguintes bibliotecas.

```
from tensorflow import keras
from tensorflow.keras import regularizers, layers
from tensorflow.keras.constraints import max_norm
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Dense, Flatten, BatchNormalization,
    Activation
from tensorflow.keras.models import Sequential
import tensorflow as tf
```

O uso desses pacotes será explicado mais a frente.

A primeira etapa do código consiste em definir as condições gerais do programa, como métrica e o número de parâmetros dos dados de entrada a serem utilizados.

```
for i in range(number_sim):
    EarlyStop = EarlyStopping(monitor='reg_mse', mode='min', patience
    =25)
    BATCH_SIZE = 64
    STEPS_PER_EPOCH = len(data)//BATCH_SIZE
    lr_schedule = tf.keras.optimizers.schedules.InverseTimeDecay(0.0001,
    decay_steps=STEPS_PER_EPOCH*1000, decay_rate=1, staircase=False)
    inputs = keras.layers.Input(5)
```

Da mesma forma como foi feito com o `GPz`, a rede neural do `keras` foi treinada 10 vezes com configurações inciais diferentes (valores dos pesos variaram).

A variável `EarlyStop` indica quando o programa deve parar de treinar de acordo com o valor que a métrica determinar. A cada rodada de treinamento, o código irá checar se a métrica está tendo calculando um valor melhor considerando os parâmetros definidos. Uma vez que for constatado que a métrica não está variando, o modelo irá parar de treinar. Neste caso, foi determinado que a métrica definida para mensurar a máquina seria o `reg_mse` que é o erro quadrático médio, definido pela equação (61).

O parâmetro `mode` significa que o treinamento irá parar quando o erro for constante a cada novo treinamento (a outra opção seria `max` para o caso de uma métrica que indica a progressão do desempenho da máquina a medida que o seu valor aumenta). A propriedade `patience` indica o número de rodadas sem melhoria que o código irá fazer até parar. Isso significa que na rodada que o programa indicar que não houve melhoria no valor da métrica, esta máquina ainda irá rodar mais 25 vezes, ou seja, treinar mais 25 vezes e se não tiver nenhum avanço no valor da métrica, irá parar. Por exemplo, se na rodada 53, o programa indicar que aa rodada 52 para 53 não houve diminuição do erro, o código contará 25 rodadas até parar. Se houver uma diminuição

do valor da métrica na rodada 65, então o programa continua rodando até indicar um novo piso do erro. Se até a rodada 78 o erro continuar igual ao valor da rodada 53, o programa irá parar. Isso garante que por mais que a máquina não tenha melhoria do seu desenvolvimento, a configuração de sua estrutura estará fixada.

A linha seguinte define o número de amostras (`BATCH_SIZE`) que serão propagadas ao longo da rede neural. Por exemplo, se tivéssemos um conjunto de dados com 1000 galáxias e fixarmos `BATCH_SIZE = 64`, isto significa que o algoritmo pegaria as primeiras 64 galáxias (da 1ª até a 64ª) do conjunto de treinamento para treinar a rede neural. Com isso, a rede atualizou os valores dos seus pesos para se adequar ao conjunto de dados. Depois, pegou as próximas 64 galáxias (da 64ª até a 128º) e treinou a mesma rede neural. Novamente, a rede neural atualizou os valores dos seus pesos para aumentar a sua acurácia. Este procedimento acontece até acabarem as amostras, no caso do exemplo como 1000 não é divisível por 64, o último conjunto de amostras treinadas teria 40 amostras apenas. A vantagem deste processo é que requer menos memoria da máquina, já que a rede é treinada com menos amostras a cada vez, o que acarreta também em um treinamento mais rápido. Além disso, a máquina também treina com mais precisão, pois, se a rede fosse treinada apenas com o conjunto total de dados, os valores dos seus pesos só seriam atualizados uma vez ao final deste treinamento, deste modo a cada subconjunto de dados que treina a rede, é feita uma atualização dos valores dos pesos para melhor se adequar ao resultado esperado. No caso do exemplo, a rede atualizou os seus parâmetros 16 vezes (15 grupos de 64 galáxias e 1 grupo de 40 galáxias). Em compensação, a rede terá menos dados para treinar a rede neural a cada rodada do que teria se treinasse com o conjunto inteiro, então haverá uma maior flutuação dos valores dos pesos até a rede acabar de treinar. A variável `STEPS_PER_EPOCH` define o número de conjuntos de amostras que serão treinadas de acordo com o valor de `BATCH_SIZE` e do número de galáxias do conjunto de treinamento, no nosso exemplo o número de iterações é dado por `STEPS_PER_EPOCH = 16`.

O termo `lr_schedule` define a taxa de decaimento do aprendizado da rede neural. Esta taxa é um valor adimensional do quanto a rede neural tem que melhorar a cada iteração. Para esta taxa são calculadas as diferenças entre o valor real e o calculado de duas camadas subsequentes. Ao treinarmos um modelo, é benéfico para a rede diminuirmos a taxa de aprendizado à medida que o treinamento avança. Este processo tem a vantagem de que no início as mudanças nos valores dos pesos da rede neural são grandes e expressivas, enquanto que a medida que a taxa de aprendizado decai, a rede neural é treinada com pequenas atualizações dos pesos após o treinamento. A vantagem disto é da máquina aprender rapidamente os valores adequados para os pesos e depois apenas refinar estes valores conforme a rede é treinada. Esta função fixa um valor inicial para a taxa de aprendizado (`initial_learning_rate = 0,0001`). A função define o passo para aplicar a taxa de decaimento, ou seja, depois de quantas iterações definidas em `STEPS_PER_EPOCH` podemos aplicar um decaimento na taxa de aprendizado. O valor deste decaimento será fornecido pela taxa de decaimento `decay_rate`. Este processo pode ser discreto ou contínuo. Em nosso trabalho, fixamos `staircase = False`, o que significa que o processo é contínuo. Temos que a função do decaimento dada pela seguinte função.

```
1 def decayed_learning_rate(step):
```

```
2   return initial_learning_rate / (1 + decay_rate * step / decay_step)
```

Onde a função `decayed_learning_rate` retorna a taxa de aprendizado para a iteração e `step` é o subconjunto de amostras.

A última linha desta parte do código fornece à rede neural o número de parâmetros dos dados de entrada da rede. No caso do `Keras`, treinamos o aprendizado de máquina com 5 parâmetros de cada galáxia: as 4 cores ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$) e a magnitude na banda `i` (m_i).

A próxima etapa do código permite definir a estrutura da rede neural do `Keras`.

```
1    x = BatchNormalization()(inputs)
2    x = Dense(30, kernel_initializer='normal',
3              kernel_constraint=max_norm(2.), activation='tanh',
   kernel_regularizer=regularizers.l1_l2(l1=1e-5, l2=1e-4),
   bias_regularizer=regularizers.l2(1e-4), activity_regularizer=
   regularizers.l2(1e-5))(x)
4      output1 = Dense(1, activation="linear", name="reg")(x)
```

A linha `x = BatchNormalization()(inputs)` do código é responsável por normalizar os dados de entrada da rede neural. A função `BatchNormalization` centraliza e redimensiona os dados de entrada de forma que a média deles esteja perto de 0 e o desvio padrão perto de 1. Este processo possibilita que a rede neural seja treinada de forma mais rápida e mais estável (com menos flutuações nos valores dos seus pesos a cada iteração) [128].

A próxima linha do código é responsável por criar a camada intermediária da rede neural. A função `Dense` é responsável por configurar esta camada do aprendizado de máquina. Para a complexidade do projeto, mostrou-se necessário construir uma rede neural com apenas uma camada intermediária. Se criássemos uma rede neural com mais camadas, a rede neural poderia sofrer com o problema do sobre-ajuste (*overfitting*) discutido na seção do modelo de rede neural (seção 2.1.3).

O primeiro argumento da função é a quantidade de nós desta camada, como discutido na seção 5.3, foram testadas outras quantidades, mas essa foi a que teve o menor erro entre o resultado calculado e o real. O argumento kernel_initializer fixa como serão definidos os valores iniciais dos pesos na camada. No nosso caso, definimos o modo como `'normal'`, o que significa que os valores dos pesos foram gerados a partir de uma distribuição normal com média em 0 e desvio padrão de 1. O argumento `kernel_constraint` permite a criação de restrições dos valores dos pesos no modelo, para o nosso caso, fizemos que `kernel_constraint=max_norm(2.)`, ou seja, definimos que os pesos têm que ter valores cuja norma é menor ou igual a 2.

O argumento `activation` define a função de ativação que irá atuar na camada intermediária. Para o nosso caso, fixamos a função de ativação da tangente hiperbólica. Foram testadas outras funções de ativação, como sigmoid, ReLu e linear, porém a função da tangente hiperbólica foi a que deu menor erro entre o redshift calculado e o real.

O termo `kernel_regularizer` aplica penalidades nos parâmetros da camada durante a otimização. Estas penalidades são somadas a função de perda calculada pela rede neural, esta função é definida na próxima parte do código. A penalidade dada pela função `l1` é calculada como $erro_{l1} = l1 * reduce\_sum(abs(\mathbf{x}))$, em que $reduce\_sumcalculaasomadoselementosdovetor\mathbf{x}el1 = 10^{-5}$. A penalidade dada pela função l2 é calculada como $erro_{l2} = l2 * reduce\_sum(square(\mathbf{x}))$ com $l2 = 10^{-4}$. Estes erros são somados a função de perda da rede à medida que a máquina é otimizada. O termo `bias_regularizer` tem a mesma função do `kernel_regularizer`, porém a penalidade é somada ao viés (*bias*) da camada, o valor da penalidade é dada pela função `l2` com o valor de $l2 = 10^{-4}$. O termo `activity_regularizer` aplica a penalidade no dado de saída da camada, o valor da penalidade é dada pela função `l2`.

Por fim, o código define a estrutura do dado de saída da rede neural. Como queremos o redshift fotométrico da galáxia, o dado de saída fornecido a rede no treinamneto foram os redshifts espectroscópicos das galáxias do `VIPERS`. Então, o dado de saída (*output*) do aprendizado de máquina é um valor unidimensional. A função de ativação desta camada é linear.

Com a estrutura da rede neural definida, o próximo passo foi treinar o aprendizado de máquina.

```
1     model_ann = keras.Model(inputs=inputs, outputs=[output1])
2     model_ann.compile(loss={'reg': 'mean_absolute_error'}, loss_weights
    =[0.1],
3                       optimizer=tf.keras.optimizers.Adam(lr_schedule),
    metrics={'reg': "mse})
4
5     history = model_ann.fit(X[:, :5], {'reg': y_total[:, 200]},
    validation_split=0.2)
6
7     model_ann.save(outpath + '/keras_model_' + str(i+1) + ".h5")
```

A primeira linha desta etapa do código é responsável por definir os dados de entrada e os dados de saída da rede neural, além de definir o nome da rede, que fixamos como `model_ann`. A linha seguinte define a métrica utilizada na função de perda da rede neural. Para esta rede neural foi definido que a métrica que calcularia o erro entre resultado calculado e o real seria o erro quadrático médio - equação (61).

Após isso, a rede neural é treinada através da função `model_ann.fit` cujos argumentos são os dados de entrada, os dados de saída e a porcentagem do conjunto de dados que será separado para o grupo de validação. Em nosso programa fixamos que 20% das galáxias seriam separadas para a validação de forma a otimizar os pesos da rede neural. Por fim, cada um dos 10 modelos da rede neural foi salvo para ser utilizado nas galáxias do `DES`.

A última fase do programa consiste em calcular o redshift fotométrico para as galáxias do `DES`.

```
1 models = load_models(number_sim, models_folder)
2 feat = [mag_G, mag_R, mag_I, mag_Z, mag_Y]
```

```
filename = os.listdir(input_folder)

for i in range(len(filename)):
    print(os.path.join(input_folder, filename[i]))
    data_aux = Table.read(os.path.join(input_folder, filename[i])).
   to_pandas()
    data = remove_outliers(data_aux, feat)
    dataset = np.zeros(shape=(len(data), 5))
    dataset[:, 0] = data[feat[0]].values - data[feat[1]].values
    dataset[:, 1] = data[feat[1]].values - data[feat[2]].values
    dataset[:, 2] = data[feat[2]].values - data[feat[3]].values
    dataset[:, 3] = data[feat[3]].values - data[feat[4]].values
    dataset[:, 4] = data[feat[4]].values
    dataset_output = pd.DataFrame()
    dataset_output["COADD_OBJECT_ID"] = data["COADD_OBJECT_ID"]

    mu_keras = []
    for j in range(number_sim):
        predictions = models[j].predict(dataset)
        zphot = predictions[0].flatten()
        dataset_output["keras_zphot" + str(j+1)] = zphot
        mu_keras.append(zphot)
        mean_keras = pd.Series(np.round(sum(np.array(mu_keras)/10),6),
   name="mean_keras")
         standard_dev_keras = pd.Series(np.round(np.std(np.array(
   mu_keras), axis=0),6), name="StanDev_keras")

        dataset_output["Mean_keras"] = mean_keras
        dataset_output["StanDev_keras"] = standard_dev_keras
        pixel = filename[i][len(prefix):len(prefix) + 5]
        nome = prefix + pixel + "_KERAS" + sufix
        print(outpath + nome)
        dataset_output.to_csv(outpath + str(nome),index = False)
```

As primeiras três linhas do código são responsáveis por preparar o programa antes de receber as informações fotométricas das galáxias. Os modelos de redes neurais salvos anteriormente são coletados e são reunidos os nomes de todos os pixels das galáxias do `DES` disponíveis. O código então baixa um catálogo de galáxias por pixel do `DES` em `filename = os.listdir(input_folder)`.

As galáxias são tratadas, tendo o valor das magnitudes com valores atípicos (igual a 99) substituídos pelos valores médias das magnitudes no mesmo filtro. É, então, criado o conjunto de entrada das galáxias do `DES` na rede neural (`dataset`). O redshift fotométrico é calculado para os 10 modelos de rede neural do textttKeras. É criada uma matriz contendo todos os redshifts fotométricos para cada galáxia nos 10 aprendizados de máquina do `Keras`. A partir desta matriz são calculadas a média (`mean_keras`) e o desvio padrão (`StanDev_keras`) dos redshifts para cada galáxia.

Por fim, o programa cria um catálogo para o pixel em questão com o identificador das galáxias (`COADD_OBJECT_ID`), os redshifts fotométricos calculados por cada máquina e a média e o desvio padrão desta propriedade (z_{phot}). Este catálogo é salvo com o nome do pixel, um exemplo deste catálogo pode ser visto na tabela 10.

COADD_OBJECT_ID	GPz_color_zphot1	GPz_color_sigma1	GPz_color_zphot2	GPz_color_sigma2	...	GPz_color_zphot10	GPz_color_sigma10	Mean_GPz_color	StanDev_GPz_color	Mean_GPz_sigma_color	StanDev_GPz_sigma_color
1481845840	1.000008	0.109983	1.054000	0.058160	...	1.002807	0.085125	1.007099	0.064808	0.096906	0.036061
1481845849	0.702038	0.003304	0.689646	0.005813	...	0.676350	0.004332	0.686519	0.012741	0.004790	0.001576
1481847255	0.738434	0.001391	0.728444	0.001547	...	0.732663	0.001685	0.730226	0.006282	0.001600	0.000590
1481845888	1.056070	0.022679	0.920046	0.121856	...	1.069974	0.227725	1.014313	0.066359	0.107888	0.065416
1481845957	0.551975	0.001239	0.549767	0.001279	...	0.550334	0.001855	0.548854	0.001818	0.001579	0.000227
1481846051	0.967023	0.058550	0.976763	0.021516	...	0.918178	0.087582	0.949061	0.043900	0.068123	0.029888
1481846090	0.715073	0.008761	0.712433	0.009310	...	0.708001	0.012521	0.709623	0.007345	0.010359	0.001349
1481846095	0.615094	0.005561	0.611272	0.007001	...	0.621301	0.006508	0.618466	0.006046	0.007248	0.001361
1481846152	1.054459	0.021210	1.036411	0.027895	...	1.040956	0.013590	1.048114	0.017231	0.019231	0.003715
1481846184	0.929447	0.025589	0.890662	0.059883	...	1.095382	0.455159	0.963442	0.087126	0.165331	0.141897
1481846212	0.573407	0.084538	0.559548	0.072725	...	0.605507	0.076461	0.588324	0.018522	0.080423	0.018662
1481846280	0.513034	0.006670	0.508844	0.005406	...	0.521900	0.005991	0.513848	0.005019	0.005433	0.000798
1481846309	0.734022	0.028531	0.754679	0.009509	...	0.726127	0.016033	0.718254	0.026599	0.019277	0.008383
1481846317	0.642379	0.020927	0.612106	0.010231	...	0.620780	0.007268	0.633258	0.014117	0.012697	0.005102
1481846390	0.548283	0.006775	0.553151	0.008478	...	0.556143	0.006241	0.553903	0.006679	0.006961	0.001105
1481846531	0.868046	0.061505	0.895201	0.043581	...	0.995719	0.093729	0.919918	0.065443	0.064397	0.027018
1481846544	0.893148	0.059958	0.832458	0.022836	...	0.786762	0.071220	0.792664	0.087478	0.040564	0.025893
1481846560	0.897612	0.086848	0.760380	0.343881	...	0.812237	0.080362	0.733874	0.137870	0.365489	0.496235
1481846631	0.629824	0.003230	0.642668	0.003173	...	0.637466	0.003447	0.638252	0.003732	0.003522	0.000424
1481846696	0.573027	0.003077	0.586210	0.003211	...	0.586456	0.003094	0.577249	0.007751	0.003814	0.000647
1481846701	0.802344	0.093624	0.797679	0.016830	...	0.832816	0.016077	0.846828	0.068707	0.057979	0.075419
1481846705	0.660898	0.001815	0.669895	0.002987	...	0.663947	0.002994	0.661713	0.005616	0.002614	0.000557
1481846757	0.744897	0.011050	0.759955	0.012569	...	0.766405	0.013889	0.762951	0.008094	0.011202	0.001913
1481846761	0.844925	0.005097	0.813869	0.007488	...	0.810744	0.010142	0.830990	0.012560	0.008260	0.002972
1481846775	0.752398	0.003503	0.754743	0.003523	...	0.757360	0.004275	0.753719	0.004873	0.003625	0.000451
1481846844	0.587316	0.019426	0.584212	0.016720	...	0.589046	0.010966	0.588917	0.013901	0.015431	0.003115
1481846850	0.801750	0.037353	0.795234	0.014312	...	0.792355	0.031109	0.787612	0.055732	0.068668	0.062801
1481846879	0.542285	0.001436	0.543659	0.001841	...	0.545931	0.001550	0.543076	0.001931	0.001622	0.000150
1481846886	0.629608	0.013493	0.609854	0.012482	...	0.646098	0.012363	0.630119	0.011099	0.010100	0.002538
1481846906	1.042299	0.005920	1.030664	0.009678	...	1.054749	0.008697	1.042478	0.019045	0.010777	0.005269
1493197050	0.715170	0.021318	0.626685	0.030727	...	0.700093	0.018083	0.693982	0.031600	0.017485	0.008133
1493198323	0.684425	0.072303	0.742532	0.087087	...	0.670842	0.036822	0.762221	0.069670	0.068132	0.021050
1493198388	0.767901	0.010048	0.744127	0.007684	...	0.773354	0.007635	0.765800	0.011925	0.007889	0.001142
1493199098	0.778675	0.069232	0.946266	0.070733	...	0.923924	0.027794	0.900154	0.061484	0.049153	0.021841
1493199308	0.644849	0.008147	0.494971	0.016307	...	0.777978	0.047625	0.683251	0.100415	0.027142	0.018038
1493197020	0.496780	0.003343	0.499855	0.003277	...	0.497316	0.004585	0.498169	0.002400	0.003848	0.000533
1493197032	1.105836	0.052221	0.872659	0.015355	...	0.982411	0.175703	0.993121	0.141977	0.108633	0.096486
1493197061	0.791740	0.030380	0.950554	0.029280	...	0.860606	0.018953	0.898497	0.050713	0.031520	0.021059
1493197100	0.459167	0.020590	0.479507	0.015174	...	0.486360	0.022826	0.484754	0.020758	0.019843	0.009160
1493197136	1.030266	0.179140	0.942248	0.090572	...	0.936734	0.053279	0.903874	0.099018	0.110081	0.097039
1493197408	0.967886	0.017845	1.000125	0.038406	...	0.978436	0.030195	0.976649	0.020386	0.027173	0.005760
1493197432	0.858779	0.012599	0.876123	0.009784	...	0.868899	0.010688	0.878306	0.014538	0.010345	0.002945
1493197529	0.592718	0.053565	0.535738	0.028018	...	0.743578	0.181481	0.590857	0.082832	0.067611	0.046412
1493197715	0.707454	0.006770	0.703437	0.006616	...	0.727726	0.005521	0.711432	0.010943	0.006511	0.001281
1493197839	0.553589	0.015480	0.568295	0.018405	...	0.551612	0.020798	0.556886	0.007391	0.018937	0.004226
1493197955	0.513262	0.085915	0.679581	0.088697	...	0.602001	0.033359	0.671017	0.093139	0.059124	0.038021
1493198006	0.990299	0.016717	1.067721	0.010016	...	1.022980	0.005248	1.033708	0.064284	0.013381	0.007942
1493198007	0.723497	0.070977	0.450998	0.034854	...	0.917402	0.081448	0.805067	0.150032	0.154158	0.169668
1493198059	0.423118	0.018115	0.452345	0.031803	...	0.434178	0.020442	0.442785	0.015102	0.023457	0.006235
1493198268	0.907129	0.019961	0.689593	0.046128	...	1.033491	0.446490	0.900798	0.118264	0.283702	0.324391

Tabela 7: Catálogo das galáxias do `DES` com o redshift fotométrico calculado pelo `GPz` com as 4 cores e a magnitude no filtro `i` ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$ e m_i).

COADD_OBJECT_ID	GPz_mag_zphot1	GPz_mag_sigma1	GPz_mag_zphot2	GPz_mag_sigma2	...	GPz_mag_zphot10	GPz_mag_sigma10	Mean_GPz_mag	StanDev_GPz_mag	Mean_GPz_sigma_mag	StanDev_GPz_sigma_mag
1481845840	1.031537	0.067241	1.049635	0.050572	...	1.078258	0.053651	1.038055	0.039683	0.052083	0.012061
1481845849	0.695135	0.005307	0.659898	0.004519	...	0.697433	0.002557	0.693474	0.013136	0.003307	0.001264
1481847255	0.723346	0.001523	0.721806	0.001550	...	0.736147	0.001488	0.728148	0.007635	0.001703	0.000583
1481845888	0.910593	0.057820	0.905599	0.047981	...	0.986785	0.073689	0.980104	0.092686	0.056834	0.012368
1481845957	0.557284	0.000529	0.556532	0.000924	...	0.556845	0.001176	0.554358	0.002230	0.000959	0.000197
1481846051	0.860249	0.056318	0.918381	0.042681	...	0.822811	0.060169	0.861282	0.045804	0.056708	0.014616
1481846090	0.715616	0.006582	0.709832	0.011259	...	0.713497	0.010738	0.710581	0.007145	0.009951	0.002865
1481846095	0.619379	0.007754	0.618514	0.008324	...	0.610958	0.004990	0.612123	0.008561	0.008156	0.001347
1481846152	1.060483	0.015560	1.093741	0.020488	...	1.044338	0.014756	1.063436	0.013249	0.016832	0.004855
1481846184	1.073254	0.052418	0.949557	0.059391	...	1.026801	0.072609	0.993968	0.041458	0.049706	0.010762
1481846212	0.561676	0.070754	0.527403	0.074861	...	0.584342	0.068705	0.556643	0.021313	0.098939	0.025947
1481846280	0.517342	0.003197	0.515010	0.004786	...	0.513216	0.005174	0.517848	0.003577	0.003567	0.000815
1481846309	0.767123	0.016163	0.723078	0.018440	...	0.736872	0.013543	0.748351	0.020788	0.015862	0.003273
1481846317	0.661402	0.006776	0.626879	0.013622	...	0.625961	0.012668	0.630170	0.015019	0.009508	0.002826
1481846390	0.567238	0.006899	0.558475	0.015200	...	0.533469	0.018703	0.557174	0.009692	0.012883	0.004730
1481846531	1.027020	0.049610	0.839475	0.058059	...	0.969423	0.077141	0.942719	0.055189	0.049739	0.012890
1481846544	0.749951	0.046665	0.738020	0.045661	...	0.680497	0.047490	0.759799	0.037913	0.044973	0.007780
1481846560	0.802079	0.052864	0.724171	0.047665	...	0.698715	0.076635	0.784047	0.058596	0.055535	0.012085
1481846631	0.640310	0.004516	0.630039	0.002966	...	0.638689	0.004384	0.635437	0.004027	0.003741	0.000731
1481846696	0.585684	0.005431	0.582977	0.004418	...	0.564456	0.004796	0.577734	0.007592	0.004852	0.001298
1481846701	0.782188	0.049745	0.820428	0.043198	...	0.777617	0.054528	0.809426	0.031469	0.044572	0.006126
1481846705	0.666733	0.002997	0.672572	0.003001	...	0.666459	0.002684	0.668885	0.005470	0.003299	0.001239
1481846757	0.753588	0.010401	0.772226	0.016637	...	0.754428	0.012241	0.759824	0.009522	0.014418	0.003019
1481846761	0.847331	0.014934	0.842964	0.012188	...	0.824842	0.014060	0.826704	0.011533	0.009830	0.002865
1481846775	0.772525	0.004080	0.758041	0.003826	...	0.755436	0.003625	0.757653	0.007731	0.004356	0.000449
1481846844	0.595846	0.010885	0.596623	0.015923	...	0.593540	0.010721	0.592919	0.008347	0.012759	0.002447
1481846850	0.860538	0.041894	0.803456	0.033224	...	0.786867	0.044702	0.809192	0.035219	0.045802	0.005380
1481846879	0.538372	0.001418	0.537598	0.001848	...	0.542466	0.001662	0.540486	0.002345	0.001650	0.000255
1481846886	0.627553	0.005317	0.639638	0.014749	...	0.634761	0.009632	0.639992	0.008513	0.009685	0.003284
1481846906	1.038375	0.012428	1.006225	0.024696	...	1.072598	0.009867	1.028146	0.017630	0.011154	0.005125
1493197050	0.752716	0.021105	0.741249	0.024072	...	0.733724	0.018979	0.736246	0.019004	0.017198	0.005859
1493198323	0.757501	0.097946	0.789728	0.061902	...	0.798080	0.073328	0.797947	0.046997	0.079084	0.036750
1493198388	0.818776	0.015503	0.791097	0.014126	...	0.772746	0.013829	0.791626	0.013684	0.015422	0.003778
1493199098	0.907146	0.037455	0.876711	0.033333	...	0.876753	0.050029	0.892438	0.037991	0.041753	0.006156
1493199308	0.568395	0.053704	0.539956	0.047632	...	0.498477	0.070209	0.532197	0.058666	0.054794	0.008363
1493197020	0.499905	0.003368	0.504775	0.004617	...	0.504416	0.003971	0.503677	0.004272	0.003734	0.000640
1493197032	1.038769	0.056059	0.838145	0.047292	...	0.965939	0.078278	0.971897	0.091055	0.062294	0.013112
1493197061	0.839392	0.028736	0.871579	0.034463	...	0.835005	0.032306	0.871639	0.025473	0.036178	0.009045
1493197100	0.494180	0.015441	0.502359	0.020085	...	0.537233	0.023317	0.501474	0.016553	0.021045	0.005112
1493197136	0.870666	0.051529	0.826489	0.046047	...	0.792718	0.072702	0.859646	0.043163	0.049832	0.009387
1493197408	0.931986	0.013994	0.919764	0.023921	...	0.947669	0.017914	0.947289	0.017031	0.021762	0.004146
1493197432	0.890082	0.011303	0.909365	0.010468	...	0.871316	0.014998	0.881616	0.012997	0.010139	0.002472
1493197529	0.666530	0.035315	0.680953	0.047495	...	0.752678	0.038008	0.672949	0.043664	0.040769	0.007624
1493197715	0.730131	0.005135	0.710413	0.009345	...	0.712622	0.006598	0.717696	0.011224	0.008717	0.001777
1493197839	0.505062	0.016519	0.485315	0.012538	...	0.534409	0.023290	0.513189	0.016324	0.018604	0.004603
1493197955	0.656586	0.071367	0.592760	0.069053	...	0.734049	0.065213	0.674471	0.069496	0.056780	0.013983
1493198006	1.002333	0.049668	0.977039	0.032630	...	0.960767	0.053773	0.996953	0.034470	0.038685	0.012675
1493198007	0.773184	0.057989	0.653946	0.047390	...	0.661531	0.077089	0.707482	0.062978	0.055686	0.009076
1493198059	0.430270	0.026407	0.384404	0.019905	...	0.408132	0.016659	0.410702	0.030012	0.020767	0.006984
1493198268	1.214172	0.052832	1.027828	0.049627	...	1.123950	0.077373	1.118016	0.067917	0.054663	0.009210

Tabela 8: Catálogo das galáxias do DES com o redshift fotométrico calculado pelo GPz com as 5 magnitudes (m_g, m_r, m_i, m_z e m_i).

F:ANNZ_best	F:ANNZ_best_err	F:ANNZ_MLM_avg_0	F:ANNZ_MLM_avg_0_err	F:ANNZ_PDF_avg_0	F:ANNZ_PDF_avg_0_err	F:ANNZ_PDF_0_0	F:ANNZ_PDF_0_1	F:ANNZ_PDF_0_2	...	F:ANNZ_PDF_0_198	F:ANNZ_PDF_0_199
1.030773	0.127973	0.979941	0.134664	0.943044	0.168771	0.002635	0.000000	0.000000	...	0.0	0.0
0.696192	0.073017	0.702817	0.065372	0.707452	0.100942	0.002092	0.000000	0.000000	...	0.0	0.0
0.746890	0.052786	0.746141	0.054413	0.736470	0.097106	0.002689	0.000000	0.000000	...	0.0	0.0
1.247249	0.154589	0.933940	0.155554	0.893495	0.203663	0.001615	0.000000	0.000000	...	0.0	0.0
0.555462	0.038274	0.543813	0.038939	0.548178	0.075765	0.003718	0.000000	0.000000	...	0.0	0.0
0.874647	0.141796	1.031317	0.146309	0.991197	0.173559	0.000743	0.000000	0.000000	...	0.0	0.0
0.722997	0.046810	0.712411	0.046063	0.705047	0.093585	0.002830	0.000000	0.000000	...	0.0	0.0
0.630130	0.083326	0.627590	0.082271	0.633703	0.103262	0.000616	0.000000	0.000000	...	0.0	0.0
1.023249	0.087682	1.017268	0.092907	0.985303	0.131374	0.002383	0.000000	0.000000	...	0.0	0.0
1.083539	0.147954	0.980695	0.150985	0.940133	0.227439	0.001588	0.000000	0.000000	...	0.0	0.0
0.513690	0.070621	0.708253	0.071376	0.723213	0.157379	0.002628	0.000000	0.000000	...	0.0	0.0
0.502162	0.053723	0.499751	0.052672	0.519100	0.074837	0.001658	0.000000	0.000000	...	0.0	0.0
0.694413	0.081683	0.722338	0.089192	0.729206	0.126055	0.002649	0.000000	0.000000	...	0.0	0.0
0.639696	0.107664	0.647809	0.105050	0.648033	0.126626	0.001812	0.000000	0.000000	...	0.0	0.0
0.577240	0.070730	0.574344	0.070116	0.582510	0.092787	0.003200	0.000000	0.000000	...	0.0	0.0
0.974010	0.113143	0.989926	0.117654	0.956872	0.166763	0.001971	0.000000	0.000000	...	0.0	0.0
0.521809	0.117796	0.715545	0.131775	0.716044	0.149483	0.001884	0.000000	0.000000	...	0.0	0.0
0.659982	0.107056	0.783175	0.127795	0.780451	0.156896	0.002444	0.000000	0.000000	...	0.0	0.0
0.654730	0.048853	0.645819	0.053288	0.643870	0.090512	0.002960	0.000000	0.000000	...	0.0	0.0
0.563780	0.024741	0.538542	0.035616	0.546797	0.077796	0.004913	0.000000	0.000000	...	0.0	0.0
0.746937	0.107056	0.795715	0.127845	0.791613	0.162495	0.002748	0.000000	0.000000	...	0.0	0.0
0.658226	0.063002	0.668882	0.060539	0.674243	0.095056	0.003034	0.000000	0.000000	...	0.0	0.0
0.795604	0.105966	0.796735	0.099503	0.785491	0.124551	0.001979	0.000000	0.000000	...	0.0	0.0
0.831698	0.078304	0.816340	0.077041	0.806451	0.111351	0.001881	0.000000	0.000000	...	0.0	0.0
0.751673	0.059854	0.749453	0.056873	0.746199	0.095106	0.001728	0.000000	0.000000	...	0.0	0.0
0.574147	0.092885	0.576925	0.094201	0.565635	0.108306	0.003463	0.000000	0.000000	...	0.0	0.0
0.724398	0.133562	0.686876	0.147431	0.679650	0.164649	0.003494	0.000000	0.000000	...	0.0	0.0
0.559381	0.045132	0.544418	0.044533	0.544444	0.079695	0.004728	0.000000	0.000000	...	0.0	0.0
0.636321	0.067480	0.625135	0.069975	0.627235	0.096548	0.001952	0.000000	0.000000	...	0.0	0.0
0.990895	0.067925	0.956073	0.070397	0.945490	0.133434	0.001687	0.000000	0.000000	...	0.0	0.0
0.713794	0.099619	0.732835	0.101939	0.735341	0.132868	0.002569	0.000000	0.000000	...	0.0	0.0
0.709015	0.154198	0.726204	0.154526	0.716965	0.173824	0.002890	0.000000	0.000000	...	0.0	0.0
0.775421	0.065824	0.775150	0.070337	0.764056	0.108274	0.002545	0.000000	0.000000	...	0.0	0.0
0.928532	0.128074	0.933522	0.148345	0.901408	0.179160	0.002161	0.000000	0.000000	...	0.0	0.0
0.900789	0.118784	0.692554	0.133817	0.688806	0.174688	0.002171	0.000000	0.000000	...	0.0	0.0
0.505235	0.053448	0.500486	0.056863	0.525329	0.080552	0.003029	0.000000	0.000000	...	0.0	0.0
0.932107	0.144664	0.967777	0.149102	0.921225	0.212480	0.002042	0.000000	0.000000	...	0.0	0.0
0.939851	0.154197	0.896408	0.141904	0.871665	0.166655	0.001915	0.000000	0.000000	...	0.0	0.0
0.476285	0.067523	0.467609	0.066790	0.495501	0.097653	0.004748	0.000000	0.000000	...	0.0	0.0
0.705475	0.108130	0.788780	0.128863	0.781168	0.147952	0.001525	0.000000	0.000000	...	0.0	0.0
0.966361	0.117734	0.934856	0.119933	0.906312	0.156488	0.002294	0.000000	0.000000	...	0.0	0.0
0.911358	0.097621	0.865183	0.100682	0.856950	0.126978	0.000838	0.000000	0.000000	...	0.0	0.0
0.490514	0.098316	0.540701	0.112831	0.547399	0.134247	0.005107	0.000000	0.000000	...	0.0	0.0
0.725128	0.056981	0.717206	0.057896	0.714293	0.097669	0.002872	0.000000	0.000000	...	0.0	0.0
0.501117	0.088818	0.491597	0.088101	0.531429	0.107532	0.004788	0.000000	0.000000	...	0.0	0.0
0.767442	0.155914	0.652834	0.149711	0.660355	0.179821	0.003599	0.000209	0.000000	...	0.0	0.0
0.835596	0.080282	0.849369	0.075754	0.849517	0.117183	0.001553	0.000000	0.000000	...	0.0	0.0
0.689889	0.150936	0.690017	0.144697	0.693616	0.162321	0.002346	0.000000	0.000000	...	0.0	0.0
0.453831	0.069986	0.500311	0.070796	0.517990	0.093570	0.003598	0.000000	0.000000	...	0.0	0.0
1.100563	0.150026	1.005987	0.156090	0.954883	0.248688	0.001564	0.000000	0.000000	...	0.0	0.0

Tabela 9: Catálogo das galáxias do `DES` com o redshift fotométrico calculado pelo `ANNz` 4 cores e a magnitude no filtro `i` ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$ e m_i).

COADD_OBJECT_ID	keras_zphot1	keras_zphot2	...	keras_zphot10	Mean_keras	StanDev_keras
1481845840	1.077528	0.873640	...	0.914025	0.947452	0.075789
1481845849	0.645486	0.663927	...	0.698160	0.668658	0.024032
1481847255	0.702396	0.734095	...	0.719349	0.722418	0.018669
1481845888	0.748147	0.939306	...	0.601511	0.768047	0.130237
1481845957	0.548814	0.546945	...	0.561623	0.545487	0.008251
1481846051	0.869604	0.871196	...	0.840843	0.866933	0.053588
1481846090	0.702436	0.699559	...	0.729304	0.711629	0.014377
1481846095	0.619330	0.671760	...	0.620129	0.633359	0.015577
1481846152	1.026081	0.977452	...	1.009483	1.001173	0.021995
1481846184	1.176884	0.942777	...	0.981937	1.025823	0.100981
1481846212	0.579324	0.562922	...	0.656608	0.611321	0.032753
1481846280	0.528575	0.513830	...	0.488301	0.515332	0.015971
1481846309	0.681441	0.743554	...	0.699377	0.698376	0.043691
1481846317	0.592012	0.598816	...	0.545360	0.593973	0.028277
1481846390	0.559507	0.563902	...	0.589866	0.566292	0.011765
1481846531	1.071181	0.937853	...	0.940382	0.978912	0.070020
1481846544	0.736908	0.712470	...	0.420027	0.660617	0.142930
1481846560	0.702781	0.885254	...	0.637521	0.621727	0.123108
1481846631	0.661122	0.658315	...	0.666256	0.654463	0.010102
1481846696	0.551658	0.510224	...	0.588016	0.568948	0.028129
1481846701	0.706314	1.003100	...	0.420710	0.688360	0.167163
1481846705	0.664145	0.697166	...	0.673423	0.672057	0.013381
1481846757	0.757278	0.785528	...	0.768012	0.759150	0.017347
1481846761	0.821611	0.817743	...	0.808532	0.819013	0.022154
1481846775	0.774366	0.769235	...	0.743275	0.758943	0.014410
1481846844	0.586761	0.540114	...	0.603197	0.587998	0.019806
1481846850	0.675139	0.834676	...	0.571947	0.692289	0.065093
1481846879	0.544199	0.577262	...	0.560591	0.558378	0.011217
1481846886	0.609977	0.643549	...	0.644616	0.638632	0.020867
1481846906	0.959053	0.923503	...	0.945172	0.961130	0.026463
1493197050	0.717502	0.681264	...	0.652332	0.693883	0.025411
1493198323	0.620462	0.736545	...	0.622113	0.645980	0.041139
1493198388	0.792443	0.804857	...	0.694636	0.769630	0.040247
1493199098	0.959221	0.913603	...	0.871078	0.943387	0.053295
1493199308	0.560227	0.536713	...	0.607557	0.539343	0.091549
1493197020	0.493828	0.507601	...	0.517974	0.511622	0.010296
1493197032	0.941456	0.863247	...	0.845619	0.876763	0.082054
1493197061	0.903011	0.839407	...	0.911603	0.871043	0.054065
1493197100	0.458566	0.428003	...	0.531810	0.462236	0.033408
1493197136	0.761985	0.997167	...	0.387923	0.708204	0.195234
1493197408	0.999045	0.858836	...	0.958377	0.928203	0.052519
1493197432	0.849436	0.916036	...	0.877303	0.881513	0.033470
1493197529	0.591176	0.554469	...	0.617518	0.528249	0.049516
1493197715	0.774009	0.723736	...	0.730918	0.739011	0.024042
1493197839	0.553567	0.528248	...	0.577365	0.542836	0.018977
1493197955	0.527676	0.540824	...	0.539260	0.541530	0.036439
1493198006	0.995957	0.991961	...	0.852961	0.973439	0.071709
1493198007	0.670712	0.685005	...	0.751172	0.657951	0.070718
1493198059	0.530297	0.397710	...	0.469774	0.485120	0.036160
1493198268	1.245091	0.995440	...	0.983900	1.080294	0.125442

Tabela 10: Catálogo das galáxias do `DES` com o redshift fotométrico calculado pelo `keras` com as 4 cores e a magnitude no filtro `i` ($m_g - m_r$, $m_r - m_i$, $m_i - m_z$, $m_z - m_Y$ e m_i).

Referências

[1] Elcio Abdalla et al. «The BINGO project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A14. DOI: 10.1051/0004-6361/202140883. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202140883.

[2] Carlos A. Wuensche et al. «The BINGO project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A15. DOI: 10.1051/0004-6361/202039962. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202039962.

[3] Filipe B. Abdalla et al. «The BINGO Project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A16. DOI: 10.1051/0004-6361/202141382. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202141382.

[4] Vincenzo Liccardo et al. «The BINGO project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A17. DOI: 10.1051/0004-6361/202140886. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202140886.

[5] Karin S. F. Fornazier et al. «The BINGO project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A18. DOI: 10.1051/0004-6361/202141707. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202141707.

[6] Jiajun Zhang et al. «The BINGO project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A19. DOI: 10.1051/0004-6361/202140887. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202140887.

[7] Andre A. Costa et al. «The BINGO project». Em: *Astronomy &amp Astrophysics* 664 (ago. de 2022), A20. DOI: 10.1051/0004-6361/202140888. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F202140888.

[8] Elcio Abdalla e Alessandro Marins. «The dark sector cosmology». Em: *International Journal of Modern Physics D* 29.14 (out. de 2020), p. 2030014. DOI: 10.1142/s0218271820300141. URL: https://doi.org/10.1142%5C%2Fs0218271820300141.

[9] DES e NOAO Data Lab Collaborations. «The Dark Energy Survey Data release 1». Em: *arXiv:1801.03181* (2018).

[10] SDSS-III Collaboration. *Sloan Digital Sky Survey (SDSS) - DR10*. 2014. URL: http://skyserver.sdss.org/dr10/en/tools/toolshome.aspx.

[11] Christopher P. Ahn et al. «The Tenth Data Release of the Sloan Digital Sky Survey: First Spectroscopic Data from the SDSS-III Apache Point Observatory Galactic Evolution Experiment». Em: *The Astrophysical Journal Supplement Series* 211.2 (mar. de 2014), p. 17. DOI: 10.1088/0067-0049/211/2/17. URL: https://doi.org/10.1088%5C%2F0067-0049%5C%2F211%5C%2F2%5C%2F17.

[12] DES e NOAO Data Lab Collaborations. «The Dark Energy Survey Data release 2». Em: *arXiv:2101.05765* (2021).

[13] F. G. Mohammad, D. Bianchi e et al. «The VIMOS Public Extragalactic Redshift Survey (VIPERS): Unbiased clustering estimate with VIPERS slit assignment». Em: *arXiv:1807.05999* (2018).

[14] Ibrahim A. Almosallam, Matt J. Jarvis e Stephen J. Roberts. «GPz: non-stationary sparse Gaussian processes for heteroscedastic uncertainty estimation in photometric redshifts». Em: *Monthly Notices of the Royal Astronomical Society* 462.1 (jul. de 2016), pp. 726–739. DOI: 10.1093/mnras/stw1618. URL: https://doi.org/10.1093%2Fmnras%2Fstw1618.

[15] Abdalla F. B. Sadeh I. e Lahav O. «ANNz2 - photometric redshift and probability distribution function estimation using machine learning». Em: *arXiv:1507.00490* (2016).

[16] KERAS. *Keras*. URL: https://keras.io/. (accessed: 15.07.2022).

[17] Simon Haykin. *Redes Neurais: Princípios e Prática*. 2000, p. 898. ISBN: 978-8573-0771-86.

[18] K. R. Chowdhary. *Natural Language Processing*. Springer India, 2020, pp. 603–649. ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19. URL: https://doi.org/10.1007/978-81-322-3972-7_19.

[19] Sourav Saha et al. «Hierarchical Deep Learning Neural Network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering». Em: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113452. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113452. URL: https://www.sciencedirect.com/science/article/pii/S004578252030637X.

[20] David C. Parkes e Michael P. Wellman. «Economic reasoning and artificial intelligence». Em: *Science* 349.6245 (2015), pp. 267–272. DOI: 10.1126/science.aaa8403. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8403. URL: https://www.science.org/doi/abs/10.1126/science.aaa8403.

[21] Malik Ghallab, Dana Nau e Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. ISBN: 9781107037274. URL: https://projects.laas.fr/planning/book.pdf.

[22] Bee Wah Yap Michael W. Berry Azlinah Mohamed. *Supervised and Unsupervised Learning for Data Science*. Springer Cham, 2020. ISBN: 978-3-030-22475-2. DOI: https://doi.org/10.1007/978-3-030-22475-2.

[23] Mohamed Alloghani et al. «A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science». Em: *Supervised and Unsupervised Learning for Data Science*. Ed. por Michael W. Berry, Azlinah Mohamed e Bee Wah Yap. Springer International Publishing, 2020, pp. 3–21. ISBN: 978-3-030-

22475-2. DOI: 10.1007/978-3-030-22475-2_1. URL: https://doi.org/10.1007/978-3-030-22475-2_1.

[24] Marco Wiering e Martijn Otterlo. *Reinforcement Learning*. Springer Berlin, Heidelberg, 2012. ISBN: 978-3-642-44685-6. DOI: https://doi.org/10.1007/978-3-642-27645-3.

[25] Russell D. Reed e II Robert J. Marks. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. The MIT Press, 1999, p. 346. ISBN: 0-262-18190-8.

[26] Marcus Brandão. «As bases biológicas do comportamento: introdução à neurociência». Em: *Revista Do Instituto De Medicina Tropical De Sao Paulo - REV INST MED TROP SAO PAULO* 47 (jun. de 2005). DOI: 10.1590/S0036-46652005000300013.

[27] Dee Unglaub Silverthorn. *Fisiologia Humana: Uma Abordagem Integrada*. 7ª edição. Artmed, 2017. ISBN: 978-8582714034.

[28] Dale A. Mackall et al. *Verification and Validation of Neural Networks for Aerospace Systems*. 2002, p. 85.

[29] Jeremy Adcock et al. «Advances in quantum machine learning». Em: (2015). URL: https://arxiv.org/abs/1512.02900.

[30] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. MIT Press, 2016.

[31] Johnson Kolluri et al. «Reducing Overfitting Problem in Machine Learning Using Novel L1/4 Regularization Method». Em: (jun. de 2020), pp. 934–938. DOI: 10.1109/ICOEI48184.2020.9142992.

[32] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. The MIT Press, 2012, p. 1067. ISBN: 978-0-262-01802-9.

[33] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, 2006, p. 738. ISBN: : 978-0387-31073-2.

[34] Mirjam Pot, Nathalie Kieusseyan e Barbara Prainsack. «Not all biases are bad: equitable and inequitable biases in machine learning and radiology». Em: *Insights into Imaging* 12 (fev de 2021), p. 13. DOI: 10.1186/s13244-020-00955-7.

[35] Ninareh Mehrabi et al. «A Survey on Bias and Fairness in Machine Learning». Em: 54.6 (jul. de 2021). ISSN: 0360-0300. DOI: 10.1145/3457607. URL: https://doi.org/10.1145/3457607.

[36] Pragati Baheti. *Activation Functions in Neural Networks [12 Types Use Cases]*. URL: https://www.v7labs.com/blog/neural-networks-activation-functions. (accessed: 16.07.2022).

[37] Patrick Haffner. «Fast Transpose Methods for Kernel Learning on Sparse Data». Em: ICML '06 (2006), pp. 385–392. DOI: 10.1145/1143844.1143893. URL: https://doi.org/10.1145/1143844.1143893.

[38] Trevor Hastie, Robert Tibshirani e Jerome Friedman. *The Elements of Statistical Learning*. Second Edition. Springer Science+Business Media, LLC, 2001. ISBN: 978-0387848570.

[39] Katarzyna Janocha e Wojciech Marian Czarnecki. *On Loss Functions for Deep Neural Networks in Classification*. 2017. DOI: 10.48550/ARXIV.1702.05659. URL: https://arxiv.org/abs/1702.05659.

[40] Jingyi Xu et al. *A Semantic Loss Function for Deep Learning with Symbolic Knowledge*. 2017. DOI: 10.48550/ARXIV.1711.11157. URL: https://arxiv.org/abs/1711.11157.

[41] L. Amendola e S. Tsujikawa. *Dark Energy: Theory and Observations*. Cambridge University Press, 2010. ISBN: 9781139488570. URL: https://books.google.com.br/books?id=Xge0hg%5C_AIIYC.

[42] P.P. Coles e P.F. Lucchin. *Cosmology: The Origin and Evolution of Cosmic Structure*. Wiley, 2003. ISBN: 9780470852996. URL: https://books.google.com.br/books?id=BGYcivB1EtMC.

[43] Andrew R. Liddle e David H. Lyth. *Cosmological Inflation and Large-Scale Structure*. 2000.

[44] Steven Weinberg. *Cosmology*. First Edition. Oxford University Press, 2008. ISBN: 978–0–19–852682–7.

[45] Scott Dodelson e Fabian Schmidt. *Modern Cosmology*. First Edition. Amsterdam: Academic Press, 2021. ISBN: 978-0-12-815948-4.

[46] S. Carroll, S.M. Carroll e Addison-Wesley. *Spacetime and Geometry: An Introduction to General Relativity*. Addison Wesley, 2004. ISBN: 9780805387322. URL: https://books.google.com.br/books?id=1SKFQgAACAAJ.

[47] J.A. Peacock e Cambridge University Press. *Cosmological Physics*. Cambridge Astrophysics. Cambridge University Press, 1999. ISBN: 9780521422703. URL: https://books.google.com.br/books?id=t8O-yylU0j0C.

[48] V. M. Slipher. «The radial velocity of the Andromeda Nebula». Em: *Lowell Observatory Bulletin* 2.8 (jan. de 1913), pp. 56–57.

[49] A Friedman. «On the Curvature of Space». Em: *General Relativity and Gravitation* 31.12 (1999). DOI: 10.1023/A:1026751225741.

[50] Ian Steer. «Who discovered Universe expansion?» Em: *Nature* 490.7419 (out. de 2012). DOI: 10.1038/490176c.

[51] Immo Appenzeller. *High-Redshift Galaxies*. Springer, 2016. ISBN: 978-3-540-75823-5.

[52] Kevork Abazajian et al. «The First Data Release of the Sloan Digital Sky Survey». Em: *The Astronomical Journal* 126.4 (out. de 2003), pp. 2081–2086. DOI: 10.1086/378165. URL: https://doi.org/10.1086%5C%2F378165.

[53] Carlos E. Cunha et al. «Spectroscopic failures in photometric redshift calibration: cosmological biases and survey requirements». Em: *Monthly Notices of the Royal Astronomical Society* 444.1 (ago. de 2014), pp. 129–146. ISSN: 0035-8711. DOI: 10.1093/mnras/stu1424. eprint: https://academic.oup.com/mnras/article-pdf/444/1/129/18503207/stu1424.pdf. URL: https://doi.org/10.1093/mnras/stu1424.

[54] V. A. Ambartsumyan. «Problems of Extragalactic Research». Em: *Problems of Cosmogeny* 8 (jan. de 1962), p. 2.

[55] Ian Ridpath. *A Dictionary of Astronomy*. Second Edition. Oxford University Press, 2012. ISBN: 9780199609055.

[56] W. A. Baum. «Photoelectric determinations of redshifts beyond 0.2 c.» Em: 62 (fev. de 1957), pp. 6–7. DOI: 10.1086/107433.

[57] D. C. Koo. «Optical multicolors : a poor person's Z machine for galaxies.» Em: 90 (mar. de 1985), pp. 418–440. DOI: 10.1086/113748.

[58] A. J. Connolly et al. «Slicing Through Multicolor Space: Galaxy Redshifts from Broadband Photometry». Em: *The Astronomical Journal* 110 (dez. de 1995), p. 2655. DOI: 10.1086/117720. URL: https://doi.org/10.1086%5C%2F117720.

[59] R. Pello et al. «Identification of a high redshift cluster. in the field of Q2345+007 through deep BRIJK' photometry». Em: 314 (out. de 1996), pp. 73–86. arXiv: astro-ph/9603146 [astro-ph].

[60] J. -M. Miralles, R. Pellö e J. -F. Le Borgne. «Photometric Analysis of the Hubble Deep Field». Em: (jan. de 1997). Ed. por Nial R. Tanvir, Alfonso Aragon-Salamanca e Jasper V. Wall, p. 161.

[61] Gustavo Bruzual A e Stephane Charlot. «Spectral evolution of stellar populations using isochrone synthesis». Em: *The Astrophysical Journal* 405 (1993), pp. 538–553.

[62] Roberto Tagliaferri et al. «Neural networks for photometric redshifts evaluation». Em: *Lecture Notes in Computer Science* (2003), pp. 226–236.

[63] Andrew E Firth, Ofer Lahav e Rachel S Somerville. «Estimating photometric redshifts with artificial neural networks». Em: *Monthly Notices of the Royal Astronomical Society* 339.4 (2003), pp. 1195–1202.

[64] B. Garilli et al. «The Vimos VLT deep survey». Em: *Astronomy &amp Astrophysics* 486.3 (jun. de 2008), pp. 683–695. DOI: 10.1051/0004-6361:20078878. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%3A20078878.

[65] Garilli, B. et al. «The VIMOS Public Extragalactic Survey (VIPERS) - First Data Release of 57 spectroscopic measurements». Em: *A&A* 562 (2014), A23. DOI: 10.1051/0004-6361/201322790. URL: https://doi.org/10.1051/0004-6361/201322790.

[66] Le Fèvre, O. et al. «The VIMOS VLT deep survey - First epoch VVDS-deep survey: 11 spectra with 17.5*textit{*textbf{*smallAB}}$andtheredshiftdistributionover0». Em: *A&A* 439.3 (2005), pp. 845–862. DOI: 10.1051/0004-6361:20041960. URL: https://doi.org/10.1051/0004-6361:20041960.

[67] Jakob Walcher and Brent Groves and Tamás Budavári and Daniel Dale. «Fitting the integrated spectral energy distributions of galaxies». Em: *Astrophysics and Space Science* 331.1 (ago. de 2010), pp. 1–51. DOI: 10.1007/s10509-010-0458-z. URL: https://doi.org/10.1007%5C%2Fs10509-010-0458-z.

[68] L. Guzzo et al. «The VIMOS Public Extragalactic Redshift Survey (VIPERS)». Em: *Astronomy &amp Astrophysics* 566 (jun. de 2014), A108. DOI: 10.1051/0004-6361/201321489.

[69] S. Arnouts e O. Ilbert. *LePHARE: Photometric Analysis for Redshift Estimate*. Astrophysics Source Code Library, record ascl:1108.009. Ago. de 2011. ascl: 1108.009.

[70] O. Ilbert et al. «Accurate photometric redshifts for the CFHT legacy survey calibrated using the VIMOS VLT deep survey». Em: *Astronomy &amp Astrophysics* 457.3 (set. de 2006), pp. 841–856. DOI: 10.1051/0004-6361:20065138.

[71] Microsoft Support Team. *Introduction to queries*. URL: https://support.microsoft.com/en-us/office/introduction-to-queries-a9739a09-d3ff-4f36-8ac3-5760249fb65c. (accessed: 14.07.2022).

[72] Krzysztof M. Gorski et al. «The HEALPix Primer». Em: (1999). DOI: 10.48550/ARXIV.ASTRO-PH/9905275. URL: https://arxiv.org/abs/astro-ph/9905275.

[73] Francisco J. Valverde-Albacete, Jorge Carrillo de Albornoz e Carmen Peláez-Moreno. «A Proposal for New Evaluation Metrics and Result Visualization Technique for Sentiment Analysis Tasks». Em: (2013).

[74] Y. Ma e H. He. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley, 2013. ISBN: 9781118646335. URL: https://books.google.com.br/books?id=CVHx-Gp9jzUC.

[75] Nitesh V. Chawla. *Data Mining for Imbalanced Datasets: An Overview*. Ed. por Oded Maimon e Lior Rokach. Boston, MA: Springer US, 2010, pp. 875–886. ISBN: 978-0-387-09823-4. DOI: 10.1007/978-0-387-09823-4_45. URL: https://doi.org/10.1007/978-0-387-09823-4_45.

[76] M. Kubat. «Addressing the Curse of Imbalanced Training Sets: One-Sided Selection». Em: *Fourteenth International Conference on Machine Learning* (jun. de 2000).

[77] Jon Louis Bentley. «Multidimensional Binary Search Trees Used for Associative Searching». Em: *Commun. ACM* 18.9 (set. de 1975), p. 9. ISSN: 0001-0782. DOI: 10.1145/361002.361007.

[78] Alvin C. Rencher. *Methods of Multivariate Analysis*. Second Edition. John Wiley Sons, 2002. ISBN: 0-471-41889-7.

[79] Scikit-Learn. *sklearn.linear$_m$odel.LinearRegression*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. (accessed: 23.10.2022).

[80] Scikit-Learn. *Linear, Quadratic, and Regularized Discriminant Analysis*. URL: https://www.datascienceblog.net/post/machine-learning/linear-discriminant-analysis/. (accessed: 23.10.2022).

[81] Kevin Beyer et al. «When Is "Nearest Neighbor" Meaningful?» Em: *Database Theory — ICDT'99*. Ed. por Catriel Beeri e Peter Buneman. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235. ISBN: 978-3-540-49257-3.

[82] T. Cover e P. Hart. «Nearest neighbor pattern classification». Em: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[83] Scikit-Learn. *sklearn.tree.DecisionTreeClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. (accessed: 23.10.2022).

[84] L. Breiman et al. «Classification and Regression Trees». Em: 1983.

[85] Scikit-Learn. *sklearn.ensemble.RandomForestClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. (accessed: 23.10.2022).

[86] Scikit-Learn. *sklearn.ensemble.AdaBoostClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html. (accessed: 23.10.2022).

[87] Ji Zhu et al. «Multi-class AdaBoost». Em: *Statistics and its interface* 2 (fev. de 2006). DOI: 10.4310/SII.2009.v2.n3.a8.

[88] Scikit-Learn. *sklearn.discriminant$_a$nalysis.QuadraticDiscriminantAnalysis*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html. (accessed: 23.10.2022).

[89] Matthias Doring. *sklearn.neighbors.KNeighborsClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. (accessed: 23.10.2022).

[90] Alexandre Gori Maia. *Econometria*. 1ª ed. Saint Paul, 2017, p. 384. ISBN: 978-8580041286.

[91] C. E. Rasmussen e C. K. I. Williams. *Gaussian Processes for Machine Learning*. First Edition. 55 Hayward Street, Cambridge, MA 02142: The MIT Press, 2006. ISBN: 10 0-262-18253-X.

[92] Marc Deisenroth. *Machine Learning Tutorial: Gaussian Processes*. https://www.youtube.com/98SYOdlY. Online; accessed 05-October-2022. Nov. de 2016.

[93] Kilian Weinberger. *Lecture 15: Gaussian Processes*. https://www.cs.cornell.edu/courses/cs478 Online; accessed 05-October-2022. Jul. de 2018.

[94] J. Görtler, R. Kehlbeck e O. Deussen. «A Visual Exploration of Gaussian Processes». Em: *Proceedings of the Workshop on Visualization for AI Explainability (VISxAI)*. 2018. URL: https://www.jgoertler.com/visual-exploration-gaussian-processes/.

[95] Wenqi Fang et al. «Sparse Gaussian Process Based On Hat Basis Functions». Em: (2020). DOI: 10.48550/ARXIV.2006.08117. URL: https://arxiv.org/abs/2006.08117.

[96] Zahra Gomes et al. «Improving photometric redshift estimation using GPz: size information, post processing, and improved photometry». Em: *Monthly Notices of the Royal Astronomical Society* 475.1 (dez. de 2017), pp. 331–342. ISSN: 0035-8711. DOI: 10.1093/mnras/stx3187. eprint: https://academic.oup.com/mnras/article-pdf/475/1/331/23417059/stx3187.pdf. URL: https://doi.org/10.1093/mnras/stx3187.

[97] Bruno Cabral. «Processos Gaussianos para Aprendizado Supervisionado». 2021. eprint: https://repositorio.ufsc.br/bitstream/handle/123456789/224907/Processos\%20Gaussianos\%20para\%20Aprendizado\%20Supervisionado\%20-\%20Bruno\%20R.\%20Cabral.pdf?sequence=1&isAllowed=y.

[98] Adrian A. Collister e Ofer Lahav. «ANNz: Estimating Photometric Redshifts Using Artificial Neural Networks». Em: *arXiv:astro-ph/0311058* (2004).

[99] Kyle S. Dawson et al. «The Baryon Oscillation Spectroscopic Survey of SDSS-III». Em: *The Astronomical Journal* 145.1 (dez. de 2012), p. 10. DOI: 10.1088/0004-6256/145/1/10.

[100] A. Hoecker et al. *TMVA - Toolkit for Multivariate Data Analysis*. 2007. DOI: 10.48550/ARXIV.PHYSICS/0703039. URL: https://arxiv.org/abs/physics/0703039.

[101] Hiroaki Oyaizu et al. «A Galaxy Photometric Redshift Catalog for the Sloan Digital Sky Survey Data Release 6». Em: *The Astrophysical Journal* 674.2 (fev. de 2008), pp. 768–783. DOI: 10.1086/523666.

[102] Francois Chollet. *Deep Learning with Python*. First Edition. Manning Publications Co, 2017. ISBN: 978-1617294433.

[103] Kin Lane. *Intro to APIs: History of APIs*. Accessed: 2022-10-19. 2019. URL: https://blog.postman.com/intro-to-apis-history-of-apis/.

[104] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. 2016. DOI: 10.48550/ARXIV.1603.04467. URL: https://arxiv.org/abs/1603.04467.

[105] Abhinav Vishnu, Charles Siegel e Jeffrey Daily. *Distributed TensorFlow with MPI*. 2016. DOI: 10.48550/ARXIV.1603.02339. URL: https://arxiv.org/abs/1603.02339.

[106] J D Rivera et al. «Degradation analysis in the estimation of photometric redshifts from non-representative training sets». Em: *Monthly Notices of the Royal Astronomical Society* 477.4 (abr. de 2018), pp. 4330–4347. DOI: 10.1093/mnras/sty880. URL: https://doi.org/10.1093%5C%2Fmnras%5C%2Fsty880.

[107] Jeffrey A. Newman et al. «The DEEP2 Galaxy Redshift Survey: Design, Observations, Data Reduction, and Redshifts». Em: 208.1 (ago. de 2013), p. 5. DOI: 10.1088/0067-0049/208/1/5. URL: %5Curl%7Bhttps://doi.org/10.1088%5C%2F0067-0049%5C%2F208%5C%2F1%5C%2F5%7D.

[108] F. (Director) Lang. *Metropolis*. Parufamet. 1937.

[109] Tim Pelan. *Gotham by Elektrisches Licht: Fritz Lang's 'Metropolis'*. URL: https://cinephiliabeyond.org/metropolis/. (accessed: 01/02/2023).

[110] Walter Pitts Warren S. McCulloch. «A logical calculus of the ideas immanent in nervous activity». Em: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 1522–9602. DOI: 10.1007/BF02478259.

[111] Donald Hebb. *The Organization of Behavior*. 1949, p. 365. DOI: 10.1007/978-3-642-70911-1_15.

[112] Nathaniel Rochester. «Symbolic programming». Em: *Trans. I R E Prof. Group Electron. Comput.* 2 (1953), pp. 10–15.

[113] J. McCarthy et al. *A proposal for the Dartmouth summer research project on artificial intelligence*. 1955.

[114] Yoav Freund e Robert Schapire. «Large Margin Classification Using the Perceptron Algorithm». Em: *Machine Learning* 37 (fev. de 1999). DOI: 10.1023/A:1007662407062.

[115] Vladimir Vapnik e Alexey Chervonenkis. *Theory of pattern recognition*. 1974.

[116] Ryan McDonald, Keith Hall e Gideon Mann. «Distributed training strategies for the structured perceptron». Em: *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. 2010, pp. 456–464.

[117] Gerald DeJong e Raymond Mooney. «Explanation-based learning: An alternative view». Em: *Machine learning* 1 (1986), pp. 145–176.

[118] John J Hopfield. «Hopfield network». Em: *Scholarpedia* 2.5 (2007), p. 1977.

[119] Williams R. Rumelhart D. Hinton G. «Learning representations by back-propagating errors». Em: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[120] Larry Greenemeier. *20 Years after Deep Blue: How AI Has Advanced Since Conquering Chess. IBM AI expert Murray Campbell reflects on the machine's long, bumpy road to victory over chess champ Garry Kasparov*. 2017. URL: https://www.scientificamerican.com/article/20-years-after-deep-blue-how-ai-has-advanced-since-conquering-chess/. (accessed: 01/02/2023).

[121] Feng-hsiung Hsu. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. Boston, MA, 2004, p. 320. ISBN: 9780691235134.

[122] Geoffrey Hinton. «Deep Learning—A Technology With the Potential to Transform Health Care». Em: *JAMA* 320.11 (set. de 2018), pp. 1101–1102. ISSN: 0098-7484. DOI: 10.1001/jama.2018.11100. eprint: https://jamanetwork.com/journals/jama/articlepdf/2701666/jama_hinton_2018_vp_180096.pdf. URL: https://doi.org/10.1001/jama.2018.11100.

[123] Hannes Schulz e Sven Behnke. «Deep Learning». Em: *KI - Künstliche Intelligenz* 26 (nov. de 2012). DOI: 10.1007/s13218-012-0198-z.

[124] Alex Krizhevsky, Ilya Sutskever e Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». Em: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[125] M. Scodeggio et al. «The VIMOS Public Extragalactic Redshift Survey (VIPERS)». Em: *Astronomy &amp Astrophysics* 609 (jan. de 2018), A84. DOI: 10.1051/0004-6361/201630114. URL: https://doi.org/10.1051%5C%2F0004-6361%5C%2F201630114.

[126] SDSS collaboration. *The Sloan Digital Sky Survey*. URL: https://classic.sdss.org/. (accessed: 25.01.2023).

[127] Andrés Almeida et al. *The Eighteenth Data Release of the Sloan Digital Sky Surveys: Targeting and First Spectra from SDSS-V*. 2023. DOI: 10.48550/ARXIV.2301.07688. URL: https://arxiv.org/abs/2301.07688.

[128] Sergey Ioffe e Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. DOI: 10.48550/ARXIV.1502.03167. URL: https://arxiv.org/abs/1502.03167.