

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/352749819>

Ciência dos Dados pelo Processo de KDD

Book · June 2021

DOI: 10.6084/m9.figshare.14850030.v1

CITATIONS

0

READS

228

1 author:



Rosalvo Ferreira de Oliveira Neto

Universidade Federal do Vale do São Francisco (UNIVASF)

30 PUBLICATIONS 56 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Project Granularity Transformations in Relational Databases [View project](#)



Project Biometry [View project](#)

Ciência dos Dados PELO PROCESSO DE KDD



ROSALVO NETO

Ciência dos Dados pelo Processo de KDD

© 2021, Rosalvo Ferreira de Oliveira Neto

Os direitos de todos os textos contidos neste livro eletrônico são reservados a seu autor, e estão registrados e protegidos pelas leis do direito autoral. Esta é uma edição eletrônica (e-book) não comercial, que não pode ser vendida nem comercializada em hipótese nenhuma, nem utilizada para quaisquer fins que envolvam interesse monetário.

O autor empenhou-se para citar adequadamente e dar o devido crédito a todos os detentores dos direitos autorais de qualquer material utilizado neste livro, dispendo-se a possíveis acertos caso, inadvertidamente, a identificação de algum deles tenha sido omitida.

Ciência dos Dados pelo Processo de KDD

ISBN: 978-65-00-24528-8

Diagramação e editoração: Rosalvo Ferreira de Oliveira Neto

Capa: Windson Chaves Fonseca

Contato: windsonfonseca@outlook.com

Revisão do texto: Neivonete Gomes Souza de Oliveira e Lorena Galvão Carvalho de Oliveira

Editora: O autor.

Endereço: Av. Avenida João Pernambuco, número 800, Condomínio Sol Nascente Orla, Fernando Idalino, CEP: 56332-710. Petrolina-PE.

Contato com o autor: rosalvo.neto@gmail.com

As imagens da capa, sumário, lista de figuras e tabelas são de domínio público e podem ser usadas desde que não sejam para fins comerciais. CC BY-NC-SA 3.0 (<http://creativecommons.org/licenses/by-nc-sa/3.0/>)

A imagem usada na abertura dos capítulos é uma foto tirada pelo autor do livro. A foto mostra o rio São Francisco e a ponte presidente Dutra, que liga as cidades de Petrolina-PE e Juazeiro-BA.

A cobra na capa é uma homenagem à distribuição Anaconda do Python e a mentalidade mamba.

**Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)**

Oliveira Neto, Rosalvo Ferreira de
Ciência dos dados pelo processo de KDD [livro
eletrônico] / Rosalvo Ferreira de Oliveira Neto. --
Petrolina, PE : Ed. do Autor, 2021.
PDF

ISBN 978-65-00-24528-8

1. Análise de sistemas 2. Ciência da computação
3. Ciência dos dados I. Título.

21-68653

CDD-004

Índices para catálogo sistemático:

1. Ciência da computação 004

Maria Alice Ferreira - Bibliotecária - CRB-8/7964

DEDICATORIA

Dedico este livro aos alunos que me pediram para escrever um livro com uma abordagem simples e direta sobre Ciência dos Dados.

AGRADECIMENTOS

À minha esposa Lorena e minha filha Luana, pelo incentivo constante durante a escrita deste livro!

À minha mãe Neivonete e meu pai Rivaldo por me mostrarem desde cedo o poder transformador da educação!

À Deus por me conceder saúde e paz!

SOBRE O AUTOR

Rosalvo Neto é doutor em Ciências da Computação pela Universidade Federal de Pernambuco. Em 2021, ele está completando 18 anos da sua jornada em Inteligência Artificial e Ciências dos Dados. Foram oito anos na NeuroTech, empresa pioneira no Brasil na criação de soluções avançadas de Inteligência Artificial, *Machine Learning* e *Big Data*. Desde 2011, ele é professor efetivo da Universidade Federal do Vale do São Francisco – UNIVASF, onde leciona as disciplinas de Inteligência Artificial e Inteligência Computacional. Ele têm publicações em relevantes jornais e conferências internacionais: *Expert Systems with Applications*, *Computers and Electronics in Agriculture*, *International Joint Conference on Artificial Neural Networks* e *International Conference on Software Engineering and Knowledge Engineering*. Ele também têm artigos publicados nos principais eventos organizados pela Sociedade Brasileira de Computação relacionados à área de Inteligência Artificial: Conferência Brasileira de Sistema Inteligente (BRACIS), Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), Simpósio Brasileiro de Banco de Dados (SBB), Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg) e Simpósio Brasileiro de Sistemas de Informação (SBSI).



Sumário

1	Introdução	15
1.1	O que é Ciência dos Dados?	15
1.2	Qual a Relação da Ciência dos Dados com <i>Big Data</i> e <i>Deep Learning</i> ?	16
1.2.1	<i>Big Data</i>	16
1.2.2	<i>Deep Learning</i>	17
1.3	Processo de Descoberta do Conhecimento	18
1.4	O papel do cientista de dados	19
1.5	Escopo e Organização do livro	21
2	Aprendizado de Máquina	23
2.1	Aprendizado Indutivo	24
2.2	Aprendizado Supervisionado	25
2.2.1	Conceitos Importantes	27
2.3	Paradigmas de Aprendizado Supervisionado	27
2.3.1	Por analogia	27
2.3.2	Baseado em regras	29
2.3.3	Conexionista	32
2.3.4	Atualização dos pesos da rede MLP	36
2.3.5	Bayesiano	48
2.4	<i>Ensemble Models</i>	49
2.5	Superfícies de Decisão	54

3	Pré-processamento	57
3.1	Definição da Granularidade do Projeto	57
3.1.1	Aplicação 1 - Análise de Risco de Crédito	57
3.1.2	Aplicação 2 - Cupons de descontos para Lojas de Comércio Eletrônico	58
3.1.3	Aplicação 3 - Classificação de textos em relação ao perfil do autor	60
3.2	Tratamento de Valores Ausentes	61
3.3	Tratamento de Valores Aberrantes	61
3.3.1	Identificação	62
3.3.2	Tratamento	62
3.4	Construção de novas variáveis	62
3.5	Redução de Dimensionalidade	64
3.5.1	<i>Filter</i> Univariável	65
3.5.2	<i>Filter</i> Multivariável	66
3.5.3	<i>Wrapper</i> Univariável	66
3.5.4	<i>Wrapper</i> Multivariável	66
3.6	Etapa de Transformação	66
4	Avaliação de Desempenho	69
4.1	Métodos de Avaliação de Desempenho	69
4.1.1	<i>Hold-out</i>	69
4.1.2	<i>K-Fold-Cross-Validation</i>	69
4.1.3	<i>Leave-one-out</i>	70
4.1.4	<i>Bootstrap</i>	70
4.2	Métricas de Avaliação de Desempenho	70
4.2.1	Classificação	71
4.2.2	Rregressão	73
5	Estudo de Caso	77
5.1	Pré-requisitos técnicos para execução do projeto	77
5.1.1	Python	77
5.1.2	Jupyter Notebook	77
5.1.3	Pandas	79
5.1.4	Scikit-learn	79
5.2	Projeto	79
5.2.1	Seleção dos dados	79
5.2.2	Pré-processamento	79
5.2.3	Transformação	87
5.2.4	<i>Data Mining</i>	88
5.2.5	Avaliação	89



List of Figures

1.1	áreas relacionadas da Ciência dos Dados. Adaptado das notas de aulas do Prof. Dr. Peer Kröger	16
1.2	Exemplo de uma arquitetura de <i>Deep Learning</i>	17
1.3	Visão geral do processo de KDD	19
1.4	O processo de KDD agrupado em cinco etapas	20
1.5	Diagrama de Venn da Ciência dos Dados	20
2.1	Exemplo de funcionamento de um autoencoder	24
2.2	Processo de aprendizado supervisionado.	26
2.3	Exemplo de três funções $h(x)$ que tentam se aproximar de uma função $f(x)$. A função em vermelho está sendo representada por um modelo linear. A função em verde representa um boa aproximação. A função em azul memorizou todas as entradas.	26
2.4	Exemplo de cálculo da similaridade entre padrões no KNN usando a distância euclidiana.	28
2.5	Fluxograma do algoritmo KNN.	29
2.6	Exemplo de uma Árvore de Decisão.	30
2.7	Funcionamento geral do algoritmo de árvore de decisão	30
2.8	Gráfico da variação dos valores da entropia com relação a mudança na distribuição da variável alvo.	32
2.9	Topologia típica de uma rede MLP.	34
2.10	Algoritmo geral de treinamento de uma rede MLP	35
2.11	Processo de perda de generalização durante o treinamento de uma rede MLP.	36
2.12	Estado do neurônio McP.	37
2.13	Função degrau.	37
2.14	Demonstração do procedimento para considerar <i>threshold</i> como um parâmetro livre.	38
2.15	Gráfico em três dimensões da função erro.	39
2.16	Superfície de decisão do Perceptron para a função AND.	41
2.17	Distribuição das classes para a função XOR.	42

2.18	Gráfico em 3D com as saídas dos três neurônios da camada oculta para a função XOR no início do treinamento da rede.	43
2.19	Gráfico em 3D com as saídas dos três neurônios da camada oculta para a função XOR após o treinamento da rede.	44
2.20	Gráfico da função Sigmoidal Logística.	45
2.21	Padrões de nomenclatura para as deduções do <i>backpropagation</i>	45
2.22	Ilustração do processo de cálculo do erro para neurônios da camada oculta.	47
2.23	Probabilidades condicionais de X em relação a Y.	49
2.24	Funcionamento do algoritmo Naive Bayes para classificação de um novo padrão	50
2.25	Exemplo de classificação com Naive Bayes.	50
2.26	Workflow da abordagem de <i>Ensemble Models Bagging</i>	51
2.27	Workflow da abordagem de <i>Ensemble Models Boosting</i>	52
2.28	Workflow da abordagem de <i>Ensemble Models Stacking</i>	52
2.29	Algoritmo geral do <i>Random Forest</i>	53
2.30	Workflow do <i>Random Forest</i>	53
2.31	Distribuição das três classes do problema Flor de Íris em relação às variáveis de entrada comprimento e a largura da sépala (cm).	54
2.32	Diferentes Superfície de Decisão para o problema Flor de Íris	55
3.1	Exemplo de definição da Granularidade do projeto a partir de um Banco de Dados Relacional	59
3.2	Exemplo de definição da Granularidade do projeto a partir de um arquivo de LOG	60
3.3	Exemplo de definição da Granularidade do projeto a partir de um <i>corpus</i>	61
3.4	Superfície de decisão hipótetica para o modelo $Y = aX + b$	63
3.5	Superfície de decisão hipótetica para o modelo $Y = aX + b + cX^3$	64
3.6	Topologia das abordagens de seleção de variáveis	65
4.1	Ilustração da divisão do conjunto de treinamento em <i>folds</i> para um <i>K Fold-Cross-Validation</i> com $K = 10$	70
4.2	Ilustração da Curva ROC	73
4.3	Ilustração de um gráfico de dispersão	75
5.1	Tela de exemplo no Jupyter	78
5.2	Ilustração da mudança de granularidade	82
5.3	Fluxograma de tratamento para os valores ausentes	83



List of Tables

2.1	Amostra de um conjunto de exemplos	27
2.2	Variação da classe alvo em dez bases de dados	31
2.3	Base de dados com informações sobre quando jogar tênis	33
2.4	Tabela verdade para a função AND	41
2.5	Tabela verdade para a função XOR	42
3.1	Tabela contendo a variável original Grupo	67
3.2	Tabela contendo as variáveis <i>Dummies</i> referente à variável Grupo	67
4.1	Elementos de uma Matriz de Confusão	71
4.2	Classificação de quatro padrões	72
4.3	Matriz de Confusão preenchida com os valores do exemplo	72
4.4	Matriz de Confusão com nomenclatura usada para auxiliar na definição de outras métricas de avaliação de desempenho	73
5.1	Lista de variáveis da competição DMC 2013	80



1. Introdução

Este capítulo tem como objetivo mostrar a relação entre a ciência dos dados e o processo de descoberta do conhecimento em bases de dados do inglês *Knowledge Discovery in Databases* (KDD). O capítulo está organizado da seguinte forma. Primeiro, são fornecidas definições para Ciência dos Dados. Segundo, é discutido como *Big Data* e *Deep Learning* estão relacionados com esta área. Terceiro, o processo de KDD é apresentado. Na sequência, exemplos de requisitos exigidos por empresas para vagas de cientista de dados são apresentados. Por fim, é apresentado o escopo deste livro e como ele está estruturado.

1.1 O que é Ciência dos Dados?

Durante a escrita deste livro, foram realizadas buscas sobre definições de Ciência dos Dados no portal *Google Scholar* e nos portais das principais editoras científicas da área de computação, *Institute of Electrical and Electronics Engineers* (IEEE) e Elsevier. A seguir serão apresentadas as definições mais interessantes encontradas.

- A Ciência dos Dados é uma prática interdisciplinar com base em métodos de Engenharia de Dados, Estatística, Mineração de Dados, Aprendizado de Máquina e Análise Preditiva [ZM14];
- A Ciência dos Dados é um novo campo que incorpora uma série de disciplinas e corpos de conhecimento relevantes, como estatística, informática, computação, comunicação, gestão e sociologia, para estudar dados e seu domínio [Cao18];
- Uma definição mais abstrata é fornecida por [PF13], a Ciência de Dados é um conjunto de princípios fundamentais que auxiliam a extração de informações e conhecimento dos dados. Ainda de acordo com [PF13], o conceito mais relacionado à Ciência de Dados é Mineração de Dados.

De acordo com as notas de aula do Prof. Dr. Peer Kröger¹ da Universidade de Munique (Ludwig-Maximilians-Universität München), a Ciência dos Dados pode ser vista como um guarda chuva que engloba todas as áreas relacionadas, conforme ilustra a Figura 1.1.

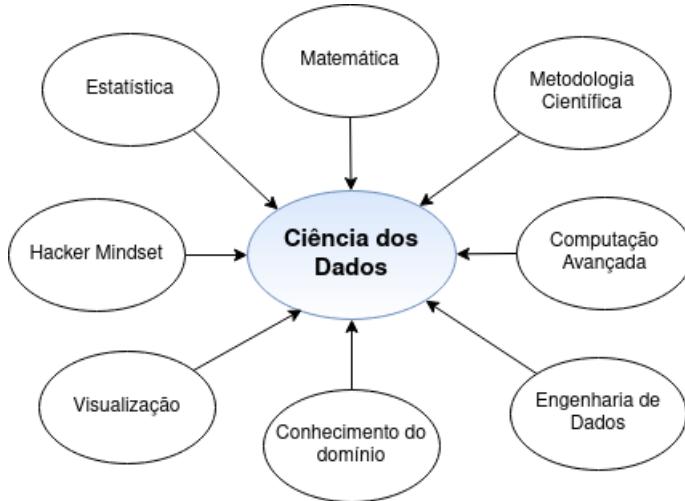


Figura 1.1: áreas relacionadas da Ciência dos Dados. Adaptado das notas de aulas do Prof. Dr. Peer Kröger

1.2 Qual a Relação da Ciência dos Dados com *Big Data* e *Deep Learning* ?

A quantidade de dados disponíveis é muito grande, e as empresas em quase todos os setores estão focadas em explorar esses dados para obter vantagem competitiva [Fer+18]. As empresas usam a Ciência dos Dados para explorar esses dados. Uma pergunta que surge é “o que é a Ciência do Dados ?”. Três definições foram fornecidas na seção anterior. No entanto, de acordo com [KD19a], a resposta a essa pergunta é variada e às vezes confusa e conflitante. De acordo com [PF13], uma das justificativas para essa dificuldade é que a Ciência de Dados está intrinsecamente relacionada a outros conceitos importantes, como *Big Data* e *Deep Learning*. Desta forma, para que a Ciência de Dados sirva aos negócios de maneira eficaz, é importante entender suas relações com esses dois conceitos importantes.

1.2.1 *Big Data*

O termo WEB 2.0 surgiu em meados dos anos 2000 [GH15] e se refere à versão atual da internet em que os usuários possuem mais interação e colaboração uns com os outros, diferente da primeira versão da internet em que os usuários eram limitados à visualização passiva de conteúdo. Como exemplos de sites criados a partir da WEB 2.0 podemos citar: as redes sociais, blogs e sites de compartilhamento de vídeo [Rib+17].

A tecnologia de banco de dados relacional não atendeu a necessidade de armazenamento dessa nova demanda. Para lidar com esta nova demanda, surgiu a tecnologia *Big Data*. [Tsa+15] descrevem *Big Data* como uma tecnologia que possui três características: Volume, Variedade e Velocidade, também conhecida como 3Vs:

¹https://www.dbs.ifii.lmu.de/Lehre/KDD_II/SoSe18/1-Introduction.pdf

Fonte: <http://www.rsipvision.com/exploring-deep-learning/>

Deep neural networks learn hierarchical feature representations

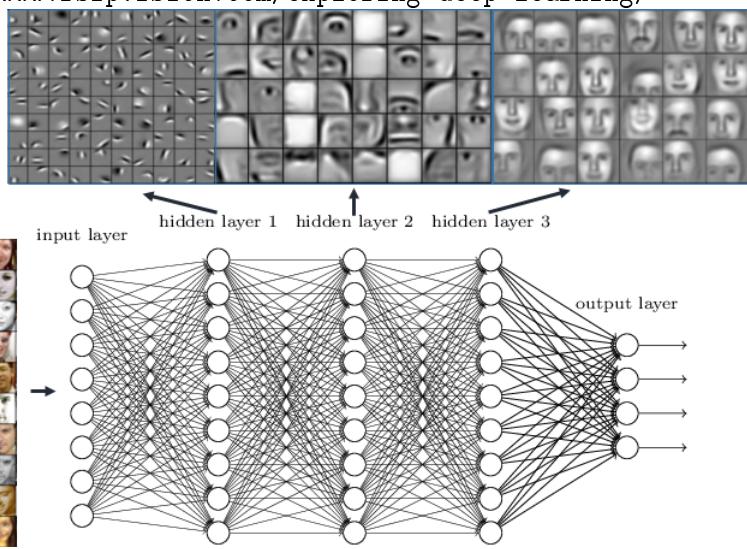


Figura 1.2: Exemplo de uma arquitetura de *Deep Learning*

- **Volume:** Armazenar e analisar um grande volume de dados;
- **Variedade:** Manipular diversos tipos de dados desde de estruturados até dados semiestruturados ou não estruturados;
- **Velocidade:** Lidar com a demanda de alta velocidade com que os dados são acessados e armazenados.

Os bancos de dados (NoSQL) (*Not only SQL*) surgiram para atender as demandas dos 3Vs de *Big Data*. [Cat11] define NoSQL como uma nova geração de banco de dados que permite um alto desempenho e processamento rápido de informações em larga escala.

1.2.2 Deep Learning

Aprendizado de máquina é um campo da Inteligência Artificial focado no estudo de programas de computador que podem melhorar o desempenho por meio da experiência [DN17]. O desempenho de algoritmos de aprendizado de máquina tradicionais depende muito da representação dos dados que eles recebem como entrada [KSH17]. Essa representação é feita de forma manual e depende de conhecimento do domínio da aplicação. Uma solução para evitar essa dependência é usar o aprendizado de máquina para descobrir não apenas o mapeamento da representação de entrada para a saída, mas também a própria representação de entrada [GBC16]. Esta abordagem é conhecida como *Representation Learning*. Métodos de *Deep learning* são métodos de *Representation Learning* com múltiplos níveis de representação, obtidos pela composição de módulos simples, mas não lineares, em que cada um transforma a representação em um nível (começando com a entrada bruta) em uma representação em um nível superior, um pouco mais abstrato [Lec+98]. Com a composição de tais transformações, funções muito complexas podem ser aprendidas [LBH15]. Por exemplo, se considerarmos a rede neural profunda mostrada na Figura 1.2, a entrada pode ser apresentada como uma matriz de pixels. A primeira camada pode identificar linhas ou curvas. A segunda camada, por sua vez, detecta características mais complexas como olhos, nariz e orelhas. Finalmente, a terceira camada pode procurar padrões como rostos ou expressões faciais [DN17].

Como a Ciência dos Dados está relacionada com *Big Data* e *Deep Learning*? a Ciência dos Dados pode usar *Big Data* e *Deep Learning* para construir soluções. Não é obrigatório o seu uso. Isso depende da necessidade de cada projeto.

1.3 Processo de Descoberta do Conhecimento

[ZM14] destacam que para garantir soluções de qualidade em projetos de Ciência dos Dados é importante a utilização de alguma metodologia que sirva de guia, especificando o fluxo a ser seguido e as atividades que devem ser realizadas. Nesta perspectiva, eu escolhi o processo de KDD para ser usado como metodologia na construção de projetos de Ciência dos Dados. A justificativa para essa escolha é porque ele é um processo mais simples, e por isso fácil para quem está iniciando na área.

O processo de KDD tornou-se uma área de pesquisa ativa em meados da década de 90. A definição do termo KDD foi introduzida por [Fay+96] como parte de um processo ainda mais amplo do que um algoritmo de Aprendizado de Máquina. KDD ou Descoberta do Conhecimento em Bases de Dados é um processo não trivial, iterativo, interativo e com múltiplos estágios que manipula e transforma os dados no intuito de descobrir padrões relevantes. A Figura 1.3 ilustra a visão geral do processo de KDD, que é composta por nove itens agrupados em cinco macro etapas, que serão detalhados a seguir. A Figura 1.4 exibe o processo de KDD com as cinco macro etapas.

1. **Entendimento do domínio da aplicação:** inclui definir o objetivo da aplicação e entender o que é relevante para o problema;
2. **Seleção dos Dados:** também chamado de amostragem dos dados, é o processo que define quais serão os dados utilizados no projeto. Os dados podem ser selecionados das mais diversas fontes, tais como: banco de dados relacional, arquivo texto, dentre outros;
3. **Limpeza dos dados e pré-processamento:** inclui escolhas de estratégias para lidar com *missing values* e *outliers*, bem como decidir questões de banco de dados, tais como: tipos de dados e esquemas;
4. **Redução dos dados e projeção:** inclui encontrar variáveis úteis para representar os dados dependendo do objetivo da tarefa. Métodos de transformação e redução de dimensionalidade são utilizados para reduzir o número efetivo de registros e variáveis a fim de encontrar a representação mais adequada para os dados;
5. **Escolha da função de mineração de dados:** inclui decidir o objetivo do modelo derivado pelo algoritmo de mineração de dados, que pode ser: sumarização, classificação, regressão ou agrupamento;
6. **Escolha do algoritmo de mineração de dados e transformação dos dados:** inclui selecionar o algoritmo de mineração de dados que será utilizado, assim como transformar os dados da etapa de pré-processamento no formato adequado do algoritmo selecionado. Por exemplo, árvores de decisão necessitam que as variáveis de entrada sejam categóricas, desta forma, as variáveis numéricas precisam ser discretizadas. Por outro lado, Redes Neurais *MultiLayer Perceptron* precisam que as variáveis de entrada sejam numéricas, por isso, as variáveis categóricas precisam ser binarizadas;
7. **Mineração de dados:** inclui a aplicação do algoritmo de mineração de dados e a busca dos seus parâmetros que maximizem o objetivo do projeto;
8. **Interpretação:** inclui a validação do conhecimento minerado, onde o especialista do domínio de aplicação é fundamental para homologação do conhecimento descoberto, pois nesta fase são validados todos os resultados obtidos no projeto;
9. **Utilização do conhecimento descoberto:** inclui incorporar o conhecimento dentro de um

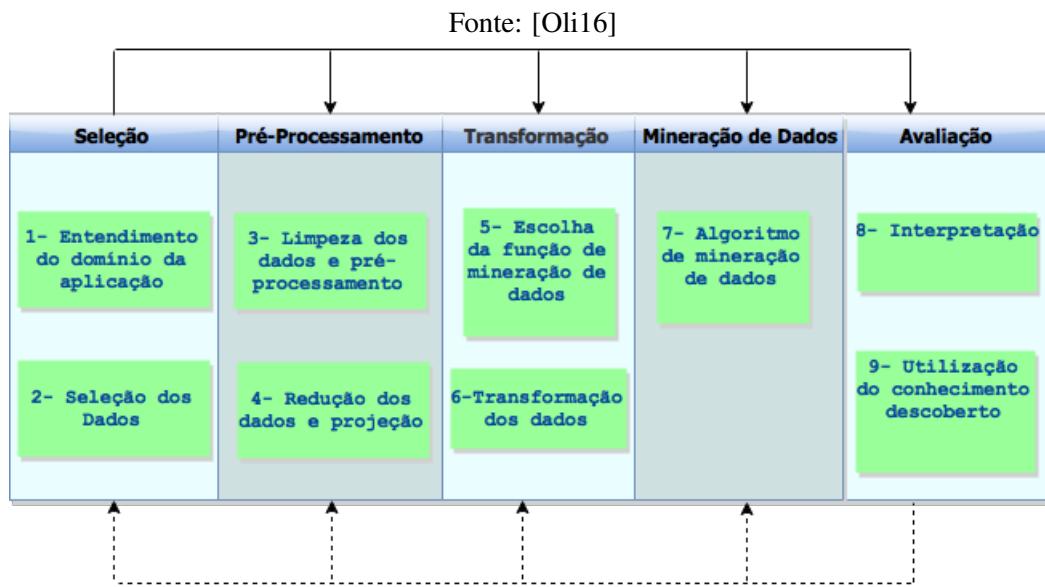


Figura 1.3: Visão geral do processo de KDD

sistema de decisão, no qual a empresa tomará decisões com base neste conhecimento ou apenas documentá-lo e apresentar para as partes interessadas.

1.4 O papel do cientista de dados

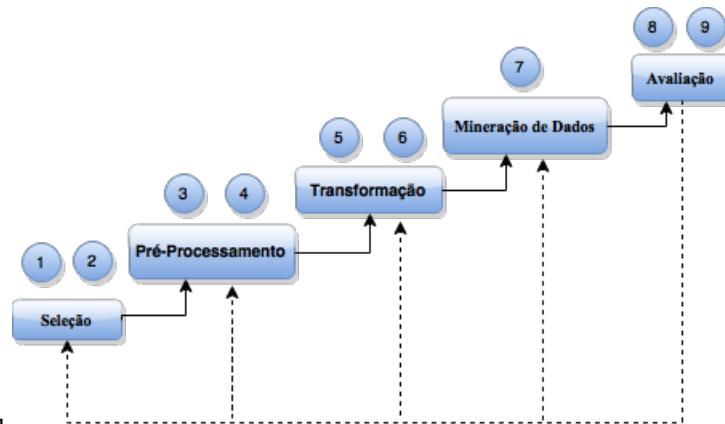
O diagrama de Venn exibido na Figura 1.5 ilustra as habilidades gerais esperadas por um cientista de dados. Este diagrama foi proposto por [Con12]. Para um maior entendimento destas habilidades, será fornecido um exemplo de atividade para cada habilidade com base em [Fin13]:

- **Computação e Dados:** Realizar a junção de vários conjuntos de dados;
- **Matemática e Estatística:** Reduzir o número de variáveis que precisam ser consideradas para uma análise particular;
- **Conhecimento do Domínio:** Explicar a origem de *outliers* em um determinado conjunto de dados.

De acordo com [ZM14], o cientista de dados é responsável por guiar um projeto de Ciência de Dados do início ao fim. Para um maior entendimento dos requisitos que uma empresa espera de um cientista de dados, foi realizada uma busca no portal LinkedIn² por vagas para cientista de dados no Brasil e no exterior. Foram selecionadas as seis primeiras vagas para extrair as habilidades mais frequentes exigidas e que são abordadas neste livro. O apêndice A contém as descrições completas das vagas. As habilidades identificadas foram:

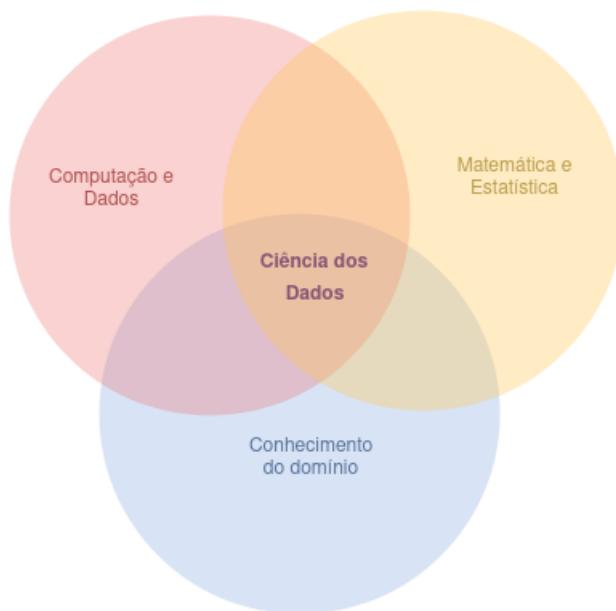
- Aprendizado de Máquina;
- Metodologia / processo para construção de projetos de Ciência dos Dados;
- Programação: escrever rotinas de preparação e análise de dados em Python usando as bibliotecas pandas e scikit-learn;

²<https://www.linkedin.com/>



Fonte: [Oli16]

Figura 1.4: O processo de KDD agrupado em cinco etapas



Adaptado de [Con12]

Figura 1.5: Diagrama de Venn da Ciência dos Dados

1.5 Escopo e Organização do livro

O escopo deste livro está delimitado para construção de projetos de Ciência dos Dados por iniciantes da área. Por esse motivo, não são abordados algoritmos de *Deep Learning* e nem banco de dados *NoSQL*. O restante do livro está organizado em capítulos que abordam as cinco macro etapas do processo do KDD. A sequência dos capítulos foi elaborada visando um melhor entendimento do leitor e está definida como segue. O Capítulo 2 apresenta os principais algoritmos de aprendizado de máquina, etapa de **Mineração de Dados**. O Capítulo 3 detalha as principais tarefas das etapas **Pré-processamento e Transformação**. O Capítulo 4 apresenta as principais métricas e métodos usados na etapa de **Avaliação**. Por fim, o Capítulo 5 fornece um estudo de caso para aplicação dos conceitos visto neste livro de forma prática.



2. Aprendizado de Máquina

Este capítulo apresenta a etapa de Mineração de Dados do processo de KDD, do inglês *Data Mining*. Essa etapa é o coração do processo de KDD por isso ela é apresentada logo no início deste livro. Mas por que o nome do capítulo é Aprendizado de Máquina e não Mineração de Dados? Porque é nesta etapa que os algoritmos de aprendizado de máquina são aplicados. Aprendizado de Máquina, do inglês *Machine Learning*, é uma subárea da Inteligência Artificial [Cha+17]. De acordo com [Mit97], essa subárea está preocupada com a questão de como construir programas de computador que melhoram automaticamente com a experiência. De acordo com [SB14], existem três tipos de algoritmos de aprendizado de máquina:

- **Supervisionado:** O algoritmo de aprendizado (indutor) recebe um conjunto de exemplos de treinamento para os quais os rótulos da classe associada são conhecidos. Cada exemplo (instância ou padrão) é descrito por um vetor de valores (atributos) e pelo rótulo da classe associada. O objetivo do indutor é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados [WFH11];
- **Não Supervisionado :** O objetivo do aprendizado não supervisionado é aprender estruturas desconhecidas que estão presentes nos dados [HZN16]. Como exemplos de tarefas de aprendizado não supervisionado podemos citar Agrupamento, Redução de Dimensionalidade e Aprendizagem de características ou representação. Na tarefa de agrupamento, do inglês *Clustering*, o algoritmo analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou *clusters* [Llo82]. Na tarefa de Redução de Dimensionalidade, o algoritmo recebe os dados de entradas com uma alta dimensão e o transforma em um espaço de baixa dimensão, mas mantendo as propriedades significativas dos dados originais [NN19]. Na aprendizagem de características ou representação, do inglês *feature learning or representation learning*, o objetivo é aprender representações dos dados originais, que podem ser usadas para redução de dimensionalidade ou para reconstrução de um padrão. Um exemplo de *feature learning* é o autoencoder, que é uma rede neural com

Fonte: <https://blog.curso-r.com/posts/2017-06-26-construindo-autoencoders>

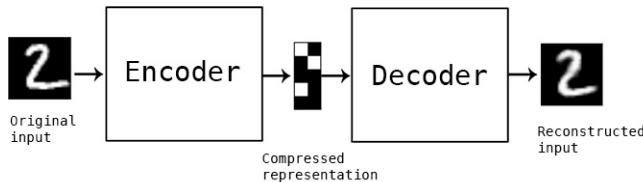


Figura 2.1: Exemplo de funcionamento de um autoencoder

objetivo de definir os valores alvo iguais às entradas. Em outras palavras, ele está tentando aprender uma aproximação para a função identidade, de modo a produzir a saída semelhante a entrada¹. A Figura 2.1 ilustra o funcionamento de um Autoencoder;

- **Por reforço:** O aprendizado por reforço pode ser classificado como uma técnica de aprendizado intermediária, entre supervisionado e não supervisionado [SB14]. Ela difere do aprendizado não supervisionado, no objetivo, pois no aprendizado por reforço existe um comportamento que o programa / “agente inteligente” deseja aprender. Por outro lado, ela difere do aprendizado supervisionado porque não é possível ter uma base de treinamento disponível com o rótulo que se deseja para cada cenário. Existe apenas os dados de entrada. E como o programa / agente aprende se não temos uma base de dados rotulada? O aprendizado ocorre através da interação entre o agente e o ambiente, o seu comportamento será “modelado” a partir de recompensas e punições a medida que o agente mapeia as percepções que ele tem do ambiente para uma determinada ação. Por exemplo: em um jogo de pingue e pongue, um agente pode utilizar cada ponto que faz como uma recompensa e cada ponto que perde como uma punição e a partir dos estados do ambiente em cada uma dessas ações, modelar seu comportamento. De acordo com [Rus+96], esse tipo de aprendizagem deve ser utilizado quando o programa / agente aprende o que fazer na ausência de exemplos rotulados.

Este capítulo aborda apenas o aprendizado supervisionado. No entanto, ele não entra em detalhes de demonstrações matemáticas de seus algoritmos. Por isso, ele deve ser utilizado como uma leitura introdutória sobre o tema, e o leitor é aconselhado a buscar mais detalhes em outras referências.

2.1 Aprendizado Indutivo

Antes de nos aprofundarmos no aprendizado supervisionado, é importante termos uma boa definição de Aprendizado de Máquina ou Aprendizado Indutivo. No entanto, você sabe a definição de indução? De acordo com [LI11], um processo de raciocínio indutivo de maneira geral, é um raciocínio que parte do particular para o geral, da parte para o todo.

Para um melhor entendimento do processo de indução, vamos utilizar um exemplo de um argumento indutivo. Lembra o que é um argumento? Um argumento é uma sequência de premissas seguido por uma conclusão!

Exemplo de argumento indutivo:

Premissas:

- Todos os apartamentos do primeiro andar do edifício Residenciais dos Corais tem dois quartos;
- Todos os apartamentos do segundo andar do edifício Residenciais dos Corais tem dois quartos;

¹<http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/>

- Todos os apartamentos do terceiro andar do edifício Residenciais dos Corais tem dois quartos;
- O edifício Residenciais dos Corais possui quatro andares.

Conclusão:

- Todos os apartamentos do edifício Residenciais dos Corais tem dois quartos.

A conclusão deste argumento não tem todas as provas nas suas premissas, pois não temos a informação dos apartamentos do quarto andar, mas ele propicia uma boa aproximação! Desta forma, ele é um argumento indutivo. Se a última premissa fosse “O edifício Residenciais dos Corais possui três andares”, ele seria um argumento dedutivo!

2.2 Aprendizado Supervisionado

Agora que temos uma definição de indução, vamos melhorar a nossa definição de aprendizado supervisionado. O objetivo do aprendizado supervisionado é encontrar uma função $h(x)$ que concorde com um conjunto de exemplos. O algoritmo de aprendizado supervisionado recebe como entrada o valor correto de uma função desconhecida $f(x)$ para entradas específicas e tenta recuperar a função [Rus+96], ou seja, dada uma coleção de exemplos de $f(x)$, o objetivo do aprendizado indutivo é retornar uma função $h(x)$ que se aproxime de $f(x)$. A Figura 2.2 ilustra esse processo. O nome Aprendizado Supervisionado vem do fato de o valor desejado para cada entrada ser fornecido, “emulando” o papel de um supervisor / professor que informa o que é esperado para cada entrada. Desta forma, para que seja realizado o aprendizado supervisionado é necessário uma base de treinamento formada por pares $(x, f(x))$, onde x representa as entradas para uma tarefa e $f(x)$ o rótulo (valor desejado de $f(x)$ para o valor de x). A Figura 2.3 ilustra um exemplo de uma função $f(x)$ e três funções $h(x)$. Entre as três opções de aproximação exibidas na Figura 2.3 qual é a melhor? A função $h(x)$ que o algoritmo de aprendizado indutivo recuperar é considerada “satisfatória” se ela tiver um bom poder de generalização! Mas o que é isso? É capacidade de a função $h(x)$ classificar novos exemplos corretamente, ou seja, exemplos diferentes dos que foram utilizados para recuperar a função $h(x)$.

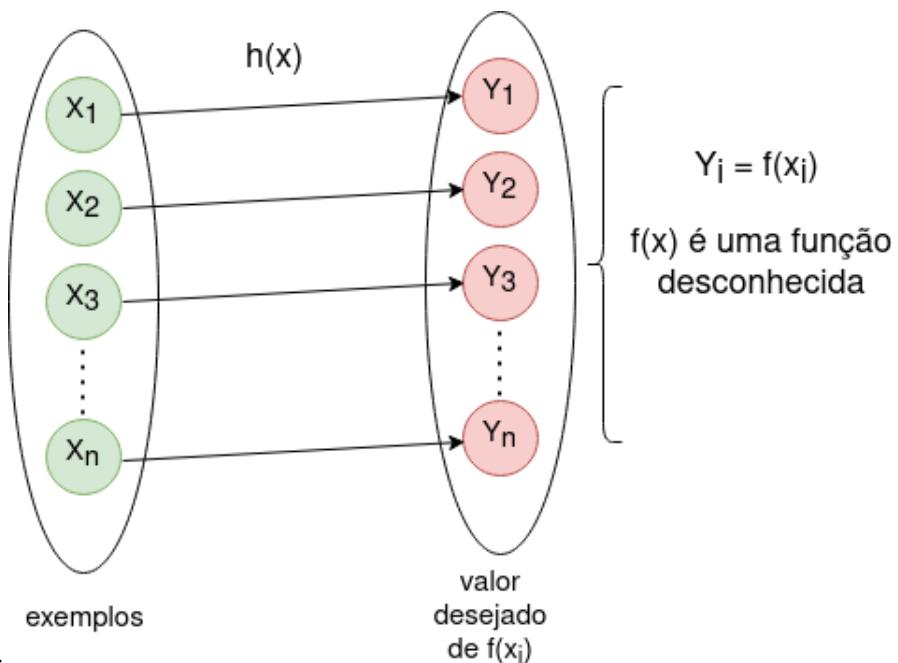
Outra definição de aprendizado indutivo muito aceita na literatura é a de [Mit97]: “Um programa aprende a partir da experiência E , em relação a uma classe de tarefas T , com medida de desempenho P , se seu desempenho em T , medido por P , melhora com E ”. Para um melhor entendimento desta definição, vamos instanciar os conceitos de experiência E , tarefas T e medida de desempenho P para dois exemplos.

Exemplo 1 - Detecção de bons clientes para um cartão de crédito

- Tarefa T : Classificar potenciais novos clientes como bons ou maus pagadores;
- Medida de Desempenho P : Porcentagem de clientes classificados corretamente;
- Experiência de Treinamento E : uma base de dados histórica em que os clientes já conhecidos são previamente classificados como bons ou maus pagadores.

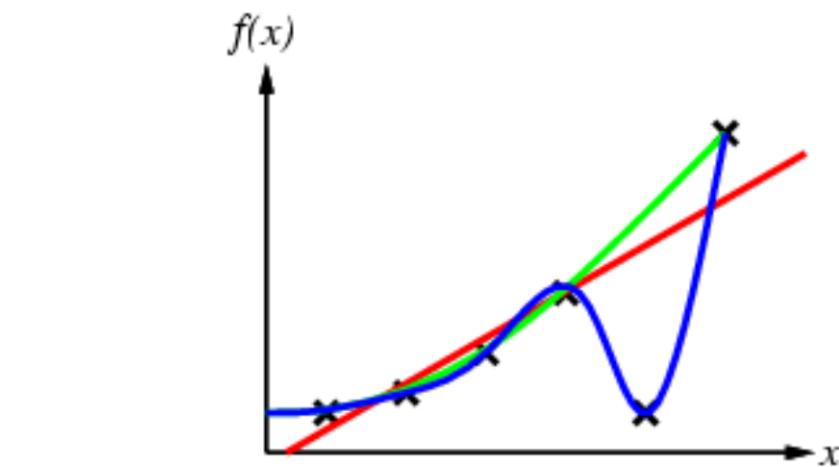
Exemplo 2 - Diagnóstico de gravidez de risco

- Tarefa T : Classificar novas gestantes com potenciais riscos na gravidez;
- Medida de Desempenho P : Porcentagem de pacientes classificadas corretamente;
- Experiência de Treinamento E : Base de dados histórica contendo exemplos de gestantes com ou sem gravidez de risco.



Fonte: O autor

Figura 2.2: Processo de aprendizado supervisionado.



Fonte: [Rus+96]

Figura 2.3: Exemplo de três funções $h(x)$ que tentam se aproximar de uma função $f(x)$. A função em vermelho está sendo representada por um modelo linear. A função em verde representa um boa aproximação. A função em azul memorizou todas as entradas.

Fonte: O autor

Sexo	Idade	Renda	Estado Civil	Classe
Masculino	18	R\$ 1.000,00	Solteiro	MAU
Feminino	21	R\$ 800,00	Solteiro	MAU
Masculino	50	R\$ 2.000,00	Casado	BOM
Feminino	60	R\$ 8.000,00	Casado	BOM

Tabela 2.1: Amostra de um conjunto de exemplos

2.2.1 Conceitos Importantes

Conforme visto anteriormente, para utilização de um algoritmo de aprendizado supervisionado é necessário uma base de treinamento e existem alguns conceitos básicos associados à base de treinamento que serão listados a seguir:

- **Exemplo:** também chamado de registro ou padrão é uma tupla de valores de atributos. Como exemplo podemos citar um cliente, um paciente, um carro e etc.
- **Atributo:** descreve uma característica ou um aspecto de um exemplo. Ele pode ser classificado como **Nominal** quando assume valores categóricos como, por exemplo: sexo e estado civil. Ou pode ser classificado como **Contínuo** quando assume valores numéricos como, por exemplo: peso, altura e idade.
- **Classe:** atributo especial também denominado rótulo. Ele é utilizado no aprendizado supervisionado para emular a figura do professor externo, que informa o valor que deseja ser aprendido para todas as variáveis de entrada correspondente aquele registro.
- **Conjunto de exemplos:** Um conjunto de exemplos é composto por exemplos contendo valores de atributos bem como a classe associada. Existem alguns nomes específicos para conjuntos de exemplos dependendo de sua finalidade: treinamento, validação ou teste como será visto no capítulo de Avaliação de Desempenho. A Tabela 2.1 ilustra uma amostra de um conjunto de exemplos.

2.3 Paradigmas de Aprendizado Supervisionado

Existe uma grande quantidade de algoritmos de Aprendizado Supervisionado na literatura. [NXC09] indicaram na literatura mais de 30 algoritmos utilizados em artigos científicos em sistemas de relacionamento com cliente. No entanto, de acordo com [Dom18], existem quatro paradigmas de aprendizado supervisionado:

- Por analogia;
- Baseado em regras;
- Conexionista;
- Bayesiano.

Cada um destes paradigmas será definido a seguir e um algoritmo de cada paradigma será detalhado.

2.3.1 Por analogia

Uma forma de classificar um novo padrão é lembrar-se de exemplos parecidos classificados anteriormente, e assim atribuir ao novo exemplo uma classe de um padrão parecido. Esta é a ideia central deste paradigma. Os algoritmos que seguem esse paradigma também são chamados de classificador baseado em instâncias, quando aplicados em problemas de classificação. Dentre os algoritmos deste

Fonte: Adaptado do material da Intel Desenvolvedor IA

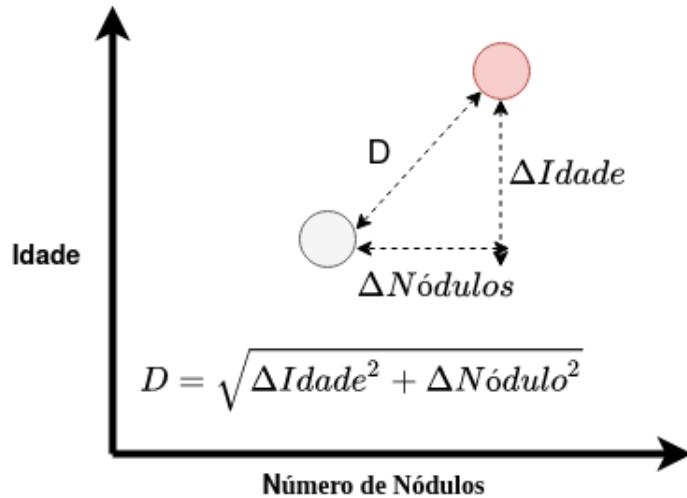


Figura 2.4: Exemplo de cálculo da similaridade entre padrões no KNN usando a distância euclidiana.

paradigma, o mais popular é o k-vizinhos mais próximos, do inglês *k-Nearest Neighbors* (KNN), que será descrito a seguir.

k-Nearest Neighbors

O KNN é um classificador baseado em instâncias, ou seja, o modelo é representado pelos exemplos de treinamento [CH06]. O valor K representa a quantidade de exemplos semelhantes do conjunto de treinamento utilizado para classificação. Dado um exemplo desconhecido, um classificador KNN procura os K exemplos semelhantes ao padrão desconhecido para realizar a classificação. O exemplo desconhecido é associado à classe que contém mais registros semelhantes. A semelhança entre exemplos é medida em termos de métricas de similaridade. Uma destas métricas é a distância euclidiana. Para um melhor entendimento deste conceito, vamos supor que estamos trabalhando em um problema de classificação de pacientes saudáveis ou não com base em duas variáveis de entrada: idade e número de nódulos. Assumindo que o novo padrão que desejamos classificar seja representado na Figura 2.4 com a cor vermelha, qual seria a sua similaridade com o padrão de treinamento representado pela cor cinza? Uma alternativa seria utilizar a distância euclidiana, que mede a distância entre dois pontos. Como estamos trabalhando com duas variáveis, a distância euclidiana pode ser obtida pela aplicação direta do teorema de Pitágoras, onde a distância é a hipotenusa e as diferenças entre as duas variáveis são os catetos.

A Figura 2.5² ilustra o fluxograma do algoritmo KNN. Primeiro, é definido o parâmetro K, depois será calculada a distância entre o padrão que pretendemos classificar e toda base de dados, esta etapa faz com que o algoritmo seja lento para conjuntos de treinamento com muitos registros. Em seguida, os registros são ordenados com base na similaridade e são avaliados apenas os K exemplos mais semelhantes. Por fim, é definida a classe do novo padrão a partir do voto majoritário dos K vizinhos mais próximos. A classe que tiver a maioria vence. Se tivermos trabalhando com um

²<https://software.intel.com/content/www/us/en/develop/training/course-machine-learning.html>

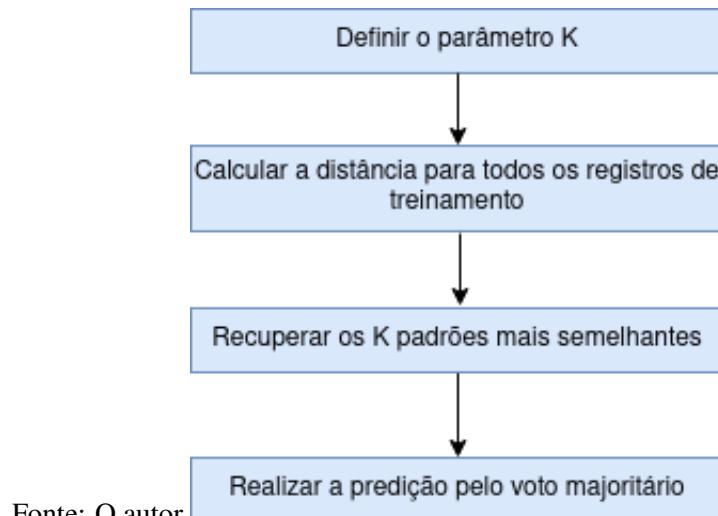


Figura 2.5: Fluxograma do algoritmo KNN.

problema de regressão a saída será a média dos K vizinhos mais próximos.

O KNN apresenta como vantagem sua simplicidade e facilidade de implementação. No entanto, ele é extremamente sensível a tributos irrelevantes e redundantes, uma vez que todos os atributos contribuem de forma igualitária para a classificação através do cálculo da distância [Bey+99; FFW11].

2.3.2 Baseado em regras

O paradigma baseado em regras contempla uma grande variedade de algoritmos que representam o modelo no formato de regras. Em geral, estas regras estão na forma de “Se...então”, o que torna fácil a interpretação deste tipo de modelo [OSC13]. Por isso, eles são muito utilizados na área de aprendizagem de máquina. Dentre os modelos baseados em regras, um dos mais populares são as árvores de decisão que serão descritas a seguir.

Árvore de Decisão

Os modelos de árvore de decisão representam sua hipótese através de uma estrutura de dados de uma árvore. A Figura 2.6 ilustra uma árvore de decisão que aprendeu a função do conectivo lógico *AND*. Cada nó da árvore representa uma variável de entrada e as arestas de cada nó representam os possíveis valores de cada variável. Os nós folhas (que não tem nós filhos) representam a saída da árvore. Uma vez que uma árvore tenha sido induzida, “construída”, é fácil extrair as regras do tipo “Se então”, pois cada caminho da árvore representa uma condição “variável valor”. A Figura 2.7 exibe o funcionamento geral do algoritmo de Árvore de Decisão.

O passo 1 do algoritmo informa que deve ser selecionado a variável que melhor separa os exemplos para criação de um nó da árvore, mas como vamos saber qual a melhor? A escolha da variável deve minimizar a profundidade da árvore. Por isso, dois pontos devem ser levados em consideração: 1) escolher um atributo que consiga ir o mais longe possível na classificação exata de exemplos e 2) um atributo perfeito divide os exemplos em conjuntos que são todos positivos ou todos negativos (em problemas de classificação binária onde a variável alvo têm dois valores). No entanto, para realizar essa avaliação de forma sistemática será necessário usar uma métrica para

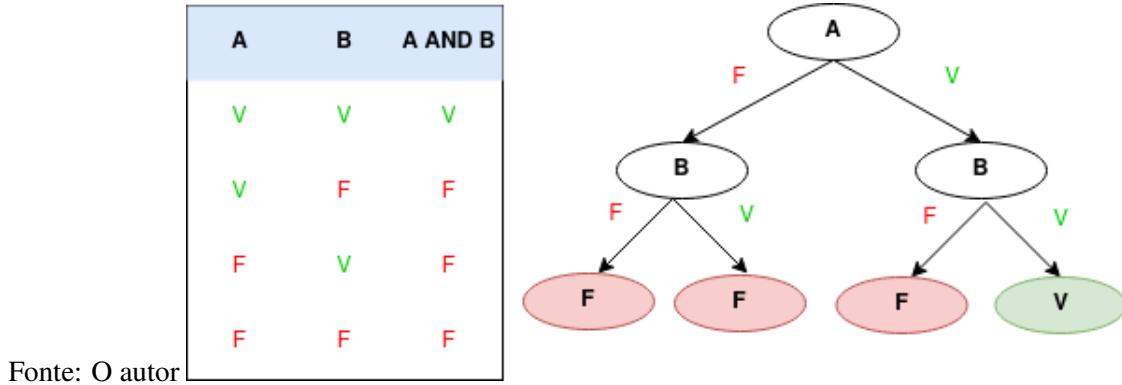


Figura 2.6: Exemplo de uma Árvore de Decisão.

Algorithm 1: Árvore de Decisão

Input: Tabela com todos os exemplos da base de treinamento**Output:** Estrutura de dados de uma árvore.

- 1: Seleciona a variável que melhor separa os exemplos para criação de um nó da árvore;
 - 2: **For all** valor da variável selecionada anteriormente **do**
 - 3: Criar uma aresta;
 - 4: Filtrar os exemplos para cada filho levando em conta o valor da aresta;
 - 5: Repetir o procedimento a partir do passo 1 para cada nó filho não “puro” levando em consideração os exemplos filtrados. Um nó filho é puro quando o atributo alvo da base filtrada pelo valor da aresta tem apenas um valor em todos os exemplos.
 - 6: **End For**
-

Figura 2.7: Funcionamento geral do algoritmo de árvore de decisão

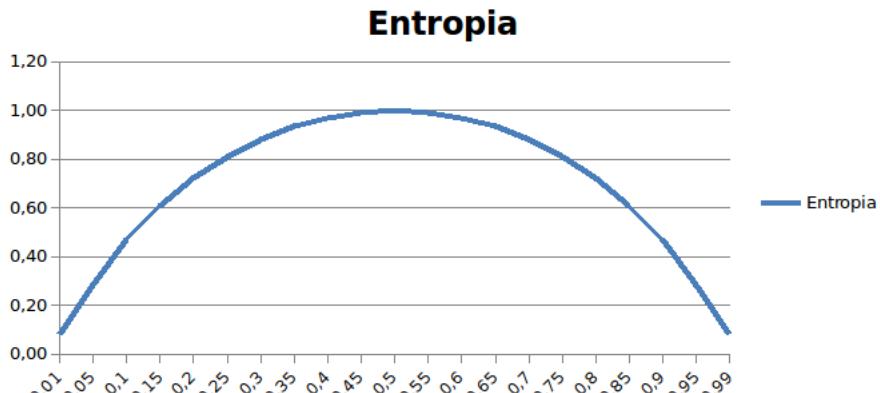
Classe_1	Classe_2
1	99
10	90
20	80
30	70
40	60
50	50
60	40
70	30
80	20
90	10
99	1

Tabela 2.2: Variação da classe alvo em dez bases de dados

avaliar cada variável. As métricas utilizadas para selecionar o melhor atributo muitas vezes são baseadas no grau de impureza dos dados. Quanto maior o grau de impureza, mais distorcida é a distribuição das classes [ORS19]. Uma das métricas mais utilizadas para definir a impureza dos dados é a entropia [TSK05]. A entropia é uma grandeza que mede a desordem, tanto de objetos físicos quanto de informações. Quanto maior o grau da entropia maior é a desordem e, quanto menor o grau da entropia melhor a organização. A equação 2.1 mostra o cálculo da entropia para uma base de treinamento. Onde y representa os rótulos (valores) da variável alvo Y e $p(y)$ representa a probabilidade a priori. Para um melhor entendimento deste cálculo, vamos utilizar a Tabela 2.2 como exemplo. Esta tabela simula 11 bases de dados com diferentes distribuições entre os valores da variável alvo. Cada base de dados contém dois valores para a classe alvo: Classe_1 e Classe_2. Desta forma, a primeira linha representa uma base de dados com a seguinte distribuição de variável alvo: Classe_1 = 1 e Classe_2 = 99. A segunda linha representa uma base de dados com a seguinte distribuição de variável alvo: Classe_1 = 10 e Classe_2 = 90. Cada linha vai alternando em 10 registros as distribuições destas classes. A Figura 2.8 exibe o gráfico com o valor da entropia para cada base de dados. Conforme pode ser observado na figura, o valor máximo da entropia ocorre quando a distribuição das classes são equivalentes (50 exemplos para cada classe), ou seja, quando existe a maior desordem na base de dados (estão todos “misturados”). Por outro lado, os menores valores de entropia ocorrem quando as bases estão mais “puras” (99 exemplos de uma classe e 1 exemplo de outra).

$$\text{Entropy } H(Y) = - \sum p(y) \log p(y) \quad (2.1)$$

Agora que definirmos o que é entropia. Vamos voltar a pergunta anterior: “Como escolher o melhor atributo?”. A resposta não é a entropia! A resposta é escolher o atributo que consegue deixar a base de dados mais pura! Para isso, existem algumas métricas que fazem isso utilizando a entropia. Dentro as mais conhecidas está o Ganhão de Informação, que é a redução esperada da entropia da base devido a “classificação” de acordo com um determinado atributo de entrada [WFH11]. A equação 2.2 exibe o cálculo do Ganhão de Informação para uma variável, onde $H(X|Y)$ representa o valor da



Fonte: O autor

Figura 2.8: Gráfico da variação dos valores da entropia com relação a mudança na distribuição da variável alvo.

entropia da variável alvo Y apenas considerando a variável X.

$$\text{Ganho de Informação } GI(X,Y) = H(Y) - H(X|Y) \quad (2.2)$$

Para um maior entendimento deste algoritmo, é importante que você tente construir uma árvore de decisão com os dados da Tabela 2.3. Esta tabela representa uma base de dados com informações sobre quando jogar tênis, a partir de informações do tempo [Rus+96]. Desta forma, a variável alvo é o atributo *play* e as variáveis de entrada que devem ser usadas para construção dos nós da árvore são: *outlook*, *temperature*, *humidity* e *windy*. Uma planilha com o passo a passo da resolução deste exercício está disponível no Github do livro³, assim como exemplos de cálculo da Entropia e Ganho de Informação.

2.3.3 Conexionalista

O paradigma conexionalista é baseado na área de pesquisa de Redes Neurais Artificiais (RNA) [RM86]. Uma rede neural artificial é um modelo computacional inspirado no funcionamento do cérebro humano [BJ90]. As redes neurais possuem como principal característica aprender através de exemplos e poder de generalização. Dizer que um modelo possui poder de generalização significa dizer que ele é capaz de prever corretamente exemplos que ele nunca viu. Dentre as redes neurais artificiais, certamente as mais populares são as redes Multi Layer Perceptron (MLP) [Hay98], que serão descritas a seguir. Uma RNA do tipo MLP possui três componentes principais: unidade de processamento “os neurônios”, as conexões “sinapses” e uma topologia.

Neurônios

A ideia do neurônio artificial é simular o funcionamento do neurônio biológico. De uma forma extremamente simplificada, podemos dizer que a comunicação entre os neurônios biológicos ocorrem a partir de impulsos elétricos. Cada neurônio recebe uma determinada quantidade de estímulo de outros neurônios e a partir de uma determinada quantidade de estímulo recebido, o neurônio “dispara”

³<https://github.com/osalvone/>

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Tabela 2.3: Base de dados com informações sobre quando jogar tênis

um novo estímulo para outro neurônio. Desta forma, o neurônio artificial possui o seguinte objetivo: receber entradas de conjunto de unidades A, computar função sobre entradas e enviar resultado para conjunto de unidades B. Um neurônio artificial possui duas propriedades: estado de ativação e função de ativação. Cada uma destas propriedades serão descritas a seguir.

- **Estado de ativação:** Este atributo define o valor recebido pelo neurônio. Ele envolve a soma ponderada das conexões de entradas (w_i) com o valor do neurônio correspondente;
- **Função de ativação:** Esse método é responsável por gerar o valor de saída do neurônio a partir do seu estado de ativação. Essa função seria equivalente ao “disparo” do neurônio biológico. Para neurônios de redes MLP, a função de ativação mais popular é a Sigmoide Logística.

Conexões

As conexões definem como os neurônios estão interligados. Cada conexão possui um peso, um valor de ponderamento associada, por esse motivo é dito na literatura que as conexões armazenam o conhecimento da rede MLP [BG92].

Topologia

A topologia de uma rede MLP define como os neurônios estão distribuídos e conectados. Os neurônios são distribuídos em camadas dentro de uma Rede. Cada Camada possui uma funcionalidade. Existem três tipos de camadas: Entrada, Intermediária ou oculta e Saída.

- **Entrada:** a camada de entrada é responsável por receber a informação do mundo externo. Desta forma, para cada variável de entrada existe um neurônio na camada de entrada. É importante destacar que nenhum cálculo é realizado na função de ativação deste neurônio. Eles apenas enviam as informações para os neurônios da camada oculta.
- **Intermediária ou Oculta:** Os neurônios das camadas intermediárias ou escondidas não possuem conexão com o mundo exterior, por isso o nome oculta. Elas recebem os dados da camada de entrada, realizam alguns processamentos e enviam os dados transformados para a camada de saída. Mas qual é o papel deste tipo de camada para as redes do tipo MLP?

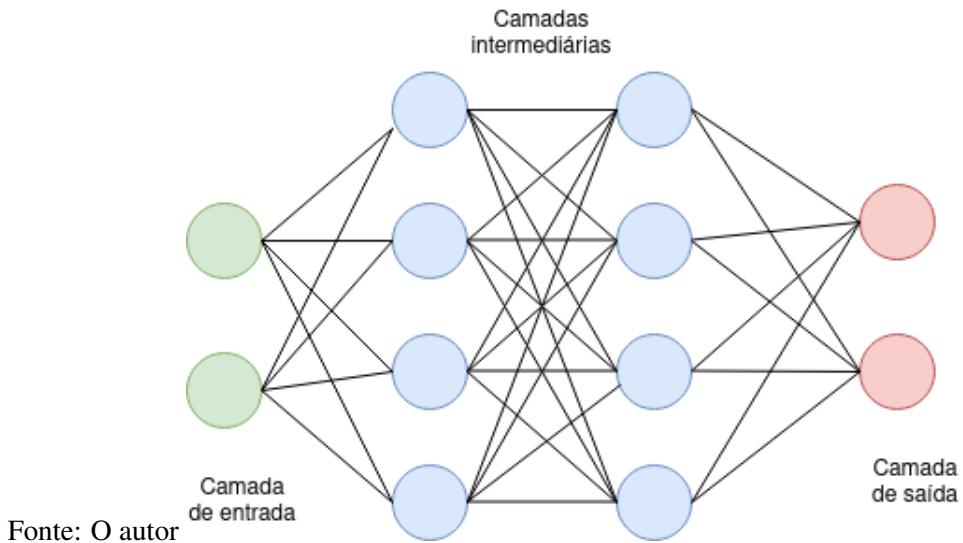


Figura 2.9: Topologia típica de uma rede MLP.

O principal papel deste tipo de camada é transformar a representação original dos dados de entradas em uma representação mais simples de serem resolvidas pela camada de saída. Por exemplo, as redes sem camadas intermediárias só conseguem resolver problemas linearmente separáveis, que são aqueles que podem separar as classes através de uma reta. Desta forma, para problemas não linearmente separável é necessário a utilização de camadas ocultas, que vão transformar o problema não linearmente separável recebido da camada de entrada em um problema linearmente separável para que a camada de saída consiga aprender. Atualmente, ainda não existe na literatura uma definição de quantos neurônios devem existir na camada oculta e nem quantas camadas ocultas são necessárias. Essa especificação depende de cada problema, embora existam alguns estudos que utilizam algoritmos de busca para obter esses parâmetros.

- **Saída:** A camada de saída é responsável por enviar a informação da rede para o mundo externo. Desta forma, para problemas de classificação, o número de neurônios na camada de saída é igual ao número de rótulos da variável alvo.

Redes MLP típicas são formadas por uma camada de entrada, uma camada de saída e n camadas escondidas (dispostas entre as duas primeiras), onde todos os neurônios de uma camada são completamente conectados com todos os neurônios da camada precedente, conforme Figura 2.9.

O método de treinamento mais popular para redes do tipo MLP é o algoritmo de retro-propagação do erro, do inglês *error backpropagation algorithm* [RM86] e seus variantes [BG92]. O algoritmo de retro-propagação do erro é baseado na regra delta proposta por [WH88]. E é por este motivo que muitos o conhecem também como regra delta generalizada.

A retro-propagação é constituída por computações em duas distintas direções ao longo da estrutura da rede neural: na direção *forward* e na direção *backward*. Na passagem *forward*, as saídas para um padrão específico de entrada são calculadas e o erro das unidades de saída é determinado [BJ90]. Na passagem *backward*, os valores das unidades de saída são usados para realizar os ajustes nos valores dos pesos de forma proporcional ao erro observado.

Uma passagem *forward* e uma passagem *backward* são realizadas para cada par de entradas e

Algorithm 2: Treinamento de uma rede MLP

```

1: Iniciar todas as conexões com valor zero;
2: For all par de treinamento ( $X, d$ ) do
3:   Calcular a saída da rede ( $y$ );
4:   Se  $y$  é diferente de  $d$  então
5:     Atualizar os pesos das conexões;
6:   Se atingiu algum critério de parada então
7:     Interromper o treinamento;
8: End For

```

Figura 2.10: Algoritmo geral de treinamento de uma rede MLP

saída de cada exemplo. É exatamente a combinação destas duas passagens para todos os exemplos que constitui um ciclo de treinamento. Muitos ciclos são normalmente necessários para treinar uma rede MLP. O algoritmo geral de treinamento é exibido na Figura 2.10, onde o padrão de treinamento é representado por X e a saída desejada é representada por d .

Critérios de parada

Em geral, os três critérios de parada mais conhecidos para o treinamento de uma rede MLP são:

- **Número de épocas:** uma época equivale a um ciclo, que é a apresentação de todos os registros de treinamento à rede para ajuste dos pesos. O número de épocas como critério de parada é utilizado para evitar que o treinamento entre em um *loop* infinito;
- **Erro aceitável:** este critério de parada é usado quando se conhece a priori qual o erro relativo ao problema que está se tentando resolver. No entanto, em problemas do mundo real dificilmente esse valor é conhecido;
- **Perda de generalização, do inglês *Generalization Loss* (GL):** este é um critério de parada que monitora quando a rede está memorizando ou decorando o conjunto de treinamento, e interrompe o treinamento neste momento para evitar que a rede perca o seu poder de generalização. Esse fenômeno de perca do seu poder de generalização da rede é conhecida na literatura como *overfitting* [Oli08]. Uma das formas para utilizar esse critério de parada é dividindo o conjunto de treinamento em dois conjuntos disjunto: Treinamento e Validação. O conjunto de treinamento será usado para ajustar os pesos da rede neural. No entanto, o conjunto de validação não será utilizado para ajustar os pesos, mas será usado para monitorar a perca de generalização. Esse critério de parada funciona da seguinte forma, para cada época é calculado o erro da rede nos dois conjuntos de Treinamento e Validação. A perca de generalização ocorrerá quando o erro no conjunto de treinamento estiver diminuindo com o aumento do número de épocas, mas o erro no conjunto de validação aumentar por uma quantidade consecutiva de épocas. Esse processo é ilustrado na Figura 2.11, onde o erro no conjunto de treinamento é ilustrado pela linha verde e o erro no conjunto de validação é representado pela linha vermelha. É importante destacar que o GL não é o único critério de parada para evitar o *overfitting*. Existem outros critérios como, por exemplo, as técnicas de regularização [TA77].

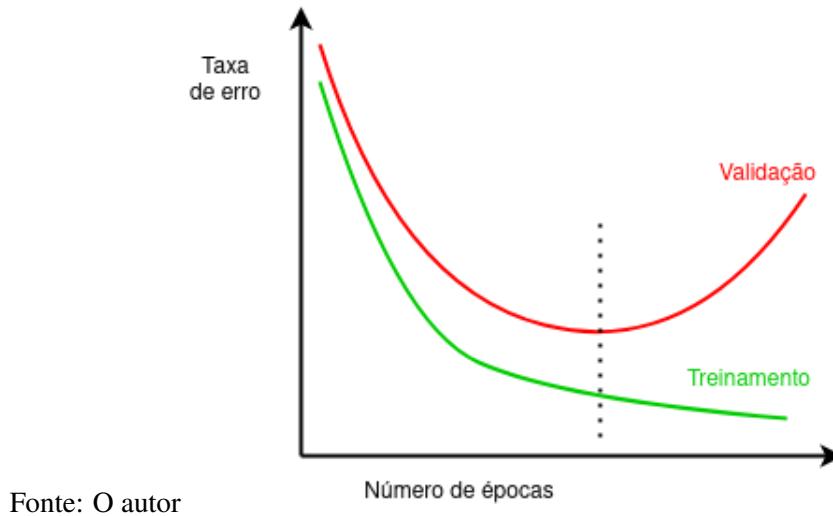


Figura 2.11: Processo de perda de generalização durante o treinamento de uma rede MLP.

2.3.4 Atualização dos pesos da rede MLP

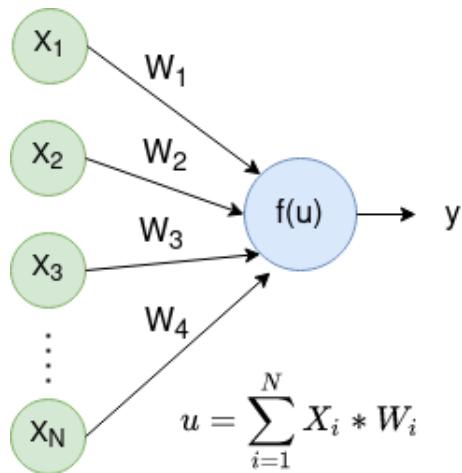
Como informado anteriormente, o algoritmo mais popular de treinamento de redes MLP é o *backpropagation*, que é baseado na regra delta proposta por [WH88]. O treinamento para redes do tipo MLP consiste em encontrar os valores dos pesos das conexões. No entanto, antes de mostrar a dedução matemática para chegar na fórmula da regra delta e sua generalização para redes de múltiplas camadas, será fornecida uma breve contextualização histórica.

O primeiro neurônio

O primeiro neurônio artificial foi proposto por [MP43] e ficou conhecido como neurônio McP, uma referência aos sobrenomes dos autores (**M**cCulloch e **P**itts). Os autores simularam o comportamento de “disparo” de um neurônio biológico quando a quantidade de impulsos ultrapassa um limiar. Essa representação foi implementada através da aplicação de uma função de ativação no estado de ativação (u) do neurônio artificial. O estado do neurônio McP é a soma ponderada das saídas dos neurônios anteriores (X_i) e sua respectiva conexão (W_i). A Figura 2.12 ilustra este mecanismo. A verificação de disparo é realizada através da função de ativação $f(u)$ do neurônio artificial. A função de ativação do neurônio McP é a função degrau que retorna 1 caso a soma ponderada ultrapasse um limiar, do inglês *threshold*, caso contrário retorna zero conforme ilustrado na Figura 2.13. Você pode estar se perguntando por que falar do neurônio McP em um resumo sobre redes MLP? Os estudos do neurônio McP tinha como objetivo a modelagem do neurônio biológico e sua capacidade computacional de execução de funções booleanas [BLC00]. A parte de treinamento de redes neurais surgiu depois com as redes de uma camada Perceptron e Adaline [BLC00] que usam o neurônio McP.

Influência da rede Perceptron

Qual a influência da rede Perceptron na MLP? A MLP é uma rede Perceptron com várias camadas de neurônios! E com a inclusão do algoritmo de treinamento *backpropagation* que será visto logo mais. A rede Perceptron de uma única camada de neurônios McP foi proposta por [Ros58]. Uma das contribuições desta rede foi a inclusão do valor de *threshold* do McP no treinamento como um parâmetro livre. A condição de disparo para o neurônio McP é que o valor de ativação $f(u)$ seja maior



Fonte: O autor

Figura 2.12: Estado do neurônio McP.

$$Y = \begin{cases} 1, & se f(u) \geq \theta \\ 0, & se f(u) < \theta \end{cases}$$

Fonte: O autor

Figura 2.13: Função degrau.

$$\left\{ \begin{array}{l} W_1 * X_1 + W_2 * X_2 >= \theta \\ W_1 * X_1 + W_2 * X_2 - \theta >= 0 \\ W_1 * X_1 + W_2 * X_2 + W_0 * X_0 >= 0 \\ \dots \dots \downarrow \\ -\theta = X_0 = -1 \end{array} \right.$$

Fonte: O autor

Figura 2.14: Demonstração do procedimento para considerar *threshold* como um parâmetro livre.

do que um *threshold*. No entanto, não sabemos qual o valor deste *threshold a priori*. A estratégia utilizada para resolver esse problema foi considerar o *threshold* como uma entrada (X_0) com valor fixo igual a menos um (-1) e atribuir um peso (W_0) a essa entrada. Desta forma, o *threshold* também será “aprendido”. A Figura 2.14 mostra esse procedimento considerando duas entradas X_1 e X_2 .

Influência do rede Adaline

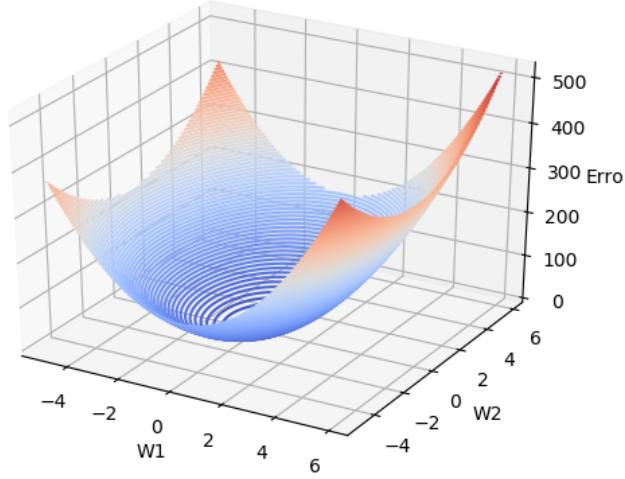
A rede de uma camada Adaline foi proposta por [WH60]. A principal contribuição da rede Adaline para a rede MLP foi a regra delta, uma vez que a rede MLP usa uma generalização da regra delta como será visto mais adiante. Mas o que é a regra delta? Ela representa a regra de atualização dos pesos durante o treinamento de uma rede. Uma vez que o treinamento consiste em encontrar o conjunto de pesos (W) das conexões que minimizem a função erro, o treinamento da rede pode ser redefinido como um problema de busca. Para a rede Adaline, o erro da rede para um padrão X é dada por $E = 1/2 * (Y - O)^2$, onde Y representa a saída desejada e O a saída calculada pela rede. Desta forma, para um conjunto de treinamento de tamanho N, a função de erro é representada pela equação 2.3.

$$Erro = \left(\frac{1}{2} \right) \sum_{i=1}^n (y_i - o_i)^2 \quad (2.3)$$

A função erro definida anteriormente é uma função quadrática. A Figura 2.15 exibe o gráfico em três dimensões da função erro para uma rede com apenas um neurônio e duas conexões. O código para geração deste gráfico está disponível no Github do Livro ⁴. As dimensões do gráfico são: o valor do erro e o peso das conexões W_1 e W_2 . A regra delta busca minimizar a função erro através da utilização do gradiente descendente, ou seja, os ajustes dos pesos são dados na direção contrária do vetor gradiente. O vetor gradiente armazena todas as informações das derivadas parciais de uma função multivariável [SGR11]. Por exemplo, se o vetor gradiente for avaliado em um ponto específico, ele apontará para a direção onde a subida é mais íngreme. Desta forma, se andarmos na direção contrária do gradiente, estaremos indo na direção de mínimo da função.

A regra delta é dada por $W_{i+1} = W_i - \Delta W$, onde W_i e W_{i+1} representam o peso da conexão antes e depois da atualização respectivamente. O ΔW representa o valor da atualização do peso e é obtido a partir da aplicação do gradiente descendente na função de erro da rede em relação ao peso. A

⁴<https://github.com/osalvoneito/>



Fonte: O autor.

Figura 2.15: Gráfico em três dimensões da função erro.

equação 2.4. mostra como calcular ΔW_{ik} , que representa a atualização do peso entre os neurônios i e k.

$$\Delta W_{ik} = \alpha * e_k * x_i \quad (2.4)$$

Onde

- α é a taxa de aprendizado, que representa uma taxa de proporcionalidade de quanto rápido será realizado o ajuste no peso da conexão. Essa taxa é um número contínuo que varia entre zero e um;
- e_k representa o erro, que é a diferença entre o valor desejado e o calculado para o neurônio k;
- x_i representa o valor calculado para o neurônio i.

Dedução da regra delta

É importante destacar que o nome do algoritmo proposto por [WH60] foi o *Least Mean Square* (LMS) e não regra delta, que se tornou popular posteriormente. Você pode estar se perguntando onde está o N do tamanho da amostra na fórmula de atualização da regra delta? Essa foi a principal contribuição de Widrow e Hoff, eles propuseram uma adaptação da aplicação do gradiente descendente na formula do erro [Wid05]. A adaptação proposta foi utilizar o gradiente instantâneo (o erro relativo apenas a um registro na fórmula do erro) ao invés do gradiente verdadeiro que leva em consideração todo os padrões no cálculo do erro. É importante destacar que é por esse motivo que o LMS é conhecido na literatura como uma versão estocástica do gradiente descendente [Spa03]. Essa adaptação tornou a atualização dos pesos mais simples e rápida. É importante lembrar que o gradiente descendente é um método iterativo que busca minimizar uma função a partir do deslocamento **contrário** do vetor gradiente. A seguir é especificada a sequência de etapas na dedução da regra delta. A equação 2.5 exibe a fórmula geral da regra delta. A equação 2.6 mostra a adaptação da função custo, que

foi considerar o erro com relação a apenas um registro. As equações 2.7, 2.8 e 2.9 demonstram a aplicação da regra da cadeia no cálculo das derivadas parciais (vetor gradiente) na nova função de custo, erro E.

$$W_{i+1} = W_i - \alpha * \nabla W_i \quad (2.5)$$

$$Erro = \left(\frac{1}{2}\right) \sum_{i=1}^n (y_i - o_i)^2 \Rightarrow E = \left(\frac{1}{2}\right) (y_i - o_i)^2 \quad (2.6)$$

$$\nabla W_i = \frac{\partial E}{\partial w_i} \quad (2.7)$$

$$\frac{\partial E}{\partial w_i} = 2 * \left(\frac{1}{2}\right) (y_i - o_i) * -\frac{\partial o_i}{\partial w_i} \quad (2.8)$$

$$\frac{\partial E}{\partial w_i} = -(y_i - o_i) * \frac{\partial o_i}{\partial w_i} \quad (2.9)$$

Uma vez que o_i é uma função composta por $g(h(x))$, onde g é a função de ativação e h é a função que define o valor do neurônio, então a derivada parcial da equação 2.9 pode ser redefinida pela equação 2.10 aplicando a regra da cadeia. Atualizando a equação 2.9 com 2.10 obtemos a equação 2.11. Uma vez que $h(x) = \sum_{j=1}^n (x_j * w_j)$, a derivada de $h(x)$ em relação a um peso específico (w_i), ou seja quando $i = j$, será $\frac{\partial h}{\partial w_i} = x_i$. Atualizando a equação 2.11 com essa informação obtemos a equação 2.12. Uma vez que $(y_i - o_i)$ representa o erro (e) podemos atualizar 2.12 e obter a equação 2.13. Uma vez que a função de ativação g no neurônio Adaline é a função identidade então $\frac{\partial o_i}{\partial g} = 1$, obtemos a equação 2.14.

$$\frac{\partial o_i}{\partial w_i} = \frac{\partial o_i}{\partial g} * \frac{\partial h}{\partial w_i} \quad (2.10)$$

$$\frac{\partial E}{\partial w_i} = -(y_i - o_i) * \frac{\partial o_i}{\partial g} * \frac{\partial h}{\partial w_i} \quad (2.11)$$

$$\frac{\partial E}{\partial w_i} = -(y_i - o_i) * \frac{\partial o_i}{\partial g} * x_i \quad (2.12)$$

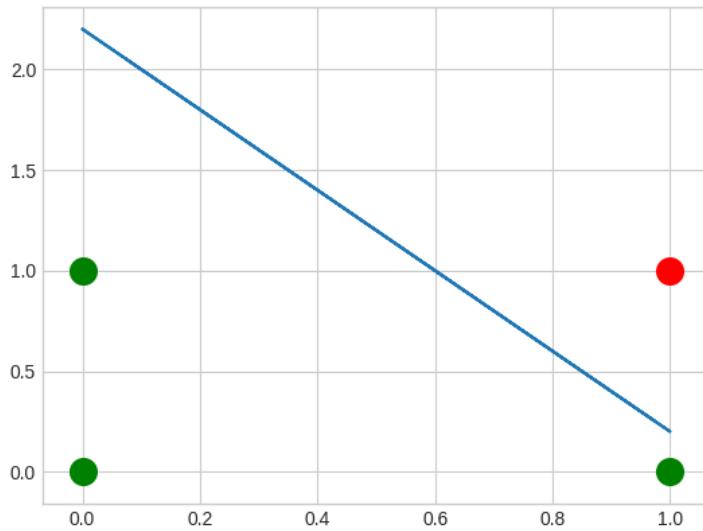
$$\frac{\partial E}{\partial w_i} = -e * \frac{\partial o_i}{\partial g} * x_i \quad (2.13)$$

$$\frac{\partial E}{\partial w_i} = -e * x_i \quad (2.14)$$

Para chegarmos na fórmula final da regra delta partimos de $W_{i+1} = W_i - \Delta W$, uma vez que ΔW é a equação 2.7 e esta pode ser reescrita pela equação 2.14, obtemos $W_{i+1} = W_i - (-e * x_i)$, e se adicionarmos uma taxa de proporcionalidade chegamos na fórmula final da regra delta como queríamos demonstrar $W_{i+1} = W_i + \alpha * e * x_i$.

X1	X2	AND
1	1	1
1	0	0
0	1	0
0	0	0

Tabela 2.4: Tabela verdade para a função AND



Fonte: O autor.

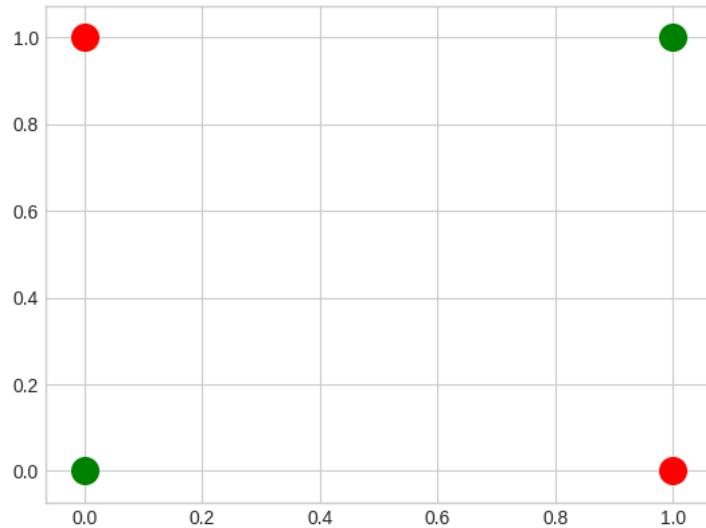
Figura 2.16: Superfície de decisão do Perceptron para a função AND.

Limitações do Perceptron

[MP69] provaram que um único Perceptron não era capaz de resolver problemas não linearmente separáveis. Essa descoberta contribuiu para uma diminuição significativa das pesquisas na área de Redes Neurais Artificiais na década de 70. Problemas linearmente separáveis são aqueles em que a separação entre as classes pode ser realizada por meio de uma reta. Para um melhor entendimento deste conceito, vamos utilizar como exemplo a função lógica AND que é linearmente separável. A Tabela 2.4 mostra os possíveis valores da função AND. Um Perceptron consegue aprender a função AND. A Figura 2.16 ilustra a reta de separação dos padrões feita pelo Perceptron após o treinamento. [MP69] mostram que um Perceptron não consegue aprender a função XOR (ou exclusivo), que não é linearmente separável, como pode ser observado na Figura 2.17. A Tabela 2.5 mostra os possíveis valores da função XOR. Para as figuras 2.16 e 2.17, os pontos em vermelho representam a classe 1 e os pontos em verde representam a classe 0.

Backpropagation

Para superar a limitação das redes de uma única camada como o Perceptron, surgiram as redes com múltiplas camadas, conforme mencionado no início da seção sobre redes MLP. O papel das camadas intermediárias neste tipo de rede é realizar transformações no espaço de entrada original,

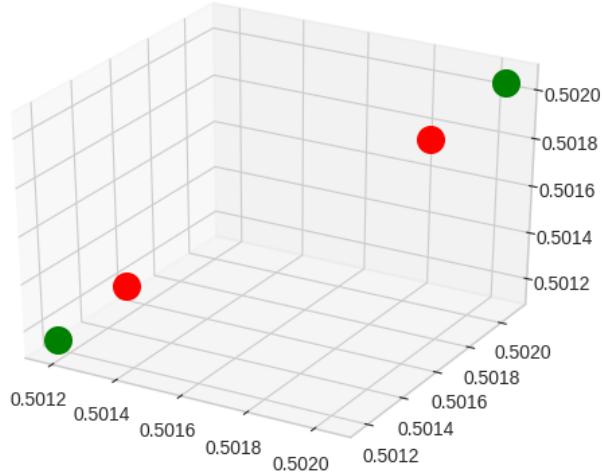


Fonte: O autor.

Figura 2.17: Distribuição das classes para a função XOR.

X1	X2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

Tabela 2.5: Tabela verdade para a função XOR

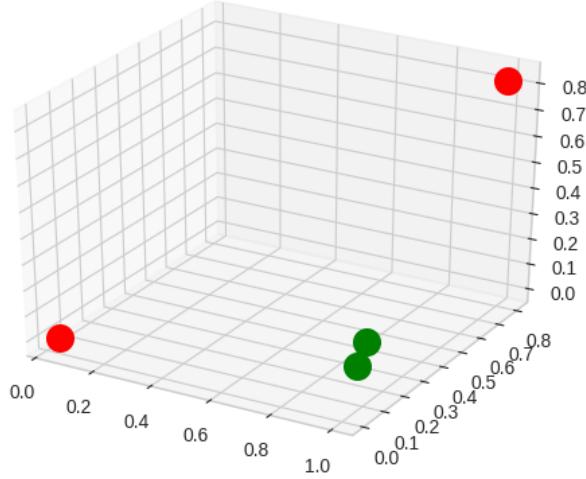


Fonte: O autor.

Figura 2.18: Gráfico em 3D com as saídas dos três neurônios da camada oculta para a função XOR **no início** do treinamento da rede.

transformando os dados em uma representação mais simples de ser resolvida pela camada posterior. Para ilustrar este conceito, vamos utilizar novamente o problema XOR. Para resolver este problema vamos usar como solução uma rede MLP com três camadas: 1) uma camada de entrada com dois neurônios de entrada, 2) uma camada oculta com três neurônios e 3) uma camada de saída com um neurônio. A Figura 2.18 ilustra a saída da camada oculta para os quatro padrões de entrada no **início do treinamento**. Cada ponto no gráfico representa um padrão de entrada e a cor de cada ponto representa a classe a qual ele pertence. É importante destacar que a representação original recebida pela rede através da camada de entrada é a Figura 2.17. Depois o espaço é transformado pela camada oculta na Figura 2.18 no **início do treinamento**. Vale destacar também que o espaço de entrada que a camada de saída recebe é a saída da camada oculta. A Figura 2.19 ilustra a saída da camada oculta para os quatro padrões de entrada no **final do treinamento**. Como pode ser observado, o espaço de entrada gerado pela camada oculta tornou mais simples a separação das classes pela camada de saída. Para que exista essa transformação no espaço de entrada, a função de ativação dos neurônios da camada escondida deve ser não linear, caso contrário a combinação de funções lineares é uma função linear. Para este exemplo, foi utilizada a função logística como função de ativação.

A grande questão para treinar as redes de multicamadas é: **como realizar a correção dos pesos dos neurônios da camada escondida?** Note que o erro da camada de saída é obtido de forma direta pela diferença entre o valor desejado e o calculado pelo neurônio da camada de saída. Para responder a essa pergunta, o algoritmo de treinamento chamado de *backpropagation* foi proposto. Este algoritmo é uma generalização da regra delta em que o ajuste do peso ΔW é distinto para conexões de neurônios das camadas de saída e oculta. Antes de mostrar as deduções para ajuste dos pesos pelo *backpropagation*, é importante dizer que ele foi o responsável pelo ressurgimento de maior interesse na área de RNA. O trabalho de [RM86] foi responsável pela popularização do *backpropagation* tendo em vista as mais de 8000 citações deste trabalho até maio de 2020. No



Fonte: O autor.

Figura 2.19: Gráfico em 3D com as saídas dos três neurônios da camada oculta para a função XOR **após** o treinamento da rede.

entanto, de acordo com o Instituto de Engenharia Elétrica e Eletrônica (IEEE)⁵, o *backpropagation* foi criado por Paul J. Werbos em sua tese de doutorado [Wer74].

Dedução dos ajustes dos pesos da Rede Multilayer Perceptron

As redes Multilayer Perceptron (MLP) utilizam o algoritmo *backpropagation* para ajuste dos pesos. Existem várias funções de ativação utilizadas em redes MLP, no entanto, a mais popular é a Sigmoide Logística, que é uma versão diferenciável da função degrau [Ahn13]. A Figura 2.20 ilustra o formato desta função. Para as deduções das fórmulas de ajustes dos pesos com *backpropagation*, será utilizada a função Sigmoide Logística.

A nomenclatura para neurônios e pesos utilizados nas próximas deduções seguirá o padrão adotado na Figura 2.21, onde W_{ij} representa o peso da conexão entre os neurônios i e j . Adaptando a equação 2.11 a nomenclatura apresentada na Figura 2.21, obtemos a equação 2.15. Lembrando que Y é a saída desejada.

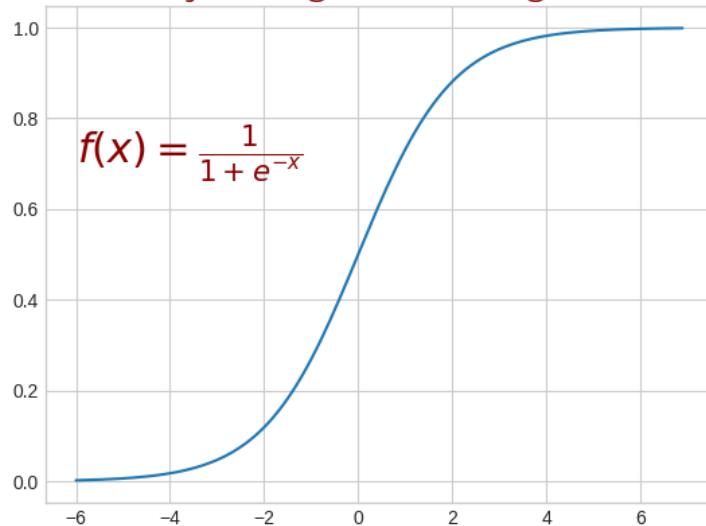
$$\frac{\partial E}{\partial w_{ij}} = -(y_j - o_j) * \frac{\partial o_j}{\partial g_j} * \frac{\partial g_j}{\partial w_{ij}} \quad (2.15)$$

A correção dos pesos na camada de saída é feita de forma direta resolvendo a derivada parcial $\frac{\partial o_j}{\partial g_j}$ na equação 2.15. Lembrando que a função de ativação g_j é a logística. A fórmula fica de acordo com a equação 2.16.

$$\frac{\partial E}{\partial w_{ij}} = -(y_j - o_j) * (o_j) * (1 - o_j) * \frac{\partial h_j}{\partial w_{ij}} \quad (2.16)$$

⁵<https://ieeexplore.ieee.org/author/37344537300>

Função Sísmoide Logística



Fonte: O autor.

Figura 2.20: Gráfico da função Sísmoide Logística.

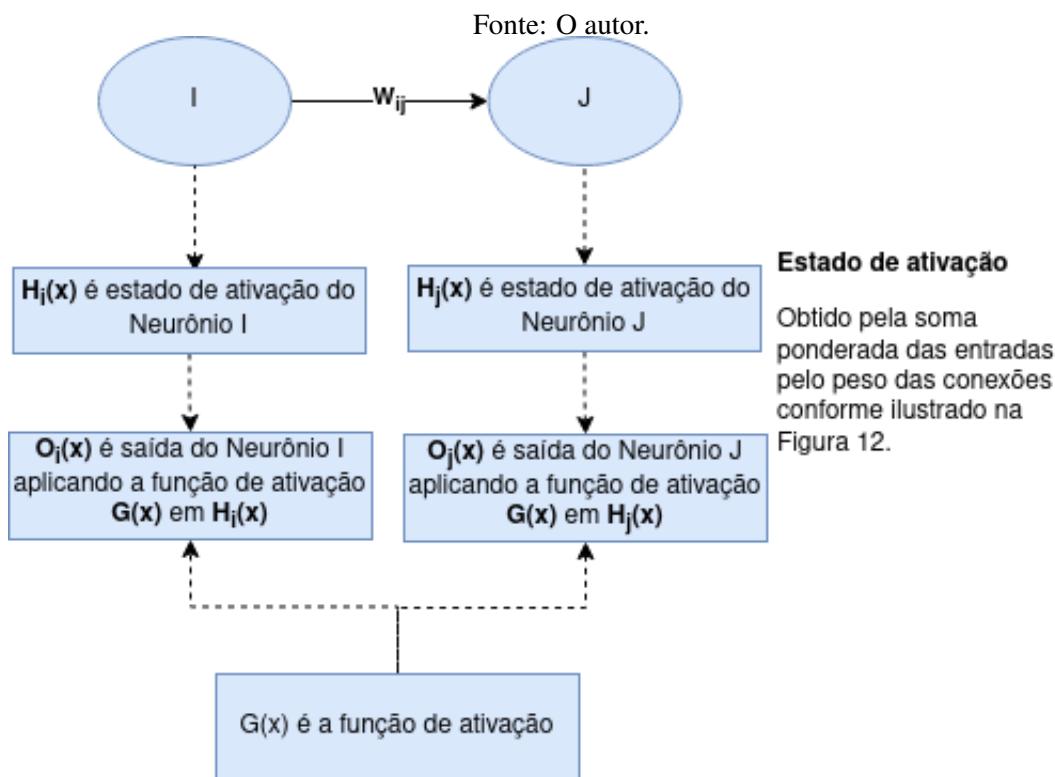


Figura 2.21: Padrões de nomenclatura para as deduções do *backpropagation*.

Uma vez que $h_j(x) = \sum_{i=1}^n (o_i * w_{ij})$, onde N é o número de conexões de entrada do neurônio j. Considerando que $\frac{\partial h_j}{\partial w_{ij}} = o_i$ e que $(y_j - o_j)$ representa o erro e_j , obtemos a equação 2.17.

$$\frac{\partial E}{\partial w_{ij}} = -(e_j) * (o_j) * (1 - o_j) * o_i \quad (2.17)$$

Atualizando a regra delta $W_{i+1} = W_i - \Delta W$ para os pesos dos neurônios da camada de saída da rede MLP obtemos a equação 2.18, uma vez que $\Delta W = \frac{\partial E}{\partial w_{ij}}$. Lembrando que α é a aprendizagem.

$$w_{ij(t+1)} = w_{ij(t)} + \alpha * (e_j) * (o_j) * (1 - o_j) * o_i \quad (2.18)$$

Nós vimos como atualizar os pesos das conexões dos neurônios da camada de saída, mas como aplicar a mesma fórmula para os neurônios da camada oculta? A grande questão para isso é: **como será calculado o erro do neurônio da camada oculta?** uma vez que no padrão de treinamento não existe o valor desejado para o neurônio da camada oculta, existe apenas o valor desejado do neurônio de saída! A resposta para essa pergunta é que o erro do neurônio da camada oculta será a soma dos erros dos neurônios da camada de saída ponderado (multiplicado) pelo peso da conexão. Para um melhor entendimento deste conceito, a Figura 2.22 ilustra o cálculo do erro para neurônios da camada oculta. Desta forma, a fórmula para atualização dos pesos dos neurônios da camada oculta é exibida na equação 2.19.

$$w_{ij(t+1)} = w_{ij(t)} + \alpha * \left(\sum_{i=1}^n (W_j l_i * E_i) \right) * (o_j) * (1 - o_j) * o_i \quad (2.19)$$

Derivada da Função Logística

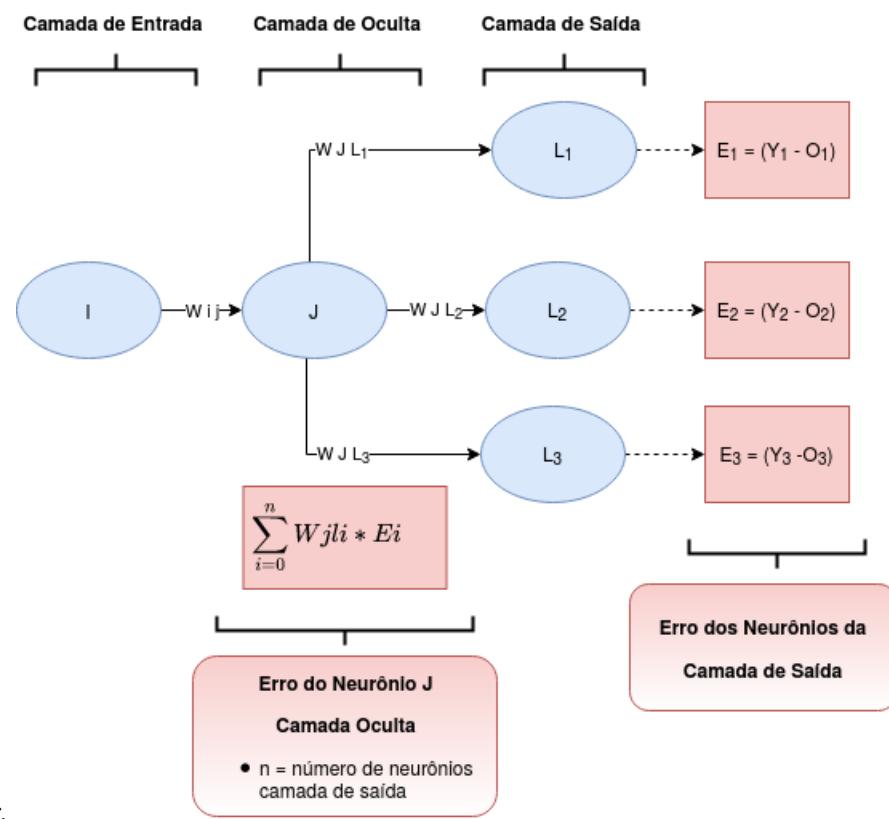
O desenvolvimento da derivada da função logística será mostrado a seguir. O objetivo é fornecer ao leitor um entendimento mais profundo das demonstrações anteriores, uma vez que a regra delta generalizada utiliza a derivada da função logística, pois essa foi a função de ativação aplicada ao neurônio.

$$f(x) = \frac{1}{1 + e^{-x}} \Rightarrow f(x) = (1 + e^{-x})^{-1} \quad (2.20)$$

$$f'(x) = -(1 + e^{-x})^{-2} * \frac{d}{dx}(1 + e^{-x}) \quad (2.21)$$

$$f'(x) = -(1 + e^{-x})^{-2} * e^{-x} * -1 \Rightarrow (1 + e^{-x})^{-2} * e^{-x} \quad (2.22)$$

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} \Rightarrow \frac{1}{(1 + e^{-x})} * \frac{e^{-x}}{(1 + e^{-x})} \quad (2.23)$$



Fonte: O autor.

Figura 2.22: Ilustração do processo de cálculo do erro para neurônios da camada oculta.

$$f'(x) = f(x) * \frac{e^{-x}}{(1+e^{-x})} \Rightarrow f(x) * \frac{(e^{-x} + 1 - 1)}{(1+e^{-x})} \quad (2.24)$$

$$f'(x) = f(x) * \frac{(1+e^{-x} - 1)}{(1+e^{-x})} \quad (2.25)$$

$$f'(x) = f(x) * \left(\frac{(1+e^{-x})}{(1+e^{-x})} - \frac{1}{(1+e^{-x})} \right) \quad (2.26)$$

$$f'(x) = f(x) * \left(1 - \frac{1}{(1+e^{-x})} \right) \quad (2.27)$$

$$f'(x) = f(x) * (1 - f(x)) \quad (2.28)$$

2.3.5 Bayesiano

De acordo com [Ber11], o paradigma Bayesiano usa a probabilidade como medida racional para lidar com a incerteza na tarefa de prever valores futuros. Neste paradigma, os parâmetros do modelo são expressos na forma de distribuições de probabilidade [And11]. Dentre os modelos Baysianos, o mais conhecido é o Naive Bayes.

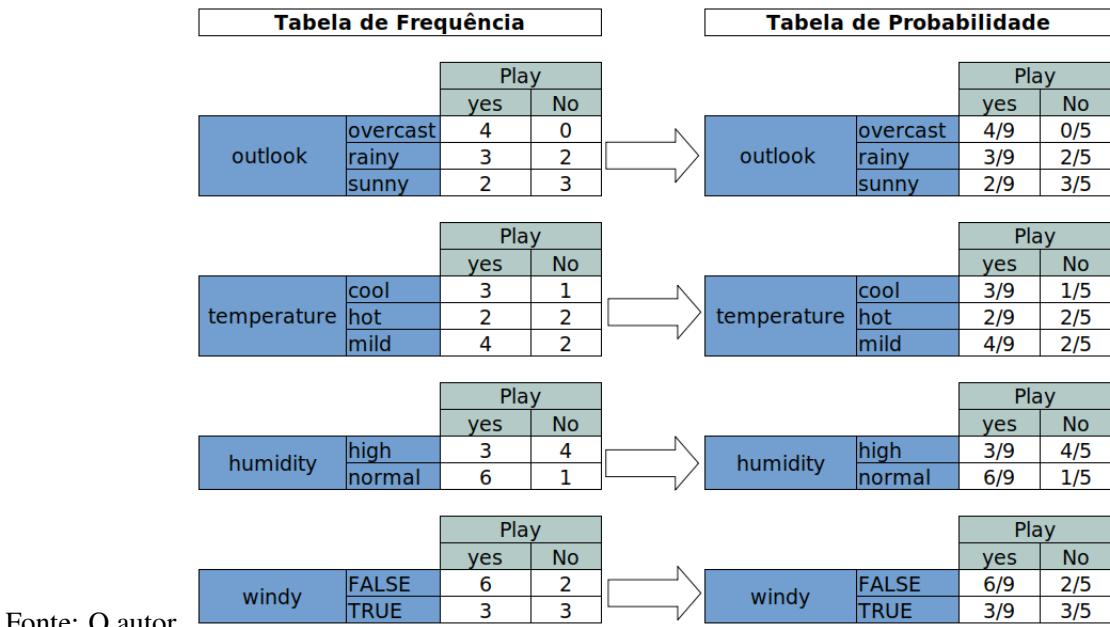
Naive Bayes

O classificador Naive Bayes é o mais simples dos modelos Baysianos [ZS04]. Ele é baseado no teorema de Bayes que é descrito na equação 2.29. Onde x e y são eventos, e p(x) e p(y) representam a probabilidade a priori dos respectivos eventos. P(x/y) representa a probabilidade a posteriori de x dado que ocorreu y. P(y / x) representa a probabilidade a posteriori de y dado que ocorreu x. A probabilidade a posteriori também é conhecida como probabilidade condicional. O teorema de Bayes também é conhecido na literatura com a nomenclatura apresentada na equação 2.30.

$$p(y/x) = \frac{p(x/y) * p(y)}{p(x)} \quad (2.29)$$

$$\text{probabilidade a posteriori} = \frac{\text{verosimilhança X probabilidade a priori}}{\text{evidência}} \quad (2.30)$$

Para um maior entendimento, vamos construir uma solução usando o classificador Naive Bayes para auxiliar na tomada de decisão, se devemos jogar tênis com base no histórico de eventos passados. Nós utilizaremos os mesmos dados da Tabela 2.3. Existem quatro variáveis de entrada para um padrão X: *outlook* (X_1), *temperature* (X_2), *humidity* (X_3) e *windy* (X_4). E existe uma variável alvo: *play* (Y). O processo de construção do classificador consiste em duas etapas: 1) calcular as probabilidades a priori de Y e 2) calcular as probabilidades condicionais de X em relação a Y. As probabilidades a



Fonte: O autor.

Figura 2.23: Probabilidades condicionais de X em relação a Y.

priori de Y são $p(\text{yes}) = 9 / 14$ e $p(\text{no}) = 5 / 14$. A Figura 2.23 mostra as probabilidades condicionais de X em relação a Y.

Para classificar um novo padrão, será necessário calcular a probabilidade a posteriori dele para cada uma das classes. A classificação será atribuída para a classe que possuir maior probabilidade a posteriori. Nós temos apenas duas classes para este exemplo: YES e NO. Você pode estar se perguntando: por que o classificador não se chama Teorema de Bayes ao invés de Naive Bayes? A resposta é que ele faz uma adaptação simples na regra do Teorema de Bayes. Na verdade, essa adaptação é uma suposição ingênuo de que as variáveis de entradas são independentes, o que permite simplificar muito o cálculo das probabilidades condicionais [Bou08]. A palavra *Naive* em inglês significa Ingênuo. Uma vez que o denominador “evidência” é igual para o cálculo da probabilidade a posteriori de todas as classes, ele pode ser eliminado da fórmula. A Figura 2.24 ilustra o algoritmo para classificação de um novo padrão e a Figura 2.25 mostra um exemplo de uso.

2.4 Ensemble Models

Além dos paradigmas de aprendizagem de máquinas citados, uma abordagem muito popular é a combinação de modelos, chamado no inglês de *Ensemble Models* [MO18]. As estratégias para combinar modelos têm sido investigadas por muitos pesquisadores. As mais populares são *Bagging* [Bre96], *Boosting* [FS96] e *Stacking* [Wol92]. As duas primeiras possuem duas características em comum: 1) utilizam o mesmo algoritmo de aprendizagem de máquina para os classificadores individuais e 2) usam a estratégia de voto para combinar os classificadores. *Bagging* é um acrônimo para *bootstrap aggregating*. A sua ideia central é construir vários classificadores individuais a partir de amostras *bootstrap* (amostra com reposição do mesmo tamanho do conjunto de treinamento, em que cada exemplo tem a mesma chance de ser escolhido). O objetivo é reduzir a variância do erro do classificador final utilizando o sistema de voto, onde cada classificador terá o mesmo peso

Algorithm 3: Naive Bayes

Input: Novo padrão X (representado por um vetor com quatro atributos: X_1, X_2, X_3 e X_4)

Output: Classe do padrão X.

Calcular as probabilidades:

- $p(\text{yes}/X)$
- $p(\text{no}/X)$

Se $p(\text{yes}/X) > p(\text{no}/X)$ **então**

X pertence a classe YES

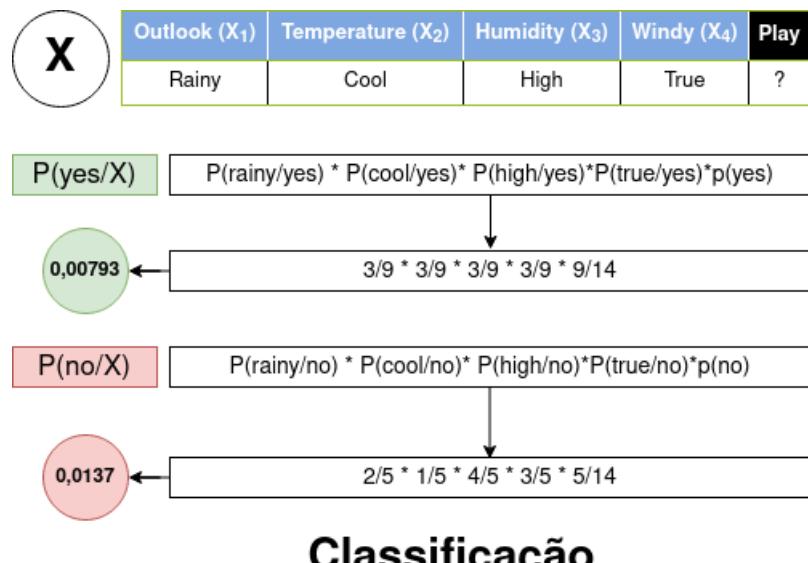
Senão

X pertence a classe NO

Onde

- $p(\text{yes}/X) = p(x_1/\text{yes}) * p(x_2/\text{yes}) * p(x_3/\text{yes}) * p(x_4/\text{yes}) * p(\text{yes})$
- $p(\text{no}/X) = p(x_1/\text{no}) * p(x_2/\text{no}) * p(x_3/\text{no}) * p(x_4/\text{no}) * p(\text{no})$

Figura 2.24: Funcionamento do algoritmo Naive Bayes para classificação de um novo padrão



X pertence a classe NO, uma vez que
 $p(\text{no}/X) > p(\text{yes}/X)$

Fonte: O autor.

Figura 2.25: Exemplo de classificação com Naive Bayes.

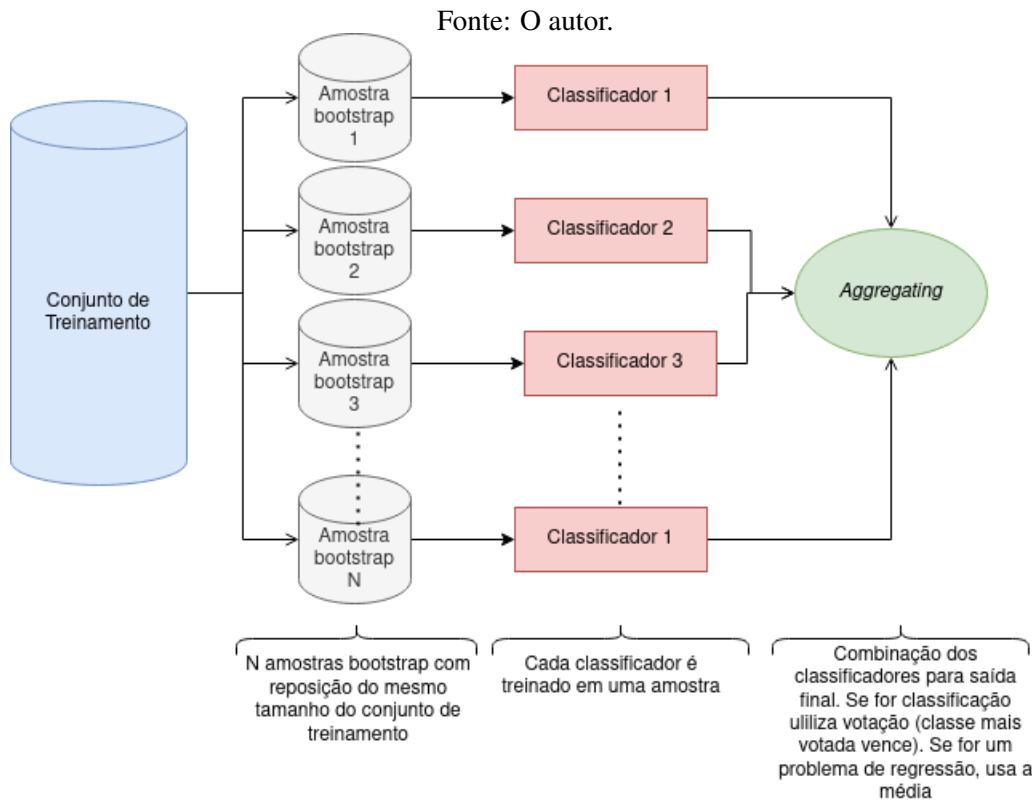


Figura 2.26: Workflow da abordagem de *Ensemble Models Bagging*.

"importância" no sistema de voto. Por outro lado, *Boosting* tem como objetivo criar classificadores individuais que sejam especialistas em determinados segmentos do conjunto da amostra original, por isso os classificadores individuais são criados de forma sequencial e o próximo classificador dará maior importância aos registros que foram classificados de forma errada pelo classificador anterior. *Boosting* atribui um peso a cada exemplo de treinamento. Este peso será usado no cálculo do erro do modelo. Desta forma, as instâncias tratadas incorretamente pelos modelos anteriores receberão um maior peso para que o modelo seguinte ofereça maior atenção a eles. *Boosting* também utiliza o sistema de voto para combinar os classificadores, porém o peso "importância" de cada classificador individual é estimado com base em seu erro. Existem diferentes algoritmos que foram inspirados em *Boosting*, o mais conhecido é o AdaBoost [Sch13]. O sistema de voto apenas faz sentido quando a performance dos classificadores são comparáveis. Por isso, o sistema de voto para combinar classificadores diferentes não é recomendado. Na estratégia de *Stacking*, o sistema de voto é substituído por um *meta-learning*, que é representado por um algoritmo de aprendizagem de máquina que utiliza como entrada as previsões dos classificadores individuais. Desta forma, *Stacking* tenta aprender a importância de cada classificador através de outro algoritmo de aprendizagem de máquina - o *meta-learning*. É importante ressaltar que a estratégia de *Stacking* larga em desvantagem em relação à estratégia de voto (*Bagging* ou *Boosting*) quando aplicados em pequenas amostras, pois para construção do *meta-learning* é necessário um conjunto independente daqueles utilizados para construção dos classificadores individuais. As Figuras 2.26, 2.27 e 2.28 ilustram o funcionamento das três abordagens de *ensemble models*.

Fonte: O autor.

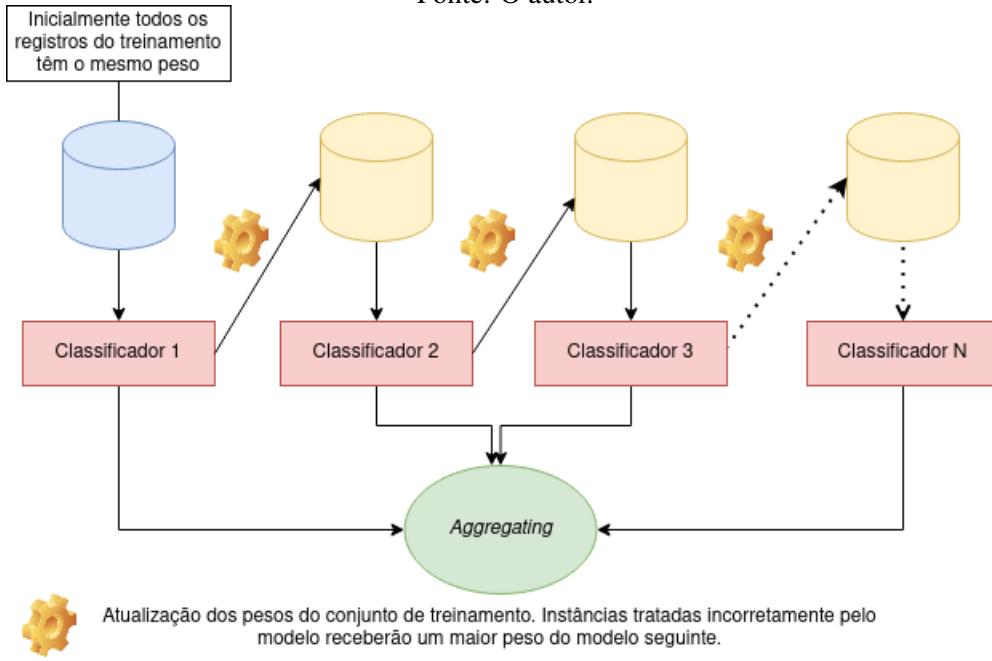
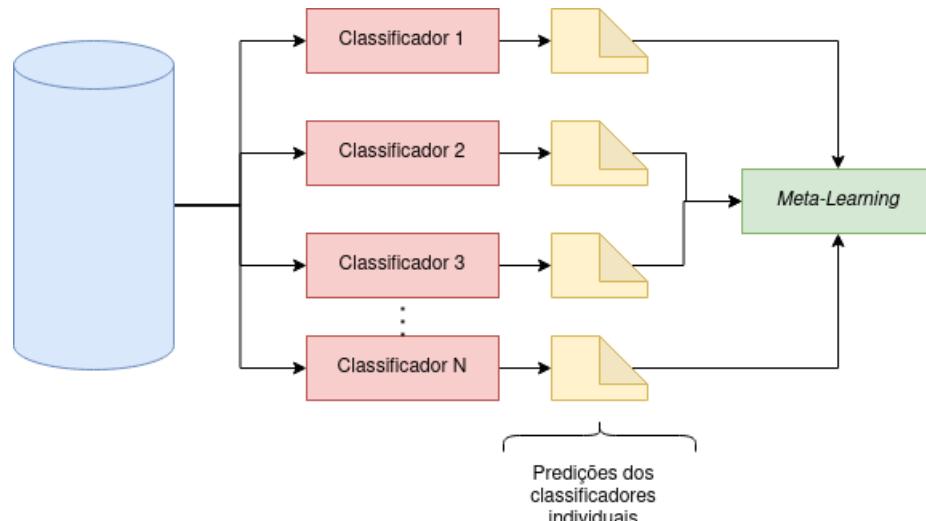


Figura 2.27: Workflow da abordagem de *Ensemble Models Boosting*.



Fonte: O autor.

Figura 2.28: Workflow da abordagem de *Ensemble Models Stacking*.

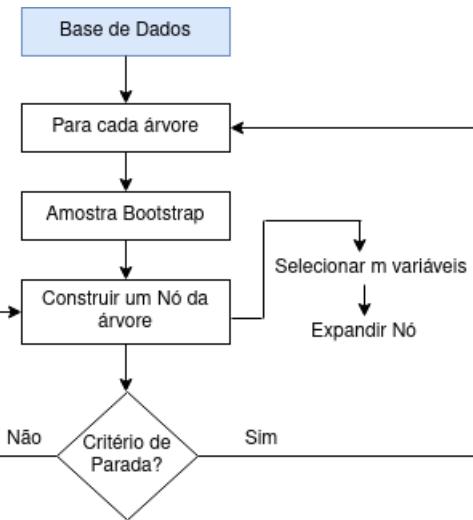
Algorithm 4: Random Forest

Input: Base de dados (B), número de árvore que compõem a floresta (n), número de variáveis de cada árvore (m)

Output: Floresta.

For i = 1 to n **Do**

- Construir uma árvore:
 - Selecionar aleatoriamente m variáveis em B para construir cada nó da árvore.
 - Adicionar a árvore à Floresta.

Figura 2.29: Algoritmo geral do *Random Forest*

Fonte: O autor.

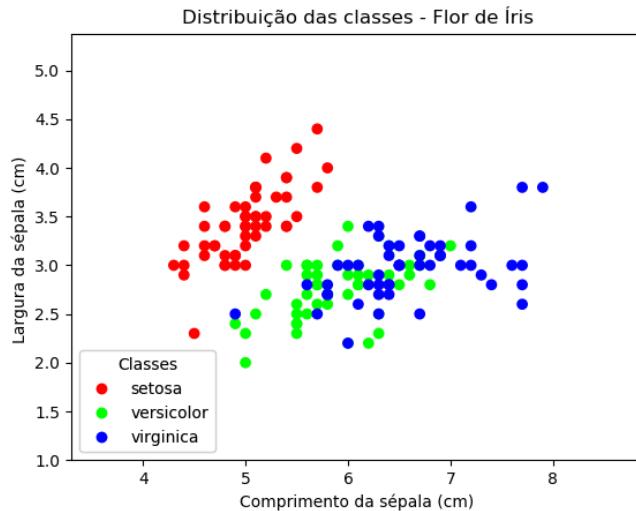
Figura 2.30: Workflow do *Random Forest*.**Random Forest**

O *Random Forest* é um classificador de aprendizagem combinada (*ensemble learning*) do tipo *bagging*⁶ proposto por [Bre01]. O *Random Forest* é um conjunto de árvores de decisão que compõe uma floresta. Este algoritmo possui dois parâmetros: *n* que representa o número de árvore que compõem a floresta e *m* que representa o número de variáveis que serão usadas para construir as árvores. A Figura 2.29 mostra a descrição do algoritmo *Random Forest* e a Figura 2.30 ilustra o seu fluxograma.

Na fase de utilização após o treinamento, cada árvore da floresta receberá o mesmo exemplo como entrada e votará em qual classe esse padrão pertence, uma vez que cada árvore gera uma saída independente, a floresta escolherá a classe com maior quantidade de votos (*aggregating*) [KHS10].

A simplicidade da árvore de decisão traz desvantagens. A principal delas é a instabilidade provocada por ruídos nos dados [HTF09]. O *Random Forest* melhora a estabilidade e precisão da árvore de decisão por incorporar um grande número de árvores em um único classificador. De acordo com [Bre01], o *Random Forest* tem um bom desempenho quando comparado com outros classificadores tradicionais, como o *Support Vector Machine* e Redes Neurais. Uma das principais

⁶Combinação de **Bootstrap** com **aggregating** proposto por [Bre96].



Fonte: O autor.

Figura 2.31: Distribuição das três classes do problema Flor de Íris em relação às variáveis de entrada comprimento e a largura da sépala (cm).

vantagens do *Random Forest* é sua facilidade de uso, uma vez que só possui dois parâmetros. [Bel08] pontua mais algumas vantagens: aprendizado direto, representação local, paralelismo e rápido tempo de treinamento.

2.5 Superfícies de Decisão

Nós vimos neste capítulo quatro paradigmas de aprendizagem de máquina, e que o objetivo de um indutor é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados [WFH11]. Uma forma didática de entender que para o mesmo problema de classificação nós podemos aplicar diferentes paradigmas e obtermos resultados semelhantes é através da visualização da Superfície de Decisão de um indutor. Uma Superfície de Decisão é uma hipersuperfície construída a partir da saída do classificador que divide o espaço das variáveis de entrada em N conjuntos subjacentes, onde N é o número de classes [Bru+20]. O indutor classificará os novos padrões de acordo com o seu posicionamento na fronteira de decisão das classes. Para um melhor entendimento deste conceito, vamos utilizar como exemplo a base de dados do problema de classificação Flor de Íris⁷. O objetivo é construir uma solução que classifique corretamente o tipo de Flor de Íris a partir de quatro variáveis de entrada: o comprimento da sépala (cm), largura da sépala (cm), comprimento da pétala (cm) e largura da pétala (cm). Existem três classes de flor de íris: Iris setosa, Iris virginica e Iris versicolor. Para facilitar a visualização, vamos considerar apenas as duas primeiras variáveis de entrada: o comprimento e a largura da sépala (cm). A Figura 2.31 exibe a distribuição das três classes em relação às duas variáveis de entrada.

Foram construídas quatro soluções para o problema de classificação Flor de Íris utilizando cada um dos paradigmas de aprendizagem de máquina visto neste capítulo. A Figura 2.32 exibe a superfície de decisão de cada uma das soluções. Como pode ser observado, a superfície de decisão

⁷https://en.wikipedia.org/wiki/Iris_flower_data_set

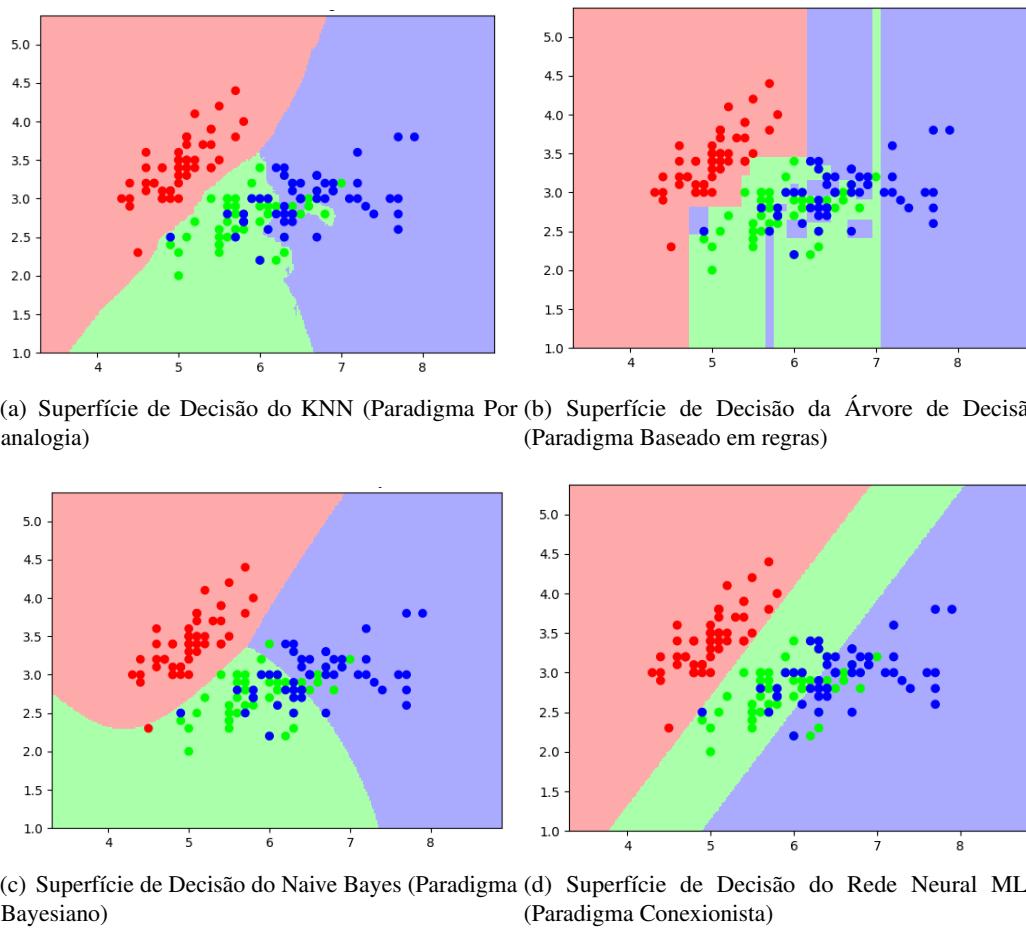


Figura 2.32: Diferentes Superfícies de Decisão para o problema Flor de Íris

delimita as regiões de fronteira entre cada uma das classes e isso pode ser feito por diferentes paradigmas. O código fonte para geração destes gráficos está disponível no Github deste livro⁸.

⁸<https://github.com/osalvonet/>



3. Pré-processamento

Este capítulo apresenta as principais atividades da etapa de pré-processamento. Essa etapa, em geral, consome entre 50% e 80% do tempo de um projeto [NAS17]. As principais tarefas realizadas nesta etapa são: Definição da Granularidade do Projeto, Tratamento de Valores Ausentes, Tratamento de Valores Aberrantes, Construção de novas variáveis e Seleção de variáveis (Redução de Dimensionalidade). Cada tarefa será detalhada a seguir.

3.1 Definição da Granularidade do Projeto

Um dos conceitos mais importante na construção de um projeto de KDD é a definição de granularidade. O conceito de granularidade está relacionado com o que se pretende decidir, ou seja, o grão da decisão. O conceito de granularidade é materializado em cada registro do arquivo que será utilizado para treinar um algoritmo de aprendizagem de máquina. Este arquivo também é chamado de *FlatFile*. Ele pode ser visto como uma matriz onde as linhas representam a granularidade do projeto (o que se pretende decidir) e as colunas representam as características do grão decisório. Um dos grandes desafios quando trabalhamos com problemas do mundo real é que os dados disponíveis não estão no formato de um arquivo *FlatFile*. Para uma melhor entendimento deste conceito, será mostrado o processo de construção do arquivo *FlatFile* para três aplicações do mundo real que possuem diferentes granularidades.

3.1.1 Aplicação 1 - Análise de Risco de Crédito

Credit scoring e *Behavior scoring* são ferramentas que auxiliam as instituições financeiras a decidir sobre a concessão de crédito aos consumidores com base no risco de crédito de suas solicitações [Tho00]. O objetivo dessas ferramentas é atribuir uma pontuação “score” que permita identificar o quão próximo o consumidor está de dois grupos: “bom” que é provável cumprir com suas obrigações

financeiras ou um grupo de “mau”, cujo pedido deve ser negado devido à sua alta probabilidade de faltar com seus compromissos na instituição financeira [OBC13].

Para construir essas soluções, é preciso do histórico de pagamento da base de dados de crédito do cliente e seus resultados de desempenho de crédito. O desempenho de crédito é normalmente medido entre 6 e 24 meses a partir da data de concessão de crédito e é classificado em bom crédito versus desempenho ruim de crédito. Em geral, essas informações estão disponíveis em banco de dados relacionais, e por isso, é necessário que os dados contidos no banco de dados sejam transformados em um formato que permita a aplicação de algoritmo de aprendizagem de máquina e também possibilite as análises necessárias para avaliação dos resultados [NAS17].

Essa transformação consiste em mudar a granularidade dos dados e transformar a representação multidimensional dos dados em uma simples relação organizada em uma tabela desnortinalizada na granularidade em que se pretende tomar a decisão, que será o *FlatFile*. Esta tabela transformada contém uma linha para cada objeto de interesse e um conjunto de colunas que descrevem as características destes objetos [Net+12].

Para uma melhor compreensão deste processo, vamos utilizar como exemplo o esquema relacional contido na Figura 3.1. Para este exemplo, a tabela que define a granularidade do projeto é Empréstimo, pois estamos construindo uma solução para verificar se a instituição financeira deve conceder um empréstimo ou não. No entanto, além desta tabela o esquema possui outras duas tabelas: Cliente e Parcela. Para construir uma solução utilizando os algoritmos de aprendizagem de máquina vistos no capítulo anterior, nós poderíamos utilizar apenas as informações da tabela de empréstimo, no entanto, você não acha que as informações contidas nas tabelas Cliente e Parcela (histórico de pagamentos) são relevantes? Se você acha que os dados destas duas tabelas são relevantes e tentar juntar todas as informações em um único arquivo, vai resultar em várias linhas para o mesmo empréstimo, pois um empréstimo pode ter várias parcelas e com isso a granularidade do projeto seria parcela e não empréstimo! Como resolver esse problema? Mudar a granularidade da relação Parcela para Empréstimo realizando sumarizações e depois juntando as informações com Empréstimo e Cliente. A Figura 3.1 ilustra esse processo, onde as sumarizações da Tabela Parcela são representadas pelas colunas Total_Parc_Pagas e Total_Parc_Nao_Pagas que representam o total pago e o total em aberto no empréstimo respectivamente.

3.1.2 Aplicação 2 - Cupons de descontos para Lojas de Comércio Eletrônico

O mecanismo de uma operação de compra na Internet funciona através das seções. Uma seção acontece quando uma visita a um site de compras é feita por um possível comprador. Durante a seção o visitante clica nos produtos a fim de ver os seus detalhes. Além disso, o comprador possivelmente adicionará ou removerá produtos de sua cesta de compras. No final de uma sessão, é possível que um ou vários produtos da cesta de compras sejam encomendados. As atividades do usuário dentro de uma seção são chamadas de transações. Uma transação é formada por uma série de variáveis. Essas transações são armazenadas em um arquivo e assim é possível trabalhar nesses dados para construir soluções para prever futuras compras [ORS19].

Para construir uma solução para prever se um cliente em uma seção irá realizar uma compra a partir do arquivo de transações, é necessário mudar a granularidade deste arquivo que possui várias linhas para o mesmo usuário. Neste caso, é preciso transformar a granularidade do arquivo para uma única linha por usuário, e para não perder informações de todas as transações do usuário, uma opção seria considerar a última transação da seção e criar novas variáveis sumarizando as informações das transações anteriores, por exemplo: valor mínimo e máximo colocado na cesta de compras da lojas

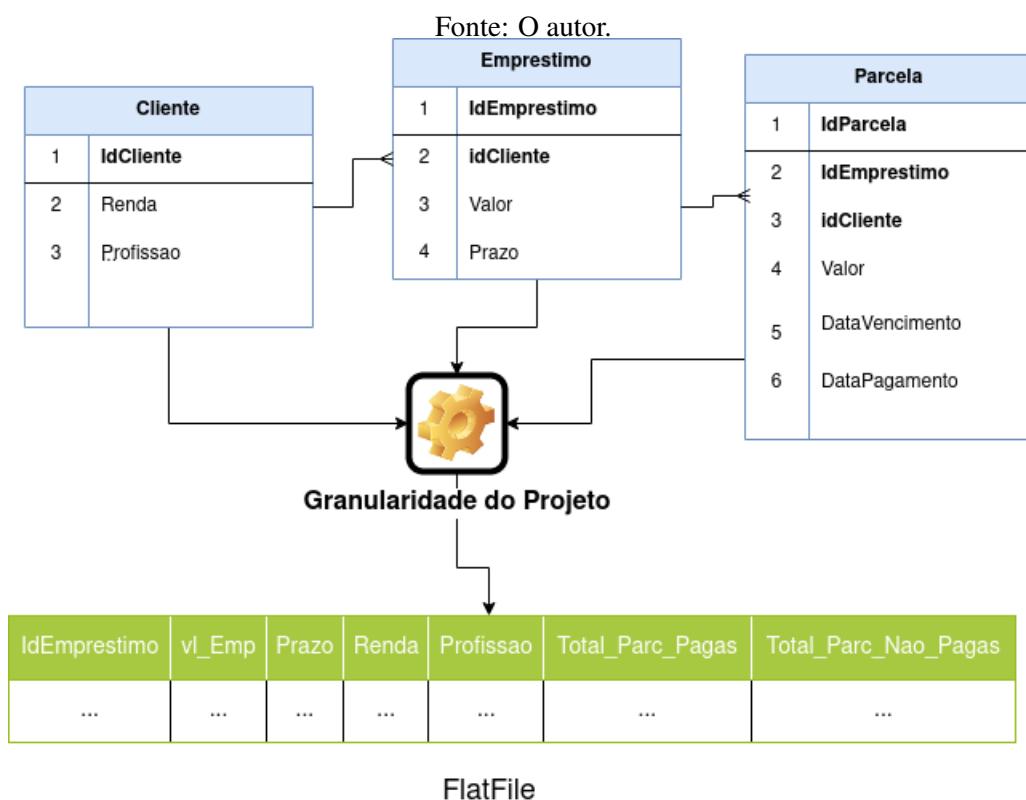


Figura 3.1: Exemplo de definição da Granularidade do projeto a partir de um Banco de Dados Relacional

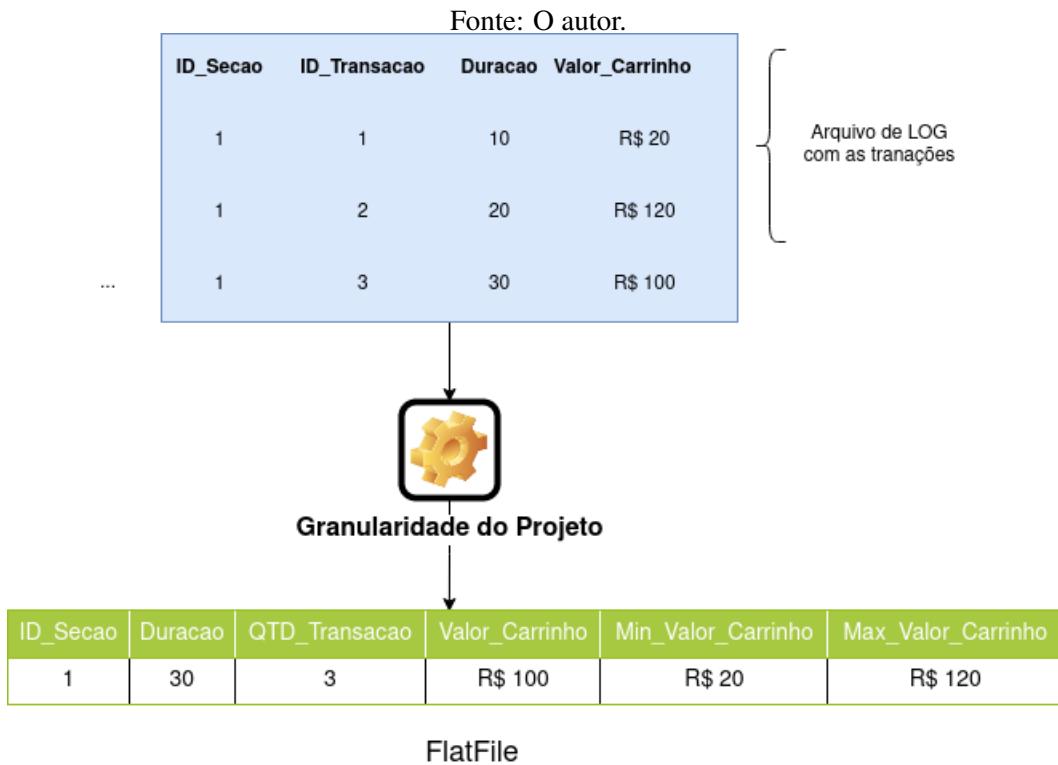


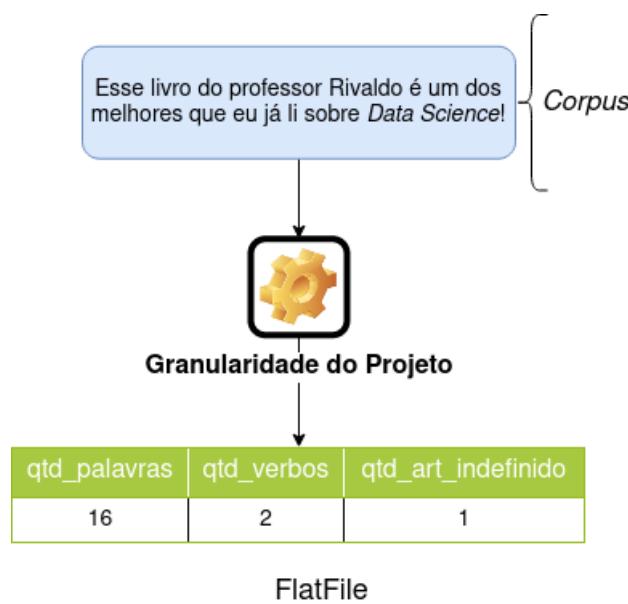
Figura 3.2: Exemplo de definição da Granularidade do projeto a partir de um arquivo de LOG

(carrinho) representado pelas varáveis Min_Valor_Carrinho e Max_Valor_Carrinho respectivamente. A Figura 3.2 ilustra este processo de mudança de granularidade.

3.1.3 Aplicação 3 - Classificação de textos em relação ao perfil do autor

De acordo com [OOQ20], a classificação de textos em relação ao perfil do autor envolve um conjunto previamente definido de autores e um conjunto de *corpus* associado a cada autor. Uma definição de *corpus* é “uma coletânea de textos em linguagem natural, escritos ou falados, geralmente armazenados de forma organizada e informada, além de serem digitalizados a fim de que possam ser lidos por computador” [She09]. A tarefa consiste em criar um classificador capaz de predizer a autoria de textos anônimos a partir de autores previamente informados. A caracterização de autoria é uma variação deste problema. Nesta variação, não são disponibilizados previamente os autores dos *corpus*. O objetivo é inferir informações relevantes sobre o autor como, por exemplo: sua idade [Arg+09]. Por consequência, a tarefa consiste em classificar um ou mais textos em categorias que indicam perfis de autoria, como por exemplo, o grupo de idade do autor.

Para construir uma solução de classificação de perfil de autor a partir de texto é necessário ter um conjunto de corpus de indivíduos com o perfil previamente identificado. Como a granularidade do projeto é o texto do indivíduo e o mesmo é representado por um corpus, é preciso transformar o texto em um formato de *FlatFile*. Para isso, cada *corpus* será transformado em uma linha onde as colunas desta tupla serão características presentes no *corpus*. A Figura 3.3 ilustra essa transformação, onde são criadas as seguintes variáveis como características do *corpus*: qtd_palavras (quantidade de palavras), qtd_verbos (quantidade de verbos) e qtd_art_indefinido (quantidade de artigos indefinidos).



Fonte: O autor.

Figura 3.3: Exemplo de definição da Granularidade do projeto a partir de um *corpus*

3.2 Tratamento de Valores Ausentes

O tratamento de valores ausentes é uma das atividades da etapa de pré-processamento, chamado no inglês de *Missing Values*. A ocorrência de valores ausentes é comum em problemas reais de Ciência dos Dados. A explicação para a ocorrência de valores ausentes pode ser falha na captura dos dados, informações que o usuário não se propôs a informar, variáveis que foram criadas após um determinado período, entre outras. Diversas estratégias para lidar com valores ausentes são encontradas na literatura [GH06], dentre elas podemos destacar:

- **Eliminação de variáveis:** esta é a abordagem mais simples, uma vez que não obriga realizar nenhum tratamento, pois as soluções são construídas apenas com variáveis que possuem informação e as variáveis que possuem valores ausentes são excluídas;
- **Substituição por um único valor:** em vez de eliminar informação, esta abordagem completa os valores ausentes utilizando um valor único. Isso garante que não serão eliminadas informações para construção da solução. O valor ausente pode ser substituído pelo valor mínimo, valor máximo ou valor médio. Esta abordagem só pode ser aplicada para variáveis numéricas;
- **Substituição por um valor estimado:** esta abordagem substitui o valor ausente por um valor estimado a partir de um algoritmo de aprendizagem de máquina como, por exemplo, Redes Neurais Artificiais, Regressão logística, *Random Forest*, entre outras;
- **Criação de um novo valor:** esta abordagem propõe a criação de um novo valor "categoria" para os valores ausentes. Esta abordagem só pode ser aplicada para variáveis categóricas.

3.3 Tratamento de Valores Aberrantes

Além do tratamento de valores ausentes, a etapa de pré-processamento também contempla o tratamento de valores aberrantes, chamado no inglês de *Outliers*. De acordo com [KD19b], os valores aberrantes podem ocorrer por valores capturados corretamente, como exemplo, poucos registros

com renda de milhões de reais ou erros na captura, como exemplo, a altura de uma pessoa com 1,73 cm ao invés de 1,73 metros. Independente do tipo de valor aberrante, ele precisa ser tratado antes do treinamento de um algoritmo de aprendizagem de máquina. Lembre-se de que o objetivo no processo de indução é encontrar uma hipótese $h(x)$ que melhor se aproxime da função real $f(x)$, por isso a presença de valores aberrantes pode distorcer a representatividade do modelo inferido. A identificação e o tratamento são as duas tarefas para tratamento de *outliers*. Elas serão descritas a seguir.

3.3.1 Identificação

De acordo com [Qin97], a forma mais simples de identificar *outliers* é usando o conhecimento do domínio da aplicação. Por exemplo, uma variável de renda não pode obter valores abaixo de zero. Outro exemplo, a nota de uma avaliação escolar não pode ser superior a 10. Geralmente, é possível determinar os possíveis valores máximos e mínimos de uma variável com base no conhecimento do domínio. Portanto, alguns *outliers* podem ser identificados simplesmente comparando com o mínimo e o máximo. Se uma amostra estiver abaixo do mínimo ou acima do máximo, é considerada um erro externo, ou seja, um valor aberrante. No entanto, existem outros métodos encontrados na literatura para identificação de *outliers*. Um dos métodos mais populares é o intervalo interquartil, do inglês *interquartile ranges* (IQR). De acordo com [PLC08], um quartil é qualquer um dos três valores que dividem o conjunto de dados em quatro partes iguais, de modo que cada parte represente 25% dos dados. O primeiro quartil, conhecido como Q1, é o quartil inferior, abaixo dele estão 25% dos dados (valores mais baixos dos dados). O segundo quartil, conhecido como Q2, é a mediana, ele divide o conjunto de dados pela metade e o terceiro quartil, conhecido como Q3, é o quartil superior, acima dele estão 25% dos dados mais altos.

O IQR é a diferença entre os quartis superior e inferior ($Q3 - Q1$). A identificação de *outliers* por este método funciona da seguinte forma:

- Qualquer valor que esteja abaixo de: $Q1 - (1,5 * IQR)$;
- Ou
- Qualquer valor que esteja acima de: $Q3 + (1,5 * IQR)$.

Será considerado um *outlier*.

3.3.2 Tratamento

De acordo com [Qin97], após de identificar os *outliers*, diversas abordagens podem ser adotadas para tratá-los, dentre elas podemos destacar:

- Um *outlier* pode ser substituído pelo valor máximo;
- Um *outlier* pode ser substituído pela valor mínimo;
- Outra abordagem é tratá-los como dados ausentes *Missing Values*.

3.4 Construção de novas variáveis

Outra atividade da etapa de pré-processamento que podemos destacar é a criação de novas variáveis de entrada a partir dos dados existentes. Por que criar novas variáveis? A resposta mais simples é que elas podem aumentar a complexidade do modelo e com isso melhorar o seu poder preditivo. De uma forma geral, a tarefa de construção de novas variáveis é muito mais dependente do conhecimento do domínio do que a construção de um classificador usando um algoritmo de aprendizagem de máquina, por isso o conhecimento do domínio é um requisito [FFJ19]. Alguns estudos têm demonstrado que

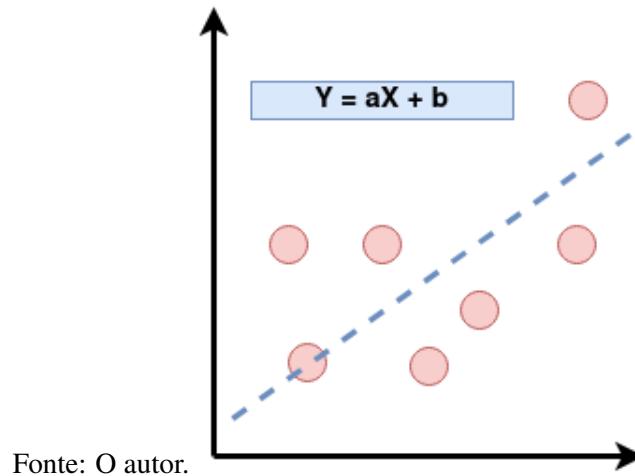


Figura 3.4: Superfície de decisão hipotética para o modelo $Y = aX + b$

a utilização do conhecimento do domínio para construção de novas variáveis de entrada aumenta o poder preditivo do modelo [Rei+20]. Soluções que atingiram boas colocações em importantes competições internacionais, utilizaram a estratégia de construção de novas variáveis para embutir conhecimento do domínio, como pode ser visto em [dos+20].

Para um maior entendimento do aumento do poder preditivo de um modelo a partir da construção de novas variáveis de entrada, vamos analisar a mudança na superfície de decisão de um modelo quando inserimos novas variáveis de entrada. Por uma questão didática, usaremos um problema com apenas uma variável de entrada. O problema é prever o faturamento com bilheteria de um filme com base no orçamento para sua construção. Este exemplo foi retirado do material do curso gratuito da Intel sobre *Machine Learning*¹.

A Figura 3.4 ilustra uma superfície de decisão para o modelo mais simples que poderíamos ter, uma regressão linear, no qual a superfície de decisão é uma reta representada pela equação 3.1.

$$Y = aX + b \quad (3.1)$$

Onde

- Y é a resposta do modelo, que representa o valor previsto para o faturamento com bilheteria;
- X é a única variável de entrada e representa o orçamento do filme;
- a é coeficiente de inclinação da reta;
- b é o intercepto, valor de Y quando X é zero.

Você deve estar se perguntando e onde entram as novas variáveis? Para aumentar a complexidade do modelo, nós podemos adicionar uma nova variável como, por exemplo: o cubo do orçamento disponível (X^3). Com isso, o nosso modelo passa a ter o seguinte formato: $Y = aX + b + cX^3$. Isso muda a superfície de decisão do modelo e sua complexidade, podendo melhorar seu poder preditivo. A Figura 3.5 ilustra a nova representação polinomial do modelo.

¹<https://software.intel.com/content/www/us/en/develop/training/course-machine-learning.html>

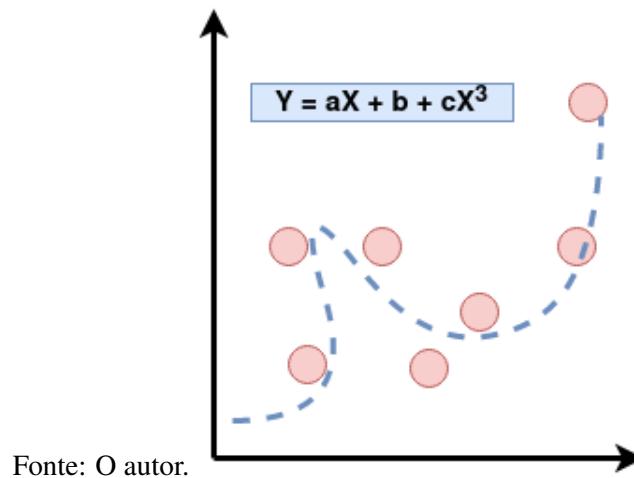


Figura 3.5: Superfície de decisão hipotética para o modelo $Y = aX + b + cX^3$

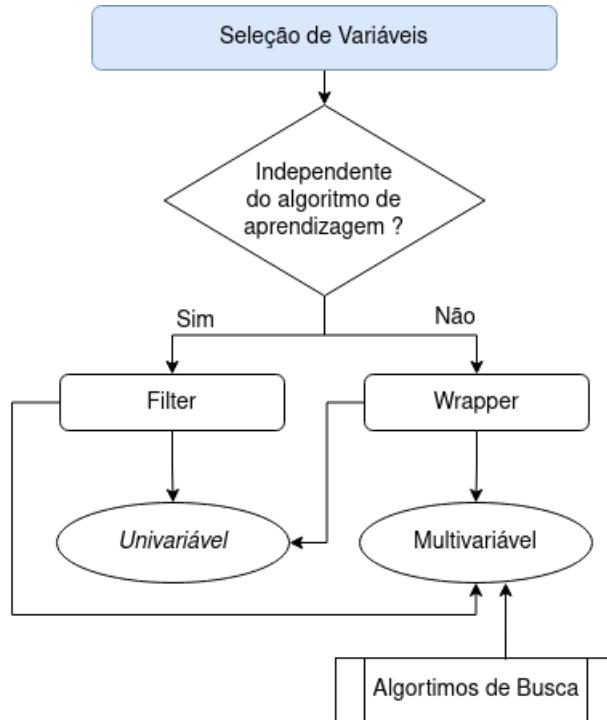
3.5 Redução de Dimensionalidade

A base de dados gerada para o treinamento de um algoritmo de aprendizagem de máquina pode ter um grande número de variáveis de entrada, e esta característica pode afetar negativamente os modelos. Este problema é conhecido na literatura como “maldição da dimensionalidade” do inglês *Curse of Dimensionality* [Bro+95]. Dentre os problemas causados pela maldição da dimensionalidade podemos destacar:

- **Piora no poder discriminatório do modelo:** algumas características não agregam valor discriminatório ou prejudicam o treinamento como, por exemplo: ruídos;
- **Dificuldade de interpretação do modelo:** maior dificuldade para interpretar o modelo que foi induzido. Por exemplo: o que é mais fácil de interpretar: uma árvore de decisão com dez ou 1000 variáveis?
- **Indução do modelo mais lenta:** quanto maior o número de variáveis para construir um modelo, maior será o custo computacional e o tempo.

Para tentar solucionar os problemas causados pela maldição da dimensionalidade, existe a tarefa de Redução da Dimensionalidade, que faz parte da etapa de pré-processamento dos dados e pode ser executado após a limpeza de dados. A tarefa de Redução da Dimensionalidade consiste em transformar a elevada dimensionalidade dos dados, número de variáveis, em uma representação de menor dimensionalidade, no entanto, preservando o máximo de informação possível [DLY97]. Existem várias técnicas de Redução da Dimensionalidade na literatura, no entanto, estas podem ser agrupadas em duas categorias:

1. **Técnicas que aplicam alguma transformação** linear ou não linear no espaço original das variáveis e geram um novo espaço com um menor número de variáveis. Como exemplo desse tipo de técnica podemos citar a Análise de Componentes Principais do inglês *Principal Component Analysis* (PCA), que utiliza uma transformação linear ortogonal para realizar uma mudança de coordenadas para uma nova base [NN19].
2. A outra abordagem de Redução da Dimensionalidade é a **Seleção de variáveis** do inglês *Feature Selection* [HH03]. Essa abordagem não realiza uma transformação no espaço original das variáveis, ela reduz a dimensionalidade selecionando os atributos relevantes e removendo atributos redundantes e/ou irrelevantes.



Fonte: O autor.

Figura 3.6: Topologia das abordagens de seleção de variáveis

Este livro contempla apenas a abordagem de Seleção de Variáveis. Existem duas formas de Seleção de Variáveis: *Filter* e *Wrapper*. *Filter* seleciona as variáveis com base em métricas que são independentes do algoritmo de aprendizagem de máquina que será utilizado para construir o modelo. Por outro lado, *Wrapper* seleciona as variáveis utilizando o desempenho de cada variável no próprio algoritmo de aprendizagem de máquina. Essas duas abordagens de Seleção de Variáveis podem avaliar uma variável por vez ou avaliar um conjunto de variáveis simultaneamente. A Figura 3.6 mostra a topologia das abordagens de seleção de variáveis. É importante destacar que independente da abordagem, quando é avaliado um conjunto de variáveis, o número de possibilidades, ou seja, os subconjuntos possíveis a partir da combinação de todas variáveis é exponencial. A quantidade de subconjunto de variáveis que podem ser avaliados a partir de N variáveis é igual a 2^N . Desta forma, se realizarmos uma análise de forma exaustiva de todos os subconjuntos possíveis de 30 variáveis teríamos 107.3741.824 possibilidades. A solução para avaliar um conjunto de variáveis simultaneamente é utilizar algoritmos de busca como, por exemplo: Algoritmo Genético e Colônia de Formigas [OMF17]. Para um melhor entendimento destas abordagens, as próximas seções fornecem exemplos de *Filter* e *Wrapper*.

3.5.1 Filter Univariável

O *Filter* seleciona variáveis independente do algoritmo de aprendizagem. O Ganho de Informação (GI) é o exemplo mais popular dessa abordagem que avalia uma variável por vez. O GI representa a redução esperada da entropia devido a “classificação” de acordo com um determinado atributo de entrada [FFJ19]. Os detalhes para o cálculo do GI estão disponíveis no Capítulo 2 na seção de Árvore de Decisão. A utilização desta técnica funciona da seguinte forma: antes de treinar um algoritmo

de aprendizagem de máquina, é gerada uma nova base de dados contendo **apenas** as variáveis que possuem um GI acima de um determinado limiar. Mas qual o valor deste limiar? Depende de cada problema! Desta forma, não existe um valor de referência, no entanto, é importante pesquisar na literatura autores que usaram o GI em problemas semelhantes para ter um ponto de partida.

3.5.2 Filter Multivariável

Um exemplo da abordagem *Filter* que avalia mais de uma variável simultaneamente é o método Seleção de Subconjunto com base em Correlação do inglês *Correlation-based Feature Subset Selection* (CfsSubset) proposto por [Hal98]. Como um *Filter*, ele é independente do algoritmo de aprendizagem de máquina. Este método seleciona subconjuntos de variáveis que são altamente correlacionados com o atributo alvo enquanto apresentam baixa correlação com as variáveis de entrada. Uma vez que ele avalia subconjunto de variáveis, ele deve ser utilizado com algum algoritmo de busca heurística para direcionar a pesquisa do melhor subconjunto de variáveis. A equação 3.2 especifica a fórmula de avaliação de cada subconjunto. O numerador pode ser visto como uma indicação do poder preditivo das variáveis de entrada, e o denominador como uma medida de redundância existente entre as variáveis selecionadas.

$$CfsSubset = \frac{k\bar{r}_{ci}}{\sqrt{k+k(k-1)\bar{r}_{ii}}} \quad (3.2)$$

Onde

- k é o número de variáveis selecionadas para avaliação;
- \bar{r}_{ci} é a média da correlação entre as variáveis de entrada e a variável alvo;
- \bar{r}_{ii} é a média da correlação entre as variáveis de entrada.

3.5.3 Wrapper Univariável

Como informado anteriormente, a abordagem *Wrapper* avalia a relevância de atributos com base no seu desempenho em um algoritmo de aprendizagem de máquina. Desta forma, para avaliação individual de atributos, é selecionada uma métrica de avaliação de desempenho e é construído um modelo com cada variável. Por exemplo, se uma base de dados possuir vinte variáveis de entrada, serão construídos vinte modelos, um para cada variável. A seleção ocorre a partir da escolha de um limiar.

3.5.4 Wrapper Multivariável

Por ser uma abordagem multivariável, ela avalia subconjuntos de variáveis e não apenas variáveis individuais. Desta forma, é preciso utilizar um método de busca heurística para direcionar a busca. Esta abordagem é a que possui o maior custo computacional, pois são construídos modelos individuais para cada subconjunto avaliado.

3.6 Etapa de Transformação

Após a etapa de pré-processamento, é necessário transformar os dados no formato adequado ao algoritmo de aprendizagem de máquina selecionado. Por exemplo, árvores de decisão necessitam que as variáveis de entrada sejam categóricas, desta forma, as variáveis numéricas precisam ser discretizadas. Por outro lado, Redes Neurais do tipo *MultiLayer Perceptron* e o KNN precisam que as

Fonte: O autor	ID	Idade	Grupo
	1	39	A
	2	40	B
	3	20	C

Tabela 3.1: Tabela contendo a variável original Grupo

Fonte: O autor	ID	Idade	Grupo_A	Grupo_B
	1	39	1	0
	2	40	0	1
	3	20	0	0

Tabela 3.2: Tabela contendo as variáveis *Dummies* referente à variável Grupo

variáveis de entrada sejam numéricas, por isso, as variáveis categóricas precisam ser transformadas em uma representação numérica [Oli16]. Existem diversas abordagem na literatura para realizar a transformação de variáveis categóricas para numérica, dentre elas podemos destacar as variáveis *Dummies* [Sui57], que funciona da seguinte forma: para cada valor distinto da variável categórica é criada uma variável *dummy* que é binária, ou seja, só pode obter valores zero ou um, indicando a ausência ou presença da categoria. As Tabelas 3.1 e 3.2 ilustram a transformação da variável Grupo usando duas variáveis *Dummies*: Grupo_A e Grupo_B.

É importante evitar redundância nas variáveis *dummies*, pois a redundância gera um problema chamado de multicolinearidade, que ocorre quando as variáveis independentes possuem relações lineares exatas ou aproximadamente exatas [MH82]. Para evitar este problema, se uma variável categórica pode assumir N valores distintos, use N - 1 variáveis *dummy*. A enésima variável *dummy* é redundante.

Outra transformação de dados muito importante quando estamos trabalhando com modelos que precisam de dados numéricos como entrada é a normalização ou padronização [Tre+14]. O objetivo desta transformação é colocar todas as variáveis na mesma escala, no entanto, sem perder informação. Variáveis que possuem grande diferença na escala podem acabar criando um viés para o modelo. A normalização evita esse problema. Dentre os tipos de normalizações mais populares, podemos destacar a MinMax definida pela equação 3.3, que coloca todas as variáveis em uma escala entre zero e um [0, 1].

$$x_{normalizado} = \frac{x - min(X)}{max(X) - min(X)} \quad (3.3)$$



4. Avaliação de Desempenho

Este capítulo apresenta os conceitos envolvidos na etapa de avaliação de desempenho no processo de descoberta do conhecimento. Ele está dividido em duas grandes seções: métodos de avaliação de desempenho e métricas de avaliação de desempenho.

4.1 Métodos de Avaliação de Desempenho

Avaliar o desempenho de um modelo com o mesmo conjunto utilizado na sua construção, não fornece uma boa estimativa de como será a sua predição em dados nunca antes vistos, pois a estimativa será otimista. Uma forma de eliminar este problema é não utilizar todos os dados disponíveis para a construção do modelo [Koh95]. Esta é a ideia básica para uma classe de métodos de avaliação que serão descritos a seguir:

4.1.1 Hold-out

Neste método, uma parte dos dados disponíveis é utilizada para avaliação do modelo, chamado conjunto de teste, e o restante é utilizado para construção do classificador, chamado conjunto de treinamento. Para problemas que existem uma grande quantidade de dados disponíveis, este método é bastante indicado, porém quando a quantidade é limitada sua avaliação pode sofrer influência de como os dados foram separados nos conjuntos de treinamento e teste [BKL99]. Em geral, é utilizado 1/3 dos dados disponíveis para o conjunto de teste e 2/3 para o conjunto de treinamento. É importante destacar que os conjuntos de treinamento e teste devem ser disjunto, ou seja, não pode existir a ocorrência do mesmo exemplo nos dois conjuntos.

4.1.2 K-Fold-Cross-Validation

Nesta abordagem, os dados do conjunto de treinamento são divididos em k subconjuntos disjuntos chamados de *folds* e são construídos k modelos. Para cada modelo, um dos k subconjuntos é

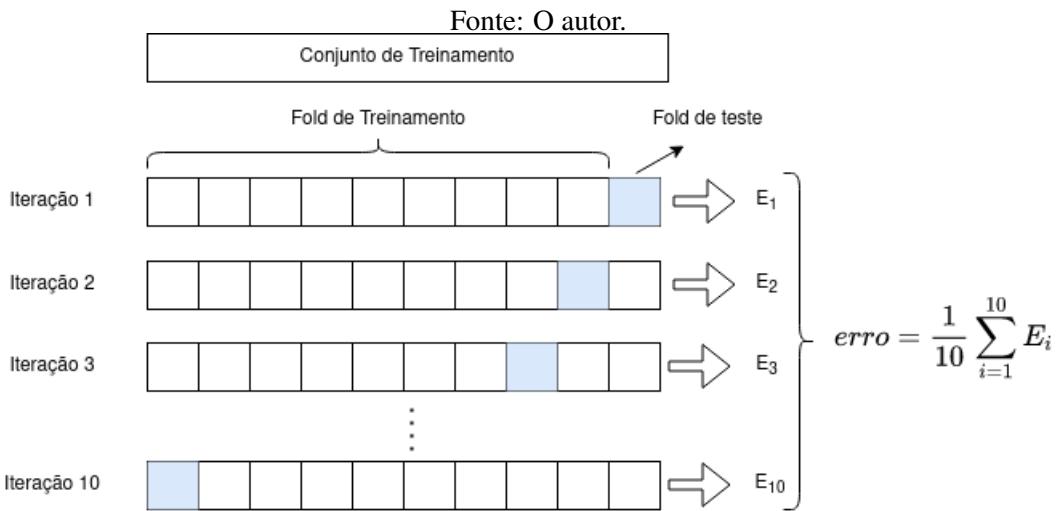


Figura 4.1: Ilustração da divisão do conjunto de treinamento em *folds* para um *K Fold-Cross-Validation* com $K = 10$

separado para teste e os outros $k-1$ conjuntos são utilizados para o treinamento, eliminando assim a dependência de como os dados foram divididos, pois todos os dados podem ser utilizados na construção e avaliação do modelo, o que fornece uma estimativa mais robusta de desempenho [JDM00]. A estimativa do erro é a média do erro dos k modelos construídos. A Figura 4.1 ilustra este processo.

4.1.3 Leave-one-out

O *Leave-one-out* pode ser definido como uma versão do método *K-Fold-Cross-Validation* levado ao extremo, onde o valor de k é igual ao número de padrões disponíveis. Desta forma, um elemento é utilizado como conjunto de teste em cada experimento. Este método é indicado quando o conjunto de dados disponível é muito pequeno [KR99].

4.1.4 Bootstrap

O método *Bootstrap* funciona da seguinte forma, dado um conjunto de dados com N exemplos, é realizada uma amostra de mesmo tamanho N com replicação, ou seja, nesta nova amostra deve existir exemplos replicados. Este conjunto gerado por amostragem com replicação será o conjunto de treinamento. Os exemplos do conjunto original que não foram selecionados para compor o conjunto treinamento formam o conjunto de teste [Koh95].

4.2 Métricas de Avaliação de Desempenho

Na seção anterior, vimos métodos para como avaliar o desempenho de modelos. No entanto, não vimos como calcular o poder de generalização destes modelos. Para calcular o poder de generalização é necessário utilizar métricas de avaliação de desempenho. Esta seção apresenta as principais métricas utilizadas na literatura. Antes de apresentar as métricas, é importante destacar que no aprendizado supervisionado, existem dois tipos de problemas: classificação e regressão. O problema de classificação ocorre quando a variável alvo possui valores categóricos. Por outro lado, a

Fonte: O autor

		Classe Prevista	
		Positivo	Negativo
Classe Real	Positivo	verdadeiro positivo	falso negativo
	Negativo	falso positivo	verdadeiro negativo

Tabela 4.1: Elementos de uma Matriz de Confusão

regressão ocorre quando a variável alvo é contínua. Por este motivo, esta seção apresenta as métricas de avaliação de desempenho dividida em métricas para problemas de classificação e regressão.

4.2.1 Classificação

Como informado anteriormente, as métricas de classificação são utilizadas quando a variável alvo possui valor categórico. Neste livro, serão apresentadas métricas para problemas de classificação binária, ou seja, em que a variável alvo possui apenas dois valores. É importante destacar que, os algoritmos de aprendizagem supervisionado apresentados no Capítulo 2 produzem como resposta uma variável continua, em geral, um valor entre zero e um [0, 1]. Desta forma, para tomar a decisão de classificar um registro como sendo pertencente a uma classe ou outra, é necessário utilizar um ponto de corte, ou seja, um valor de limiar. Se a saída do modelo estiver abaixo deste limiar, o exemplo pertence a uma classe, caso contrário, a classificação será pela outra classe. Por padrão, os principais *framework* de aprendizado de máquina como *Weka*¹ e *Scikit Learning*² utilizam o valor de 0,5 como limiar. Em geral, os nomes das duas classes em problema de classificação binária são: positiva (1) e negativa (0). No entanto, os nomes das classes depende do domínio da aplicação.

De uma forma geral, as métricas de avaliação de desempenho para problemas de classificação binária podem ser descritas a partir de uma matriz de confusão. Ela é formada por uma tabela com duas linhas e duas colunas (2 x 2). As linhas representam a classe real dos padrões. As colunas representam as classes preditas pelo modelo. Desta forma, cada célula da tabela pode conter a informação de acerto ou erro do modelo. Mais especificamente, a célula 1 x 1 representa os acertos da classe positiva, ou seja, os padrões que foram classificados como positivo e realmente são positivos. Desta forma, a célula 1 x 1 da matriz de confusão contém o número de verdadeiros positivos. A célula 2 x 1 representa os erros da classe positiva, ou seja, os padrões que foram classificados como positivo, mas na verdade são negativos. Desta forma, a célula 2 x 1 da matriz de confusão contém o número de falso positivo. A célula 1 x 2 representa os erros da classe negativa, ou seja, os padrões que foram classificados como negativo, mas na verdade são positivos. Desta forma, a célula 1 x 2 da matriz de confusão contém o número de falso negativo. A célula 2 x 2 representa os acertos da classe negativa, ou seja, os padrões que foram classificados como negativo e realmente são negativos. Desta forma, a célula 2 x 2 da matriz de confusão contém o número de verdadeiro negativo. A Tabela 4.1 ilustra o formato da matriz de confusão.

Para um maior entendimento sobre a matriz de confusão, vamos construir uma a partir dos dados de classificação apresentados na Tabela 4.2. Esta tabela simula a classificação de quatro padrões. A primeira coluna representa a classe real do padrão. A segunda coluna representa a classe prevista pelo modelo. A matriz de confusão deve ser preenchida com a quantidade de verdadeiros positivos, falsos positivos, falsos negativos e verdadeiros negativos. A Tabela 4.3 exibe a matriz de confusão para este exemplo.

¹<https://www.cs.waikato.ac.nz/ml/weka/>

²<https://scikit-learn.org/stable/>

	Classe Atual	Classe Prevista
Fonte: O autor	POSITIVA	NEGATIVA
	POSITIVA	POSITIVA
	POSITIVA	NEGATIVA
	NEGATIVA	NEGATIVA

Tabela 4.2: Classificação de quatro padrões

Fonte: O autor		Classe Prevista	
		Positivo	Negativo
		Classe Real	
	Positivo	1	2
	Negativo	0	1

Tabela 4.3: Matriz de Confusão preenchida com os valores do exemplo

Outras métricas de avaliação de desempenho podem ser derivadas a partir da matriz de confusão. Entre elas podemos destacar: acurácia, precisão, sensibilidade, especificidade e a área sobre a curva ROC (*Receiver Operating Characteristic*). Estas métricas serão definidas a partir da nomenclatura usada para definir os elementos da matriz de confusão ilustrados na Tabela 4.4. A equação 4.1 mostra a fórmula de cálculo da acurácia, que é a medida mais intuitiva, pois representa o percentual de acerto total. A equação 4.2 mostra a fórmula de cálculo da precisão, que representa a razão entre os exemplos que foram previsto como positivos corretamente sobre o total de previstos como positivos (verdadeiro positivo + falso positivo). A sensibilidade por sua vez representa o quanto de exemplos positivos são classificados corretamente, ou seja, é a razão entre os exemplos que foram previsto como positivos corretamente sobre o total de positivos reais (verdadeiro positivo + falso negativo). A equação 4.3 mostra a fórmula de cálculo da sensibilidade que também é conhecida na literatura como *Recall*.

Por outro lado, a especificidade representa o quanto de exemplos negativos são classificados corretamente, ou seja, é a razão entre os exemplos que foram previsto como negativos corretamente sobre o total de negativos reais (verdadeiro negativo + falso positivo). A equação 4.4 mostra a fórmula de cálculo da especificidade.

$$\text{Acurácia} = \frac{a + d}{a + b + c + d} \quad (4.1)$$

$$\text{Precisão} = \frac{a}{a + b} \quad (4.2)$$

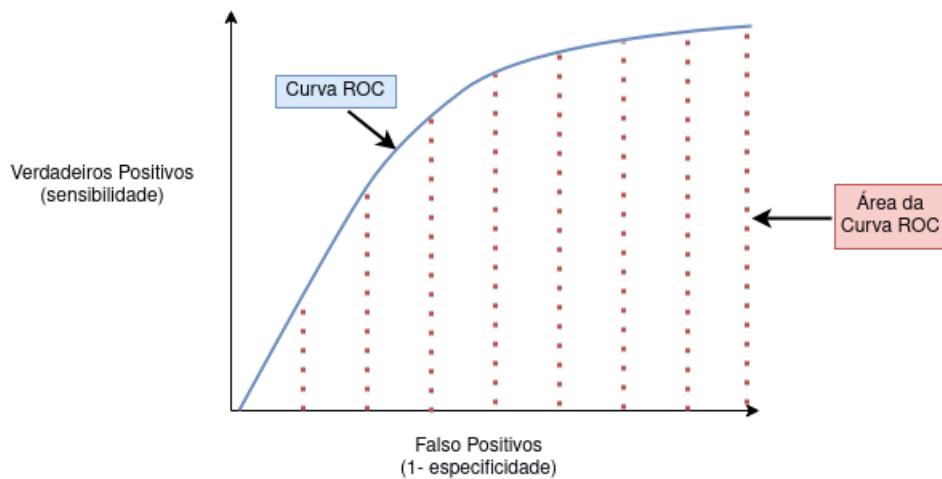
$$\text{Sensibilidade(Recall)} = \frac{a}{a + c} \quad (4.3)$$

$$\text{Especificidade} = \frac{d}{b + d} \quad (4.4)$$

Fonte: O autor

		Classe Prevista	
		Positivo	Negativo
Classe Real	Positivo	a	c
	Negativo	b	d

Tabela 4.4: Matriz de Confusão com nomenclatura usada para auxiliar na definição de outras métricas de avaliação de desempenho



Fonte: O autor.

Figura 4.2: Ilustração da Curva ROC

Conforme informado no início deste capítulo, a decisão binária é tomada a partir de um limiar, abaixo do qual a decisão é feita para uma classe ou outra. No entanto, a definição deste limiar é influenciada de acordo com o objetivo do problema, levando-se em consideração, geralmente, que os custos dos erros são diferentes para cada classe [Ade+08]. As métricas apresentadas anteriormente necessitam que a saída do modelo seja categórica para poder se construir a matriz de confusão, o que obriga a utilização de um limiar. Por isso, a métrica mais utilizada na literatura para avaliação de problemas de classificação binária é independente deste limiar. A métrica é a área sobre a curva ROC [Ade+10]. Essa métrica realiza a análise por meio de um método gráfico simples e robusto, o qual permite estudar a variação da sensibilidade e especificidade do modelo para diferentes valores de ponto de corte [TPS08]. As curvas ROC mostram a relação das taxas de falsos positivos e verdadeiros positivos através da variação de um limiar. Esta relação prediz o comportamento dos classificadores, independentemente dos custos e da distribuição das classes. Em uma curva ROC, o eixo das ordenadas (y) representa a taxa de verdadeiros positivos e o eixo das abscissas (x) representa a taxa de falsos positivos. Para cada ponto de corte, a sensibilidade e a taxa de falso positivos são calculados e colocados um em cada eixo de um gráfico bidimensional, produzindo a curva ROC. A Figura 4.2 ilustra o gráfico de uma curva ROC. A taxa de falso positivos pode ser obtida pelo complemento da especificidade ($1 - \text{especificidade}$).

4.2.2 Regressão

As métricas de regressão são utilizadas quando a variável alvo possui valor contínuo. Neste livro, serão apresentadas as métricas mais populares de uma forma simples e direta. Não está no escopo

deste livro realizar uma análise aprofundada dos pontos fortes e fracos de cada uma delas. Uma análise inicial pode ser encontrada em [San20]. As definições das métricas utilizam os seguintes símbolos:

- N : representa o tamanho da amostra, ou seja, o número de exemplos de avaliação;
- d_i representa o valor real da variável alvo para o exemplo i ;
- o_i representa o valor previsto pelo modelo para a variável alvo para o exemplo i .

O erro médio quadrático é conhecido pela sigla MSE do inglês *Mean Squared Error*. A sua fórmula é exibida na equação 4.5. A principal desvantagem do MSE é sua escala, pois o valor do erro é quadrático. Uma alternativa para solucionar o problema da escala do MSE é a métrica da raiz quadrada do erro médio quadrático, que é conhecida pela sigla RMSE do inglês *Root Mean Squared Error*. O RMSE é a raiz quadrada do MSE, conforme pode ser observado na equação 4.6. O erro médio absoluto é outra métrica que pode ser utilizada como alternativa para solucionar o problema da escala do MSE. O erro médio absoluto é conhecida pela sigla MAE do inglês *Mean Absolute Error*. MAE calcula o erro como módulo, conforme pode ser observado na equação 4.7. Existem métricas que são independentes de escala, dentre elas, podemos citar o erro percentual médio absoluto, que é conhecida pela sigla MAPE do inglês *Mean Absolute Percentage Error*. O MAPE calcula o erro de previsão na forma de percentual, o que facilita a interpretação do resultado. A sua fórmula é exibida na equação 4.8.

$$MSE = \frac{1}{N} \sum_{i=1}^N (o_i - d_i)^2 \quad (4.5)$$

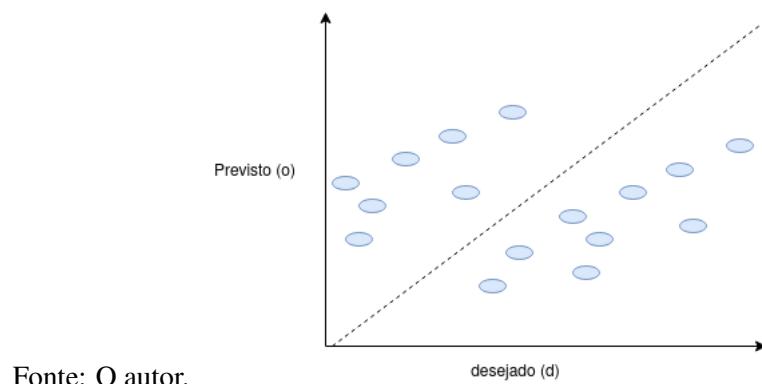
$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (o_i - d_i)^2} \quad (4.6)$$

$$MAE = \frac{1}{N} \sum_{i=1}^N |o_i - d_i| \quad (4.7)$$

$$MAPE = \frac{1}{N} \sum_{i=1}^N \frac{|o_i - d_i|}{d_i} * 100 \quad (4.8)$$

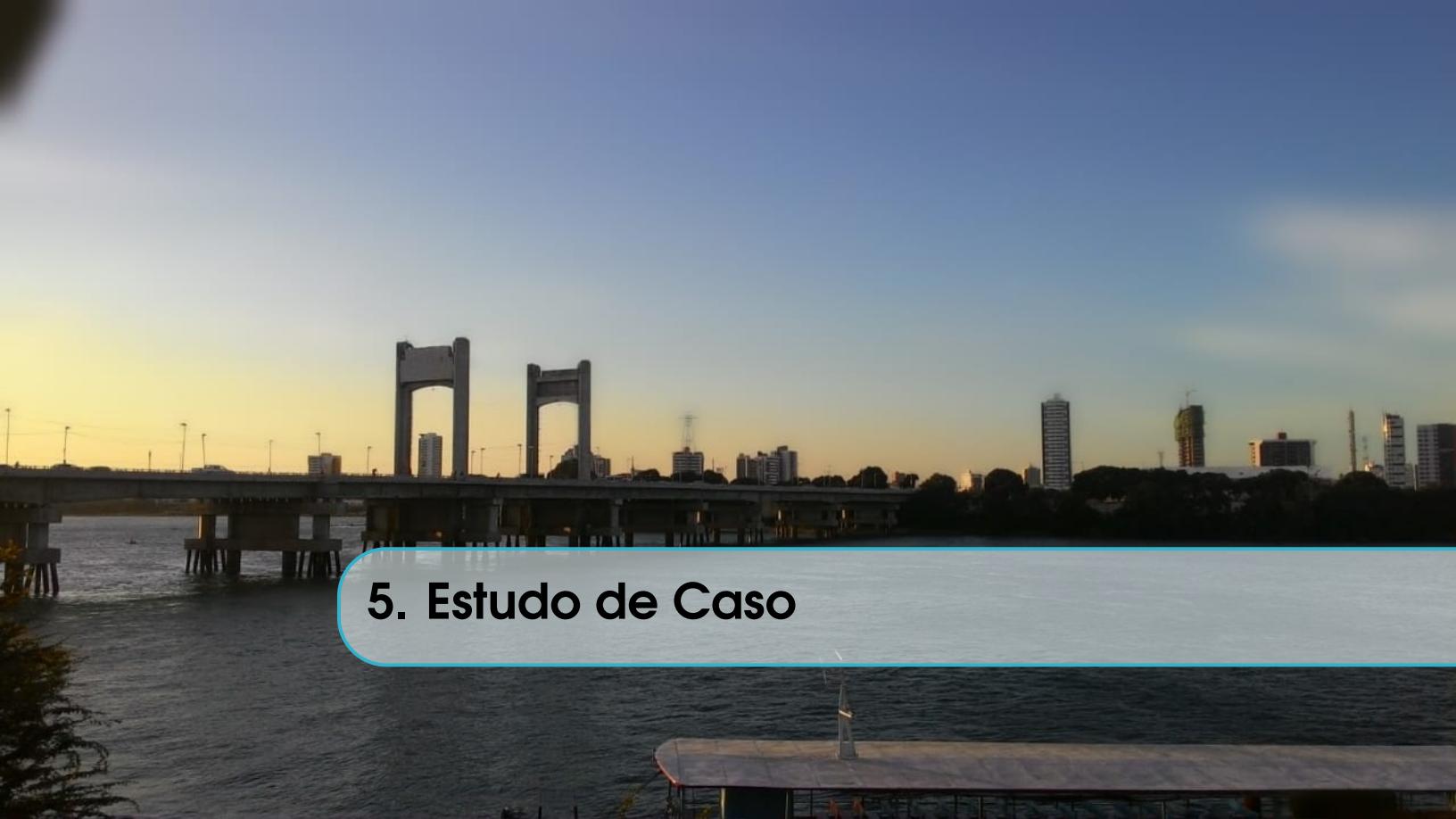
De acordo com [HA18], o Coeficiente de Determinação, também conhecido como R^2 , é uma métrica que mede a qualidade de ajuste do modelo com base na proporção da variabilidade do atributo alvo (d) que é explicada pela predição do modelo (o). O Coeficiente de Determinação é obtido a partir da elevação ao quadrado do coeficiente de correlação de Pearson. O coeficiente de Pearson também conhecido pela letra R mede a correlação linear entre duas variáveis e pode assumir valores entre -1 e +1. Desta forma, o R^2 pode assumir valores entre zero e um, onde um $R^2 = 1$ significa um modelo perfeito com 100% de acerto. Por outro lado, um $R^2 = 0$ representa um modelo que erra tudo. No entanto, em problemas do mundo real é pouco provável obter valores extremos de R^2 . A equação 4.9 exibe a fórmula de cálculo do R^2 , nesta fórmula, as siglas cov e var presentam a covariância e a variância. Na literatura, é comum apresentar o valor do R^2 associado a um gráfico de dispersão para auxiliar em uma análise visual conforme pode ser observado na Figura 4.3.

$$R^2 = \left(\frac{cov(d, o)}{var(d) * var(o)} \right)^2 \quad (4.9)$$



Fonte: O autor.

Figura 4.3: Ilustração de um gráfico de dispersão



5. Estudo de Caso

O objetivo deste capítulo é colocar em prática todo conteúdo visto neste livro. Nós vamos construir um projeto de Ciência de Dados usando o processo de KDD. Para isso, usaremos como estudo de caso uma competição internacional. Inicialmente, serão descritos os pré-requisitos necessários para execução do projeto. Em seguida, cada etapa do processo de KDD será instanciada.

5.1 Pré-requisitos técnicos para execução do projeto

Esta seção descreve os pré-requisitos técnicos necessários para execução do projeto. É importante destacar que existem vários *frameworks* para construção de projetos de Ciência de Dados. Além disso, é possível construir soluções a partir do zero sem uso de qualquer *frameworks*. No entanto, nós usaremos as ferramentas e os *frameworks* mais populares atualmente para criação do nosso projeto, são eles:

5.1.1 Python

Ela é uma linguagem de programação de propósito geral. No entanto, usando Python com as bibliotecas Pandas e Scikit-learn é possível realizar a maioria, senão todas, as operações de um projeto de Ciência de Dados [BK16].

5.1.2 Jupyter Notebook

É uma aplicação WEB de código aberto que permite criar e documentar códigos com uso de equações, imagens e textos com formatação. Jupyter¹ é um acrônimo das linguagens de programação Julia, Python e R. Nós usaremos o Jupyter para escrever e documentar o código do nosso projeto. A Figura 5.3 mostra um exemplo de código no Jupyter.

¹<https://jupyter.org/>

Fonte: Site do Jupyter.

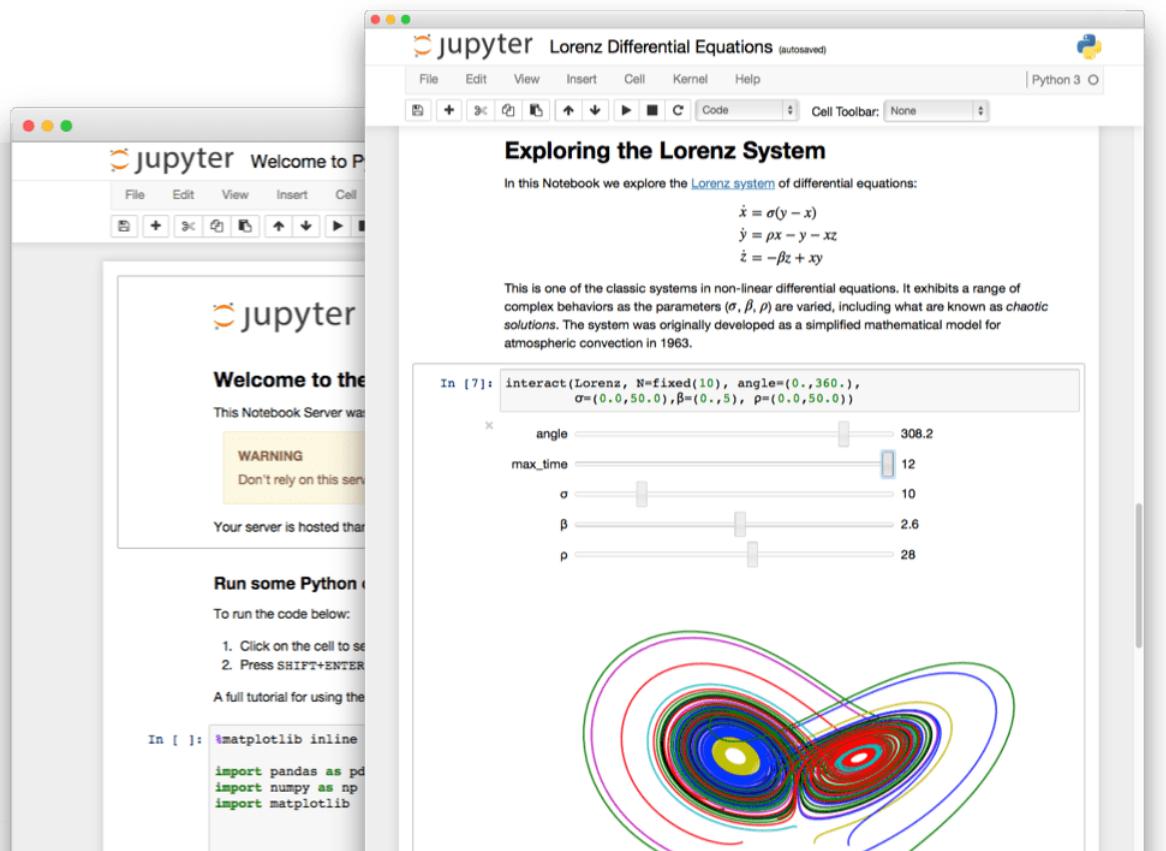


Figura 5.1: Tela de exemplo no Jupyter

5.1.3 Pandas

É uma biblioteca em Python para manipulação de dados tabulares. Uma única tabela pode ter dados de diferentes tipos. O pandas² permite a leitura e escrita de dados em diferentes formatos de arquivos como, por exemplo: CSV, Microsoft Excel, Bancos de Dados e HDF5. Outra ponto forte desta biblioteca é que ela disponibiliza funções para agregação de dados e estatísticas.

5.1.4 Scikit-learn

É uma biblioteca em Python para aprendizado de máquina. O Scikit-learn³ disponibiliza diversos algorítimos tanto para o aprendizado supervisionado como o não supervisionado. Além disso, ela também fornece várias ferramentas para ajuste de modelo, pré-processamento de dados, seleção e avaliação de modelos.

5.2 Projeto

5.2.1 Seleção dos dados

Uma das tarefas desta etapa é o entendimento do domínio da aplicação, que inclui definir o objetivo da aplicação e entender o que é relevante para o problema. Desta forma, essa seção descreve o problema e os dados que serão usados para construção da nossa solução.

Definição do Problema

O mecanismo de uma operação de compra na internet funciona através de seções. Uma seção acontece quando uma visita a um site de compras é feita por um possível comprador. Durante a seção o visitante clica nos produtos a fim de ver os seus detalhes. Além disso, o comprador possivelmente irá adicionar ou remover produtos de sua cesta de compras. No final de uma sessão, é possível que um ou vários produtos da cesta de compras sejam encomendados. As atividades do usuário dentro de uma seção são chamadas de transações. Uma transação é formada por uma série de variáveis. Essas transações são armazenadas em arquivos de LOG e assim é possível trabalhar nesses dados para prever futuras compras.

Prever se o visitante faz uma compra ou não com base nos dados das transações coletados durante a sessão foi o desafio proposto pela *Data Mining Cup* (DMC) de 2013. A DMC⁴ é uma competição mundial que tem como foco tarefas de Mineração de Dados reais fornecidas por empresas. Em 2013, a DMC disponibilizou LOGs de uma loja web para construção de uma solução de classificação binária tendo como variável alvo se a sessão resultou em compra ou não. Foram disponibilizados dois arquivos:

- **Transact_train**: Arquivo para ser usado para construção da solução, contendo aproximadamente 400,000 transações;
- **Transact_class**: Arquivo para medir a qualidade da solução desenvolvida, contendo aproximadamente 150,000 transações.

A lista com todas as variáveis disponíveis é apresentada na Tabela 5.1.

5.2.2 Pré-processamento

Este seção foi escrita no Jupyter-notebook e contém a descrição e o código para realização da etapa de Pré-processamento do projeto.

²<https://pandas.pydata.org/>

³<https://scikit-learn.org/stable/>

⁴<https://www.data-mining-cup.com/>

Fonte: O autor

Variável	Descrição
sessionNo	Número de execução da seção
startHour	Hora em que a sessão começou
startWeekday	Dia da semana em que a seção começou (1Mo; 2Tu;...;7Su)
duration	Tempo corrido em segundos desde o início da seção.
cCount	Número de produtos clicados
cMinPrice	O preço mais baixo de um produto clicado
cMaxPrice	O preço mais alto de um produto clicado
cSumPrice	Soma dos preços de todos os produtos clicados
bCount	Quantidade de produtos colocados no cesto de compras
bMinPrice	Preço mais baixo de todos os produtos colocados no cesto de compras.
bMaxPrice	Preço mais alto de todos os produtos colocados no cesto de compras
bSumPrice	Soma dos preços de todos os produtos colocados no cesto de compras
bStep	Etapa de processamento da compra
onlineStatus	Indicação se o cliente está on-line (y - sim; n - não)
availability	Status de entrega
customerID	Número do cliente
maxVal	Preço de compra máximo admissível para o cliente
customerScore	Avaliação do cliente do ponto de vista da loja
accountLifetime	Tempo de vida da conta do cliente em meses
Payments	Número de pagamentos efetuados pelo cliente
Age	Idade do cliente
address	Forma de endereço do cliente (1 - Sr.; 2 - Sra.; 3 - Empresa)
lastOrder	Tempo em dias transcorridos desde a última venda
order	Resultado da sessão (y - compra; n - não compra)

Tabela 5.1: Lista de variáveis da competição DMC 2013

Acesso aos arquivos

- Fazer o download dos arquivo de Treinamento e Teste.
- Link:<https://www.data-mining-cup.com/reviews/dmc-2013/>
- Descompactar no mesmo diretório deste arquivo PreProcessamento.ipynb os arquivos:
 - transact_train.txt (Treinamento com a variável alvo)
 - transact_class.txt (Teste sem a variável alvo)
 - realclass_t1.txt (Variável alvo do teste)

Importando as bibliotecas

```
import numpy as np
import pandas as pd
```

Carregando os dados de treinamento

```
TRN_Original = pd.read_csv('transact_train.txt', sep='|')
num_linhas_TRN, num_colunas_TRN = TRN_Original.shape
print('Número de linhas do arquivo de treinamento: ', num_linhas_TRN)
print('Número de colunas do arquivo de treinamento: ', num_colunas_TRN)
```

Carregando os dados de teste

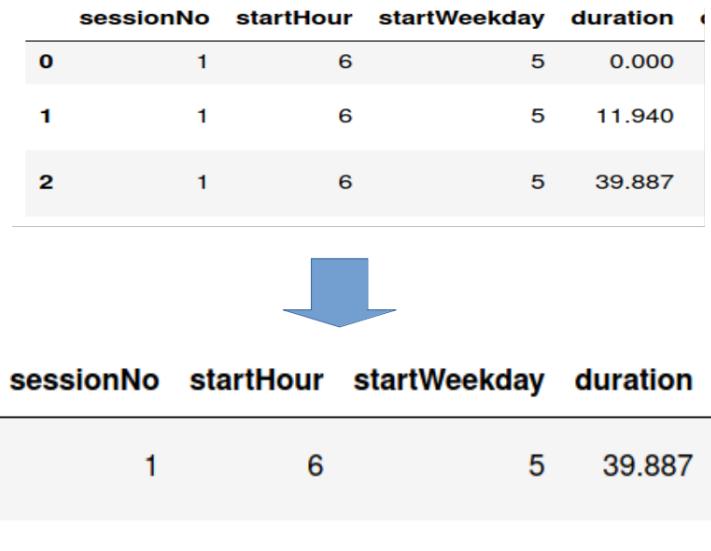
```
TST_Original = pd.read_csv('transact_class.txt', sep='|')
num_linhas_TST, num_colunas_TST = TST_Original.shape
print('Número de linhas do arquivo de teste: ', num_linhas_TST)
print('Número de colunas do arquivo de teste: ', num_colunas_TST)
```

Mudando a Granularidade dos arquivos

Os arquivos originais disponíveis pela competição estavam com a granularidade de transação, ou seja, cada linha do arquivo representa uma transação de uma seção. No entanto, o objetivo da tarefa era prever a probabilidade de compra de cada visitante, que é representada por cada seção. Desta forma, é necessário mudar a granularidade dos arquivos disponíveis de transação para seção. Para isso, nós vamos gerar novas bases de treinamento e teste onde cada linha representa uma seção, que será a última transação da seção. A Figura 5.3 ilustra o processo de mudança de granularidade para a seção.

Para realizar a mudança de Granularidade, são sugeridos os seguintes passos:

- **Passo 1:** criar uma função que retorna um Dicionário onde: Key=Session e Value=Lista com o índice das linhas que sessionID = Key;
- **Passo 2:** criar um dicionário com sessionID e a lista de transações;
- **Passo 3:** recuperar o índice da linha da última transação por sessionID;
- **Passo 4:** criar um DataFrame com a última transação por sessionID.



The diagram illustrates the change in granularity. At the top, there is a table with 3 rows and 5 columns. The columns are labeled: sessionNo, startHour, startWeekday, duration, and a column with values 0.000, 11.940, and 39.887. Below this table is a large blue downward-pointing arrow. Below the arrow is a second, smaller table with 1 row and 5 columns. The columns are labeled: sessionNo, startHour, startWeekday, duration, and a single value 39.887.

	sessionNo	startHour	startWeekday	duration
0	1	6	5	0.000
1	1	6	5	11.940
2	1	6	5	39.887

	sessionNo	startHour	startWeekday	duration
	1	6	5	39.887

Fonte: O autor.

Figura 5.2: Ilustração da mudança de granularidade

```

sessionIDs_TRN = TRN_Original['sessionNo']
sessionIDs_TST = TST_Original['sessionNo']

#Passo 1
def getSessionWithTransaction(ReplicatedSession):
    idx_TransactionSession = {s:[] for s in set(ReplicatedSession)}
    for i in range(len(ReplicatedSession)):
        s = ReplicatedSession[i]
        idx_TransactionSession[s].append(i)
    return idx_TransactionSession
#Passo 2
idx_SessionWithTransaction_TRN = getSessionWithTransaction(sessionIDs_TRN)
idx_SessionWithTransaction_TST = getSessionWithTransaction(sessionIDs_TST)
#Passo 3
idx_lastTransactionBySession_TRN = [np.max(x) for x in
                                      idx_SessionWithTransaction_TRN.values
                                      ()]
idx_lastTransactionBySession_TST = [np.max(x) for x in
                                      idx_SessionWithTransaction_TST.values
                                      ()]
#Passo 4
TRN_X = TRN_Original.iloc[idx_lastTransactionBySession_TRN, :-1]
TRN_Y = TRN_Original.iloc[idx_lastTransactionBySession_TRN, -1]
# Passo auxiliar para eliminar o SettingWithCopyWarning
TST_X_tmp = TST_Original.iloc[idx_lastTransactionBySession_TST, :]
TST_X = TST_X_tmp.copy()
TST_Y = TST_Y_Original['prediction']

```

```
num_linhas_TRN_X, num_colunas_TRN_X = TRN_X.shape
print('Número de linhas: ', num_linhas_TRN_X)
print('Número de colunas: ', num_colunas_TRN_X)
```

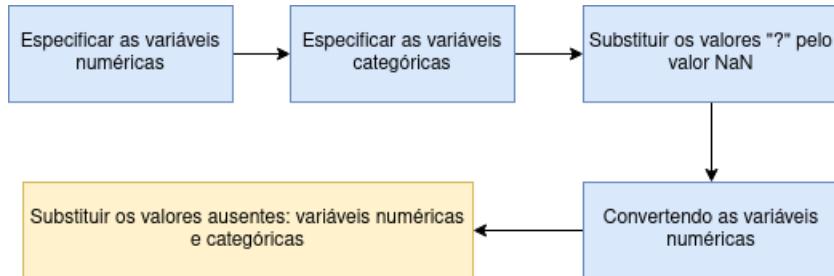
```
num_linhas_TST_X, num_colunas_TST_X = TST_X.shape
print('Número de linhas: ', num_linhas_TST_X)
print('Número de colunas: ', num_colunas_TST_X)
```

```
#Converter y -> 1 e n -> 0
TRN_Y = TRN_Y.replace({'y': 1, 'n': 0})
```

Tratando os valores ausentes

Conforme apresentado no capítulo 3, a ocorrência de valores ausentes, do inglês Missing Values, é comum em projetos de Ciência dos Dados. O capítulo 3 apresentou diferentes estratégias para lidar com valores ausentes. No entanto, para esse Notebook nós selecionamos duas abordagens: 1) substituir pela média para variáveis numéricas e 2) substituir por um valor fixo para variáveis categóricas.

O framework Pandas identifica os valores ausentes pelo valor np.NaN. Desta forma, é preciso substituir os valores “?” que estão nos arquivos, fornecidos pela DMC por np.NaN. A Figura 5.3 ilustra o fluxograma de tratamento para os valores ausentes.



Fonte: O autor.

Figura 5.3: Fluxograma de tratamento para os valores ausentes

O Tratamento de valores ausentes deve ser realizado para os conjuntos de treinamento e teste na grão decisório “última transação da sessão”, ou seja, para X_TRN e X_TST. No entanto, para o grão original “cada transação da sessão” deve ser realizado apenas o tratamento do valor “?”, pois vamos utilizar o grão original para criar novas variáveis e substituir os valores ausentes no grão original mudaria a estatística das novas variáveis. Por exemplo, suponha que uma sessão teve 10 transações e que 1 das transações não possui o valor do carrinho, se calcularmos a média do valor do carrinho da sessão com o valor ausente, a média será calculada com o valor das nove transações que possuem valor. No entanto, se substituirmos o valor ausente no grão original, essa estatística muda.

Obs-1: Não deve realizar tratamento de valores ausentes para código como por exemplo: customerNo e sessionNo.

Obs-2: Identificar quais variáveis devem ser consideradas como numéricas e categóricas e quais devem realizar tratamento de valores ausentes é parte do entendimento do negócio. Para esta tarefa, você deve ler o documento *features.pdf* que possui informações sobre o tipo das variáveis e se possuem valores ausentes.

Lista de variáveis que devem ser consideradas como numérica

```
l_varNumeric = ['cMinPrice', 'cMaxPrice', 'cSumPrice', 'bMinPrice',
                 'bMaxPrice', 'bSumPrice', 'bStep',
                 'maxVal', 'customerScore', 'accountLifetime', 'payments',
                 'age', 'address', 'lastOrder']
```

Lista de variáveis que devem ser consideradas como String

```
l_varString = ['availability', 'onlineStatus']
```

Funções auxiliares para o tratamento de Missing Values

- **replaceValueMissing**: função que substitui um valor por np.nan em todo dataframe;
- **convertFloat** : função que converte a lista de variáveis para float;
- **replaceMissingByMean**: função que substitui missing value pela média (apenas variáveis numéricas);
- **replaceMissingByFixedValue**: função que substitui missing value por valor fixo (apenas variáveis string).

```
def replaceValueMissing(vl, listVar, data):
    for v in listVar:
        rows = data[v] == vl
        data.loc[rows, v] = np.NaN
```

```
def convertFloat(listVar, data):
    for v in listVar:
        data[v] = data[v].astype(float)
```

```
def replaceMissingByMean(listVar, data):
    for v in listVar:
        avg = data[v].mean(axis=0)
        data[v].fillna(avg, inplace=True)
```

```
def replaceMissingByFixedValue(vl, listVar, data):
    for v in listVar:
        data[v].fillna(vl, inplace=True)
```

Aplicando as funções para tratamento dos valores ausentes

A limpeza dos dados deve seguir:

- **Passo 1:** substituir a string ? por np.nan em todas as variáveis da base na granularidade do projeto;
- **Passo 2:** converter as variáveis numéricas de object para float (porque int não aceita calcular média com NaN);
- **Passo 3:** substituir os missing values das variáveis numéricas pela média;
- **Passo 4:** substituir os missing values das variáveis String por um valor fixo.

```

# Passo 1
replaceValueMissing('?', l_varNumeric, TRN_X)
replaceValueMissing('?', l_varString, TRN_X)
replaceValueMissing('?', l_varNumeric, TST_X)
replaceValueMissing('?', l_varString, TST_X)

# Passo 2
convertFloat(l_varNumeric, TRN_X)
convertFloat(l_varNumeric, TST_X)

# Passo 3
replaceMissingByMean(l_varNumeric, TRN_X)
replaceMissingByMean(l_varNumeric, TST_X)

# Passo 4
replaceMissingByFixedValue('ausente', l_varString, TRN_X)
replaceMissingByFixedValue('ausente', l_varString, TST_X)

```

Nem todos os valores ausentes devem ser tratados de forma igual para a base na granularidade do projeto

Observe que a variável customerNo informa o número do cliente e toda vez que essa variável é “?” as variáveis maxVal, customerScore, accountLifetime, payments, age, address, lastOrder também tinham valor “?” (Ver base na granularidade do projeto antes do tratamento dos valores ausentes). Desta forma, não se trata de um valor ausente, pois se é um novo cliente que está realizando a compra, essas informações não existem. Para resolver essa questão, vamos atribuir essas variáveis para o valor zero quando o customerNo for igual a “?”.

```

# Treinamento
rows_TRN = TRN_X['customerNo']=='?'
TRN_X.loc[rows_TRN, 'maxVal'] = 0
TRN_X.loc[rows_TRN, 'customerScore'] = 0
TRN_X.loc[rows_TRN, 'accountLifetime'] = 0
TRN_X.loc[rows_TRN, 'payments'] = 0
TRN_X.loc[rows_TRN, 'age'] = 0
TRN_X.loc[rows_TRN, 'address'] = 0
TRN_X.loc[rows_TRN, 'lastOrder'] = 0

# Teste
rows_TST = TST_X['customerNo']=='?'
TST_X.loc[rows_TST, 'maxVal'] = 0
TST_X.loc[rows_TST, 'customerScore'] = 0
TST_X.loc[rows_TST, 'accountLifetime'] = 0
TST_X.loc[rows_TST, 'payments'] = 0
TST_X.loc[rows_TST, 'age'] = 0
TST_X.loc[rows_TST, 'address'] = 0
TST_X.loc[rows_TST, 'lastOrder'] = 0

```

Criando novas variáveis

Conforme visto no Capítulo 3, de uma forma geral, a tarefa de construção de novas variáveis é muito mais dependente do conhecimento do domínio do que a construção de um classificador. Desta forma, esse processo vai depender de cada tipo de projeto. Para o nosso projeto, vamos exemplificar esse processo construindo três variáveis:

- QtdTransacoes: quantidade de transações da sessão;
- MaxbCount: maior valor de bCount por sessão;
- FlagCustomer: Variável booleana informando se é um cliente, isto é, se foi informado customerNo.

É importante lembrar que algumas informações foram perdidas quando mudamos a granularidade de transação para última transação da sessão. O objetivo dessas novas variáveis é recuperar um pouco dessas informações. Para criar essas variáveis seguiremos os seguintes passos:

- **Passo 1:** tratar os valores ausentes na base de dados original, pois tratamos apenas na base de dados da granularidade do projeto. E a base de dados original será usada para calcular as novas variáveis;
- **Passo 2:** definir o index da base de dados para poder usar a função o *group by*;
- **Passo 3:** criar a variável QtdTransacoes;
- **Passo 4:** criar a variável MaxbCount;
- **Passo 5:** criar a variável FlagCustomer.

```
# Passo 1
replaceValueMissing('?', l_varNumeric, TRN_Original)
replaceValueMissing('?', l_varString, TRN_Original)
replaceValueMissing('?', l_varNumeric, TST_Original)
replaceValueMissing('?', l_varString, TST_Original)

convertFloat(l_varNumeric, TRN_Original)
convertFloat(l_varNumeric, TST_Original)
```

```
# Passo 2
# Conjunto de treinamento
TRN_X = TRN_X.set_index('sessionNo')
# Conjunto de teste
TST_X = TST_X.set_index('sessionNo')
```

```
# Passo 3
# Conjunto de treinamento
TRN_X['QtdTransacoes'] = TRN_Original.groupby('sessionNo').duration.count()
# Conjunto de teste
TST_X['QtdTransacoes'] = TST_Original.groupby('sessionNo').duration.count()
```

```
# Passo 4
# Conjunto de treinamento
TRN_X['MaxbCount'] = TRN_Original.groupby('sessionNo').bCount.max()
# Conjunto de teste
TST_X['MaxbCount'] = TST_Original.groupby('sessionNo').bCount.max()
```

```
# Passo 5
# Conjunto de treinamento
TRN_X['FlagCustomer'] = TRN_X.apply(lambda row: 0 if row.customerNo=="?" else 1, axis=1)
# Conjunto de teste
TST_X['FlagCustomer'] = TST_X.apply(lambda row: 0 if row.customerNo=="?" else 1, axis=1)
```

Selecionando Variáveis - Parte 1

Algumas variáveis que são identificadores únicos, em geral, que representam chaves primárias em banco de dados devem ser excluídas pois não agregam informações, como exemplo podemos citar o número do CPF. Para nosso exemplo prático, temos a variável código do cliente `customerNo`. Por isso, vamos excluí-la. É importante notar que essa é uma decisão que depende do conhecimento do domínio!

```
# Conjunto de treinamento
TRN_X = TRN_X.drop(['customerNo'], axis = 1)
# Conjunto de teste
TST_X = TST_X.drop(['customerNo'], axis = 1)
```

5.2.3 Transformação

Este seção foi escrita no Jupyter-notebook e contém a descrição e o código para realização da etapa de Transformação do projeto. Todos os modelos que serão usados neste curso aceitam apenas variáveis numéricas como entrada no *scikit-learn*. Para realizarmos a etapa de Transformação, seguiremos os seguintes passos:

- **Passo 1:** criar variáveis *Dummies*;
- **Passo 2:** normalizar as variáveis.

Mais informações sobre essas duas transformações estão disponíveis no final do Capítulo 3.

```
# Passo 1
# Conjunto de treinamento
TRN_X = pd.get_dummies(TRN_X, prefix_sep='_')
# Conjunto de teste
TST_X = pd.get_dummies(TST_X, prefix_sep='_')
```

O processo de construção de variáveis Dummies pode tornar o **número de colunas diferentes entre os conjuntos de treinamento e teste**, por isso é preciso que seja analisado essa situação. Isso pode ocorrer quando existem valores de variáveis que ocorrem em apenas um dos conjuntos. O código para essa verificação, as funções `get_Min_Max()` e `normalize()` estão disponíveis no Github do Livro⁵.

```
# Passo 2
norm_min_max = get_Min_Max(TRN_X)
# Conjunto de treinamento
TRN_X = normalize(TRN_X, norm_min_max)
# Conjunto de teste
TST_X = normalize(TST_X, norm_min_max)
```

Selecionando Variáveis - Parte 2

Para eliminar as variáveis com valores constantes seguiremos os seguintes passos:

- **Passo 1:** informar o limiar de variância - abaixo ou igual a esse limiar a variável será excluída dos dois conjuntos;
- **Passo 2:** criar uma lista com as variáveis do conjunto de treinamento que tem variância inferior ou igual ao `threshold_var`;
- **Passo 3:** eliminar as variáveis constante dos dois conjuntos.

⁵<https://github.com/osalvone/0>

```
# Passo 1
threshold_var = 0

# Passo 2
l_var = [x for x in TRN_X.columns if TRN_X[x].var() <= threshold_var]

# Passo 3
for v in l_var:
    TRN_X = TRN_X.drop([v], axis = 1)
    TST_X = TST_X.drop([v], axis = 1)
```

Salvando a bases de dados

A última parte da etapa de Transformação dos Dados é salvá-los para que os mesmos possam ser usados pela etapa de *Data Mining*.

```
TRN_X.to_csv('TRN_X.csv', index=False)

TRN_Y.to_csv('TRN_Y.csv', index=False)

TST_X.to_csv('TST_X.csv', index=False)

TST_Y.to_csv('TST_Y.csv', index=False)
```

5.2.4 Data Mining

Este seção foi escrita no Jupyter-notebook e contém a descrição e o código para realização da etapa de *Data Mining* do projeto.

Importando as bibliotecas

```
import numpy as np
import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score
import sklearn.metrics as metrics

import matplotlib.pyplot as plt
```

Carregando os dados de treinamento

```
TRN_X = pd.read_csv('TRN_X.csv')
TRN_Y = pd.read_csv('TRN_Y.csv')
```

Carregando os dados de teste

```
TST_X = pd.read_csv('TST_X.csv')
TST_Y = pd.read_csv('TST_Y.csv')
```

Classificador KNN

Nós vamos usar o *K-Nearest Neighbor* (KNN) como classificador. Lembrar que o valor K representa a quantidade de exemplos semelhantes do conjunto de treinamento utilizado para classificação. Para mais detalhes ver o Capítulo 2.

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(TRN_X, TRN_Y.order.to_numpy())
```

5.2.5 Avaliação

Conforme visto no Capítulo 4, avaliar o desempenho de um classificador com o mesmo conjunto utilizado na sua construção, não fornece uma boa estimativa de como será a sua predição em dados nunca antes vistos, pois a estimativa será otimista. Para este exemplo, estamos usando *Hold-out* como método de avaliação e as seguintes métricas de avaliação:

- Matriz de Confusão;
- Acurácia;
- Precisão;
- Sensibilidade;
- Especificidade;
- Área sob a curva ROC.

Para realizarmos a avaliação seguiremos os seguintes passos:

- **Passo 1:** recuperar a resposta do modelo para o conjunto de teste e armazenar em um vetor;
- **Passo 2:** gerar a Matriz de Confusão (linhas = real | colunas = previsto);
- **Passo 3:** recuperar os valores da matriz de confusão.

```
# Passo 1
pred = knn.predict(TST_X)
```

```
# Passo 2
cm = confusion_matrix(TST_Y.prediction.to_numpy(), pred)
df_cm = pd.DataFrame(cm)
# Mudando a ordem para vir primeiro 1 e depois 0
df_cm = df_cm[[1, 0]]
df_cm = df_cm.sort_index(ascending=False)
df_cm
```

```
# Passo 3
# Verdadeiro positivo
a = df_cm.iloc[0,0]

# Falso positivo
b = df_cm.iloc[1,0]

# Falso negativo
c = df_cm.iloc[0, 1]

# Verdadeiro negativo
d = df_cm.iloc[1, 1]
```

Calculando a acurácia do modelo

```
(a + d) / (a + b + c + d)
```

Calculando a precisão do modelo

```
a / (a + b)
```

Calculando a sensibilidade do modelo

```
a / (a + c)
```

calculando a especificidade do modelo

```
d / (b + d)
```

Calculando a área sob a curva ROC

```
roc_auc = roc_auc_score(TST_Y.prediction.to_numpy(), pred)
roc_auc
```

Gerando o gráfico da área sob a curva ROC

```
fpr, tpr, _ = metrics.roc_curve(TST_Y.prediction.to_numpy(), pred)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.3f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Referências Bibliográficas

- [Ade+08] P. Adeodato et al. “The Power of Sampling and Stacking for the PaKDD-2007 Cross-Selling Problem”. In: *Int. J. Data Warehous. Min.* 4.2 (2008), pages 22–31. DOI: 10.4018/jdwm.2008040104. URL: <https://doi.org/10.4018/jdwm.2008040104> (cited on page 73).
- [Ade+10] P. Adeodato et al. “The Power of Sampling and Stacking for the PAKDD-2007 Cross-Selling Problem”. In: *Strategic Advancements in Utilizing Data Mining and Warehousing Technologies: New Concepts and Developments*. Edited by David Taniar and Laura Irina Rusu. IGI Global, 2010, pages 297–306. URL: <http://www.igi-global.com/Bookstore/chapter.aspx?titleid=40412> (cited on page 73).
- [Ahn13] Bieng Chearl Ahn. “Numerical Implementation of the Heaviside Step Function”. In: *Journal of the Research Institute for Computer and Information Communication* 2.21 (2013), pages 5–8 (cited on page 44).
- [And11] Tomohiro Ando. “Predictive Bayesian Model Selection”. In: *American Journal of Mathematical and Management Sciences* 31.1-2 (2011), pages 13–38. DOI: 10.1080/01966324.2011.10737798. URL: <https://doi.org/10.1080/01966324.2011.10737798> (cited on page 48).
- [Arg+09] Shlomo Argamon et al. “Automatically profiling the author of an anonymous text”. In: *Communications of the ACM* 52.2 (2009), pages 119–123. ISSN: 0001-0782 (cited on page 60).
- [BJ90] R. Beale and T. Jackson. *Neural Computing - An Introduction*. Taylor & Francis, 1990. ISBN: 9781420050431 (cited on pages 32, 34).
- [Bel08] VAISHAK Belle. “Detection and recognition of human faces using random forests for a mobile robot”. In: *Master of Science Thesis, Academic Knowledge-based Systems Group* (2008) (cited on page 54).

- [Ber11] José M. Bernardo. “Modern Bayesian Inference: Foundations and Objective Methods”. In: *Philosophy of Statistics*. Edited by Prasanta S. Bandyopadhyay and Malcolm R. Forster. Volume 7. Handbook of the Philosophy of Science. Amsterdam: North-Holland, 2011, pages 263–306. DOI: <https://doi.org/10.1016/B978-0-444-51862-0.50008-3>. URL: <http://www.sciencedirect.com/science/article/pii/B9780444518620500083> (cited on page 48).
- [Bey+99] Kevin S. Beyer et al. “When Is ”Nearest Neighbor” Meaningful?” In: *Proceedings of the 7th International Conference on Database Theory*. ICDT ’99. London, UK, UK: Springer-Verlag, 1999, pages 217–235. ISBN: 3-540-65452-6 (cited on page 29).
- [BG92] James L. Blue and Patrick J. Grother. “Training Feed-Forward Neural Networks Using Conjugate Gradients”. In: *SPIE*. 1992, pages 179–190 (cited on pages 33, 34).
- [BKL99] Avrim Blum, Adam Kalai, and John Langford. “Beating the Hold-out: Bounds for K-Fold and Progressive Cross-Validation”. In: *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*. COLT ’99. Santa Cruz, California, USA: Association for Computing Machinery, 1999, pages 203–208. ISBN: 1581131674. DOI: 10.1145/307400.307439. URL: <https://doi.org/10.1145/307400.307439> (cited on page 69).
- [Bou08] M. Boulle. “An efficient parameter-free method for large scale offline learning”. In: *International Conference on Machine Learning (ICML 2008)*. 2008, pages 567–570 (cited on page 49).
- [BLC00] Antonio de Pádua Braga, Teresa Bernarda Ludermir, and André Carlos Ponce de Leon Ferreira Carvalho. *Redes neurais artificiais: teoria e aplicações*. LTC, 2000 (cited on page 36).
- [Bre96] Leo Breiman. “Bagging Predictors”. In: *Mach. Learn.* 24.2 (1996), pages 123–140. ISSN: 0885-6125. DOI: 10.1023/A:1018054314350. URL: <http://dx.doi.org/10.1023/A:1018054314350> (cited on pages 49, 53).
- [Bre01] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pages 5–32 (cited on page 53).
- [Bro+95] M. Brown et al. “High dimensional neurofuzzy systems: overcoming the curse of dimensionality”. In: *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*. Volume 4. 1995, pages 2139–2146. DOI: 10.1109/FUZZY.1995.409976 (cited on page 64).
- [Bru+20] L. Brunese et al. “Lung Cancer Detection and Characterisation through Genomic and Radiomic Biomarkers”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pages 1–8. DOI: 10.1109/IJCNN48605.2020.9206797 (cited on page 54).
- [BK16] Robert J. Brunner and Edward J. Kim. “Teaching Data Science”. In: *Procedia Computer Science* 80 (2016). International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA, pages 1947–1956. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2016.05.513>. URL: <http://www.sciencedirect.com/science/article/pii/S1877050916310006> (cited on page 77).

- [Cao18] Longbing Cao. “What Is Data Science”. In: *Data Science Thinking: The Next Scientific, Technological and Economic Revolution*. Cham: Springer International Publishing, 2018, pages 29–58. ISBN: 978-3-319-95092-1. DOI: 10.1007/978-3-319-95092-1_2. URL: https://doi.org/10.1007/978-3-319-95092-1_2 (cited on page 15).
- [Cat11] Rick Cattell. “Scalable SQL and NoSQL Data Stores”. In: *SIGMOD Rec.* 39.4 (2011), pages 12–27 (cited on page 17).
- [Cha+17] Gabriel Chartrand et al. “Deep Learning: A Primer for Radiologists”. In: *RadioGraphics* 37.7 (2017). PMID: 29131760, pages 2113–2131. DOI: 10.1148/rg.2017170077. eprint: <https://doi.org/10.1148/rg.2017170077>. URL: <https://doi.org/10.1148/rg.2017170077> (cited on page 23).
- [Con12] Drew Conway. *The data science venn diagram. Dataists [Web page]*. Acessado: 02/02/2021. Disponível em: <http://www.dataists.com/2010/09/the-data-science-venn-diagram/>. 2012 (cited on pages 19, 20).
- [CH06] T. Cover and P. Hart. “Nearest Neighbor Pattern Classification”. In: *IEEE Trans. Inf. Theor.* 13.1 (2006), pages 21–27. ISSN: 0018-9448. DOI: 10.1109/TIT.1967.1053964. URL: <http://dx.doi.org/10.1109/TIT.1967.1053964> (cited on page 28).
- [DLY97] M. Dash, H. Liu, and J. Yao. “Dimensionality reduction of unsupervised data”. In: *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*. 1997, pages 532–539. DOI: 10.1109/TAI.1997.632300 (cited on page 64).
- [DN17] Luis Dias and Rosaldo F. O. Neto. “Tensorflow vs R: A Comparative Study of Usability”. In: *Proceedings of the 13th Brazilian Symposium on Information Systems, SBSI 2017, Lavras, Brazil, June 5-8, 2017*. Edited by Heitor Costa et al. 2017, pages 96–99. DOI: 10.5753/sbsi.2017.6093. URL: <https://doi.org/10.5753/sbsi.2017.6093> (cited on page 17).
- [Dom18] Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. USA: Basic Books, Inc., 2018. ISBN: 0465094279 (cited on page 27).
- [dos+20] Daniel dos Santos Costa et al. “IRIS-GRAPE: An approach for prediction of quality attributes in vineyard grapes inspired by iris biometric recognition”. In: *Computers and Electronics in Agriculture* 168 (2020), page 105140. ISSN: 0168-1699. DOI: <https://doi.org/10.1016/j.compag.2019.105140>. URL: <https://www.sciencedirect.com/science/article/pii/S0168169918315436> (cited on page 63).
- [FFW11] Jianqing Fan, Yingying Fan, and Yichao Wu. “High-Dimensional Classification”. In: *High-Dimensional Data Analysis*. World Scientific, 2011. Chapter 1, pages 3–37 (cited on page 29).
- [Fay+96] Usama Fayyad et al. “The KDD Process for Extracting Useful Knowledge from Volumes of Data”. In: *Communications of the ACM* 39 (1996), pages 27–34 (cited on page 18).
- [Fer+18] Alberto Fernández et al. “Introduction to KDD and Data Science”. In: *Learning from Imbalanced Data Sets*. Cham: Springer International Publishing, 2018, pages 1–17. ISBN: 978-3-319-98074-4. DOI: 10.1007/978-3-319-98074-4_1. URL: https://doi.org/10.1007/978-3-319-98074-4_1 (cited on page 16).

- [FFJ19] J. P. Figueirôa Nascimento, R. Ferreira de Oliveira Neto, and L. Júnior Macário Amorim. “An Efficient Kick Strategy for Agents in the 2D Simulation League”. In: *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. 2019, pages 461–466. DOI: 10.1109/BRACIS.2019.00087 (cited on pages 62, 65).
- [Fin13] William Finzer. “The Data Science Education Dilemma”. In: *Technology Innovations in Statistics Education* 7 (2013) (cited on page 19).
- [FS96] Yoav Freund and Robert E. Schapire. “Experiments with a New Boosting Algorithm”. In: *Proceedings of the Thirteenth International Conference*. Bari, Italy: ICML, 1996, pages 148–156 (cited on page 49).
- [GH15] Amir Gandomi and Murtaza Haider. “The hype: big data concepts, methods, and analytics”. In: *International Journal of Information Management* 35.2 (2015), pages 137–144 (cited on page 16).
- [GH06] Andrew Gelman and Jennifer Hill. “Missing-data imputation”. In: *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. <https://doi.org/10.1017/CBO9780511790942.031>. Cambridge University Press, 2006, pages 529–544. DOI: <https://doi.org/10.1017/CBO9780511790942.031> (cited on page 61).
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618 (cited on page 17).
- [Hal98] Mark A. Hall. “Correlation-based feature selection for machine learning”. PhD thesis. University of Waikato, 1998 (cited on page 66).
- [HH03] Mark A. Hall and Geoffrey Holmes. “Benchmarking Attribute Selection Techniques for Discrete Class Data Mining”. In: *IEEE Transactions On Knowledge And Data Engineering* 15 (2003), pages 1437–1447 (cited on page 64).
- [HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer, 2009. ISBN: 9780387848587 (cited on page 53).
- [Hay98] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN: 0132733501 (cited on page 32).
- [HZN16] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui. “Deep Learning of Part-Based Representation of Data Using Sparse Autoencoders With Nonnegativity Constraints”. In: *IEEE Transactions on Neural Networks and Learning Systems* 27.12 (2016), pages 2486–2498. DOI: 10.1109/TNNLS.2015.2479223 (cited on page 23).
- [HA18] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. 2nd. Australia: OTexts, 2018. URL: <https://otexts.com/fpp2/> (cited on page 74).
- [JDM00] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. “Statistical Pattern Recognition: A Review”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.1 (2000), pages 4–37. ISSN: 0162-8828. DOI: 10.1109/34.824819. URL: <http://dx.doi.org/10.1109/34.824819> (cited on page 70).
- [KR99] M. Kearns and D. Ron. “Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation”. In: *Neural Computation* 11.6 (1999), pages 1427–1453. DOI: 10.1162/089976699300016304 (cited on page 70).

- [KHS10] Rehanullah Khan, Allan Hanbury, and Julian Stoettinger. “Skin detection: A random forest approach”. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE. 2010, pages 4613–4616 (cited on page 53).
- [Koh95] Ron Kohavi. “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection”. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2. IJCAI'95*. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995, pages 1137–1143. ISBN: 1558603638 (cited on pages 69, 70).
- [KD19a] Vijay Kotu and Bala Deshpande. “Chapter 1 - Introduction”. In: *Data Science (Second Edition)*. Edited by Vijay Kotu and Bala Deshpande. Second Edition. Morgan Kaufmann, 2019, pages 1–18. ISBN: 978-0-12-814761-0. DOI: <https://doi.org/10.1016/B978-0-12-814761-0.00001-0>. URL: <http://www.sciencedirect.com/science/article/pii/B9780128147610000010> (cited on page 16).
- [KD19b] Vijay Kotu and Bala Deshpande. “Chapter 2 - Data Science Process”. In: *Data Science (Second Edition)*. Edited by Vijay Kotu and Bala Deshpande. Second Edition. Morgan Kaufmann, 2019, pages 19–37. ISBN: 978-0-12-814761-0. DOI: <https://doi.org/10.1016/B978-0-12-814761-0.00002-2>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128147610000022> (cited on page 61).
- [KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Commun. ACM* 60.6 (2017), pages 84–90. DOI: 10.1145/3065386. URL: <http://doi.acm.org/10.1145/3065386> (cited on page 17).
- [Lec+98] Y. LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pages 2278–2324. DOI: 10.1109/5.726791 (cited on page 17).
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. “Deep learning”. In: *Nat.* 521.7553 (2015), pages 436–444. DOI: 10.1038/nature14539. URL: <https://doi.org/10.1038/nature14539> (cited on page 17).
- [LI11] Amadeu N. Lima and Joshua O. Imoniana. “Um estudo sobre a importância do uso das ferramentas de controle gerencial nas micro, pequenas e médias empresas industriais no município de São Caetano do Sul”. In: *Revista da micro e pequena empresa* 2.1 (2011), pages 28–48. DOI: <https://doi.org/10.6034/30>. URL: <http://www.cc.faccamp.br/ojs-2.4.8-2/index.php/RMPE/article/view/30> (cited on page 24).
- [Llo82] Stuart P. Lloyd. “Least squares quantization in PCM”. In: *IEEE Trans. Inf. Theory* 28.2 (1982), pages 129–136. DOI: 10.1109/TIT.1982.1056489. URL: <https://doi.org/10.1109/TIT.1982.1056489> (cited on page 23).
- [MH82] Edward R. Mansfield and Billy P. Helms. “Detecting Multicollinearity”. In: *The American Statistician* 36.3a (1982), pages 158–160. DOI: 10.1080/00031305.1982.10482818. URL: <https://doi.org/10.1080/00031305.1982.10482818> (cited on page 67).
- [MP43] Warren McCulloch and Walter Pitts. “A Logical Calculus of Ideas Immanent in Nervous Activity”. In: *Bulletin of Mathematical Biophysics* 5 (1943), pages 127–147 (cited on page 36).

- [MO18] R. de L. Mendes and R.F. Oliveira Neto. “The Power of Ensemble Models in Fingerprint Classification: A case study”. In: *INFOCOMP Journal of Computer Science* 17.1 (2018), pages 1–10 (cited on page 49).
- [MP69] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969 (cited on page 41).
- [Mit97] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2 (cited on pages 23, 25).
- [NN19] Delmiro Neto and Rosalvo Neto. “Estudo comparativo entre dois métodos de localização da fronteira externa da íris: um estudo de caso”. In: *Anais da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe*. Ilhéus: SBC, 2019, pages 225–234. URL: <https://sol.sbc.org.br/index.php/erbase/article/view/8983> (cited on pages 23, 64).
- [NAS17] Rosalvo Neto, Paulo Jorge Adeodato, and Ana Carolina Salgado. “A framework for data transformation in Credit Behavioral Scoring applications based on Model Driven Development”. In: *Expert Systems with Applications* 72 (2017). <https://doi.org/10.1016/j.eswa.2016.10.059>, pages 293–305. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2016.10.059> (cited on pages 57, 58).
- [Net+12] Rosalvo F. O. Neto et al. “Estudo Comparativo entre Proposicionalização e Mineração de Dados Multidimensional sobre um Banco de Dados Relacional”. In: *XXVII Simpósio Brasileiro de Banco de Dados - Short Papers, São Paulo, São Paulo, Brasil, October 15-18, 2012*. 2012, pages 240–247. URL: <http://www.1bd.dcc.ufmg.br/colecoes/sbbd/2012/0031.pdf> (cited on page 58).
- [NXC09] E. W. T. Ngai, Li Xiu, and D. C. K. Chau. “Review: Application of Data Mining Techniques in Customer Relationship Management: A Literature Review and Classification”. In: *Expert Syst. Appl.* 36.2 (2009), pages 2592–2602. ISSN: 0957-4174 (cited on page 27).
- [OOQ20] R. Oliveira Neto, R. Oliveira, and A. Queiroz. “Investigating the influence of groups of variables on the task of predicting the age of an author in blog posts”. In: *IEEE Latin America Transactions* 18.05 (2020), pages 838–844. DOI: [10.1109/TLA.2020.9082911](https://doi.org/10.1109/TLA.2020.9082911) (cited on page 60).
- [OBC13] R. F. OLIVEIRA NETO, R. M. BOKOWSKI SOBRINHO, and A. M. CAVALCANTI. “Estudo comparativo entre modelos de classificação para Behavior Scoring em procedimentos de análise de risco de crédito”. In: *XXXVIII Encontro da ANPAD, Rio de Janeiro, Brazil*. 2013 (cited on page 58).
- [OMF17] R. F. OLIVEIRA NETO, C. C. L. MARIANO, and M. S. R. FARIAS. “Horário Universitário Personalizado por Metaheurística: Um Estudo de Caso”. In: *XLIX Simpósio Brasileiro de Pesquisa Operacional (SBPO), Blumenau, Santa Catarina, Brazil*. 2017 (cited on page 65).
- [Oli16] Rosalvo Oliveira Neto. “CoMoVi: um Framework para Transformação de Dados em Aplicações de Credit Behavior Scoring baseado no Desenvolvimento Dirigido por Modelos.” PhD thesis. Universidade Federal de Pernambuco, 2016 (cited on pages 19, 20, 67).

- [Oli08] Rosaldo de Oliveira Neto. “Investigação sobre o efeito de ruído na generalização de redes neurais sem peso em problemas de classificação binária”. Master’s thesis. Brasil: Universidade Federal de Pernambuco, 2008. URL: <https://repositorio.ufpe.br/handle/123456789/2115> (cited on page 35).
- [ORS19] Rosaldo de Oliveira Neto, Ricardo Ramos, and Cleidson da Silva. “Uma solução de mineração de dados para concessão de cupons de descontos em comércio eletrônico: um estudo de caso”. In: *Revista Brasileira de Computação Aplicada* 11.3 (2019), pages 122–132. DOI: 10.5335/rbca.v11i3.9077. URL: <http://seer.upf.br/index.php/rbca/article/view/9077> (cited on pages 31, 58).
- [OSC13] Rosaldo Ferreira de Oliveira Neto, Roberto Bokowski Sobrinho, and Andre Marques Cavalcanti. “Estudo comparativo entre modelos de classificação para Behavior Scoring em procedimentos de análise de risco de crédito”. In: *EnANPAD*. Rio de Janeiro, Brasil, 2013 (cited on page 29).
- [PLC08] Jin Hee Park, Won Hee Lee, and Hae Soo Chung. “Incidence and risk factors of breast cancer lymphoedema”. In: *Journal of Clinical Nursing* 17.11 (2008), pages 1450–1459. DOI: <https://doi.org/10.1111/j.1365-2702.2007.02187.x> (cited on page 62).
- [PF13] Foster Provost and Tom Fawcett. “Data Science and its Relationship to Big Data and Data-Driven Decision Making”. In: *Big Data* 1.1 (Mar. 2013), pages 51–59. DOI: 10.1089/big.2013.1508. URL: <https://doi.org/10.1089%2Fbig.2013.1508> (cited on pages 15, 16).
- [Qin97] S.Joe Qin. “Chapter 8 - Neural Networks for Intelligent Sensors and Control — Practical Issues and Some Solutions”. In: *Neural Systems for Control*. Edited by Omid Omidvar and David L. Elliott. San Diego: Academic Press, 1997, pages 213–234. ISBN: 978-0-12-526430-3. DOI: <https://doi.org/10.1016/B978-012526430-3/50009-X>. URL: <https://www.sciencedirect.com/science/article/pii/B978012526430350009X> (cited on page 62).
- [Rei+20] Daniele Reis et al. “A data mining approach for prediction of quality attributes in Palmer mango from images”. In: *Revista Brasileira de Computação Aplicada* 12.2 (June 2020), pages 54–66. DOI: 10.5335/rbca.v12i2.10866. URL: <http://seer.upf.br/index.php/rbca/article/view/10866> (cited on page 63).
- [Rib+17] Jardel Ribeiro et al. “NoSQL vs relational database: A comparative study about the generation of the most frequent N-grams”. In: *4th International Conference on Systems and Informatics, ICSAI 2017, Hangzhou, China, November 11-13, 2017*. IEEE, 2017, pages 1568–1572. DOI: 10.1109/ICSAI.2017.8248535. URL: <https://doi.org/10.1109/ICSAI.2017.8248535> (cited on page 16).
- [Ros58] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6 (1958), pages 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: <http://dx.doi.org/10.1037/h0042519> (cited on page 36).

- [RM86] David Rumelhart and James McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Edited by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. ISBN: 0-262-68053-X (cited on pages 32, 34, 43).
- [Rus+96] Stuart J. Russell et al. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN: 0-13-103805-2 (cited on pages 24–26, 32).
- [San20] W. B. S. Santos. *Análise de métricas de avaliação de desempenho em modelos de mineração de dados*. Monografia (Graduação em Engenharia de Computação), UNIVASF (Universidade Federal do Vale do São Francisco), Juazeiro, Brazil. 2020 (cited on page 74).
- [Sch13] Robert E Schapire. “Explaining adaboost”. In: *Empirical inference*. Springer, 2013, pages 37–52 (cited on page 51).
- [SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014, pages 1–397. ISBN: 978-1-10705713-5 (cited on pages 23, 24).
- [She09] Tania Shepherd. “O estatuto da Linguística de corpus: metodologia ou área da Linguística?” In: *Matraga - Revista do Programa de Pós-Graduação em Letras da UERJ* 16.24 (2009). ISSN: 2446-6905 (cited on page 60).
- [SGR11] Aline Cristina Soterroni, Roberto Luiz Galski, and Fernando Manuel Ramos. “The q-Gradient Vector for Unconstrained Continuous Optimization Problems”. In: *Operations Research Proceedings 2010*. Edited by Bo Hu et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pages 365–370 (cited on page 38).
- [Spa03] James C. Spall. *Introduction to Stochastic Search and Optimization*. 1st edition. USA: John Wiley, 2003. ISBN: 0471330523 (cited on page 39).
- [Sui57] Daniel B. Suits. “Use of Dummy Variables in Regression Equations”. In: *Journal of the American Statistical Association* 52.280 (1957), pages 548–551. DOI: 10.1080/01621459.1957.10501412. URL: <https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1957.10501412> (cited on page 67).
- [TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321321367 (cited on page 31).
- [Tho00] Lyn Thomas. “A survey of credit and behavioral scoring: forecasting financial risk of lending to consumers”. In: *International Journal of Forecasting* (2000), pages 149–172 (cited on page 57).
- [TA77] Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of ill-posed problems*. Washington, D.C.: John Wiley & Sons, New York: V. H. Winston & Sons, 1977, page 258 (cited on page 35).
- [TPS08] A. Townsend, M. Papeş, and J. Soberón. “Rethinking receiver operating characteristic analysis applications in ecological niche modeling”. In: *Ecological Modelling* 213.1 (2008), pages 63–72. ISSN: 0304-3800. DOI: <https://doi.org/10.1016/j.ecolmodel.2007.11.008>. URL: <https://www.sciencedirect.com/science/article/pii/S0304380007006163> (cited on page 73).

- [Tre+14] P. Trebuña et al. “The importance of normalization and standardization in the process of clustering”. In: *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 2014, pages 381–385. DOI: 10.1109/SAMI.2014.6822444 (cited on page 67).
- [Tsa+15] Chun-Wei Tsai et al. “Big data analytics: a survey”. In: *Journal of Big Data* 2.1 (2015), pages 1–32 (cited on page 16).
- [Wer74] P. J. Werbos. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”. PhD thesis. Harvard University, 1974 (cited on page 44).
- [Wid05] B. Widrow. “Thinking about thinking: the discovery of the LMS algorithm”. In: *IEEE Signal Processing Magazine* 22.1 (2005), pages 100–106. DOI: 10.1109/MSP.2005.1407720 (cited on page 39).
- [WH60] Bernard Widrow and Marcian E. Hoff. “Adaptive Switching Circuits”. In: *1960 IRE WESCON Convention Record, Part 4*. New York: IRE, 1960, pages 96–104 (cited on pages 38, 39).
- [WH88] Bernard Widrow and Marcian E. Hoff. “Neurocomputing: Foundations of Research”. In: *Adaptive Switching Circuits*. Cambridge, MA, USA: MIT Press, 1988. Chapter Adaptive Switching Circuits, pages 123–134. ISBN: 0-262-01097-6. URL: <http://dl.acm.org/citation.cfm?id=65669.104390> (cited on pages 34, 36).
- [WFH11] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN: 0123748569, 9780123748560 (cited on pages 23, 31, 54).
- [Wol92] David H. Wolpert. “Stacked generalization”. In: *Neural Networks* 5 (1992), pages 241–259 (cited on page 49).
- [ZS04] H. Zhang and Shengli Sheng. “Learning weighted naive Bayes with accurate ranking”. In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*. 2004, pages 567–570. DOI: 10.1109/ICDM.2004.10030 (cited on page 48).
- [ZM14] Nina Zumel and John Mount. *Practical Data Science with R*. 1st. USA: Manning Publications Co., 2014. ISBN: 1617291560 (cited on pages 15, 18, 19).

Apêndice A – Perfil das vagas de cientista de dados

Este apêndice apresenta o resultado da busca por vagas de cientista de dados no portal [LinkedIn](#). A busca foi realizada no dia 28/09/2020. Foram selecionados os seis primeiros resultados da busca. O nome das empresas não são exibidos porque não foi solicitado autorização. No entanto, essa informação não é relevante para extração do perfil das vagas.

Vaga	1
Setor	Indústria química
Tamanho da empresa	5.001-10.000 funcionários
País	Estados Unidos
Habilidades para a vaga	<ul style="list-style-type: none">Desenvolvendo modelos preditivos na plataforma da Microsoft Azure;Escrever rotinas de preparação e análise de dados em Python usando as bibliotecas pandas e scikit-learnParticipar do projeto de banco de dados e discussões de modelagem estatística;

Vaga	2
Setor	Recursos humanos
Tamanho da empresa	10 funcionários
País	Brasil
Habilidades para a vaga	<ul style="list-style-type: none">Python, Alteryx e RStudioREstatística, modelagem de dados, e programação

Vaga	3
Setor	Tecnologia da informação e serviços
Tamanho da empresa	11-50 funcionários
Pais	Colômbia
Habilidades para a vaga	<ul style="list-style-type: none"> • Criar rotinas de acesso, limpeza e integração de dados (R - Python - SAS - SQL) • Implementar algoritmos descritivos, preditivos e de otimização para resolver os desafios de negócios (R- Python - SAS) • Gerar relatórios de alto impacto e visualizações tanto estáticas (Power Point) como dinâmicas (Power Bi - Tableau). • Gerar documentação de trabalho sob metodologias padrão (Crisp-DM) • Trabalho em equipe.

Vaga	4
Setor	Lazer, viagens e turismo
Tamanho da empresa	5.001-10.000 funcionários
Pais	Brasil
Habilidades para a vaga	<ul style="list-style-type: none"> • Aprendizado de Máquina e Mineração de Dados; • Inteligência Artificial; • Bancos de Dados e Gerenciamento de Dados; • Otimização e Simulação; • Estatística; • Algoritmos e Programação Paralela; • Arquiteturas de Computador, Sistemas Operacionais, Redes de Computadores e Sistemas de Armazenamento de Dados; • Computação Gráfica e Visualização de Dados.

Vaga	5
Setor	Varejo
Tamanho da empresa	1.001-5.000 funcionários
País	México
Habilidades para a vaga	<ul style="list-style-type: none"> Desenvolvimento de soluções que incluem modelos matemáticos, aprendizagem de máquina e IA. Lidar com grandes volumes de dados. Desenvolver métricas para monitoramento e manutenção de modelos. Conhecimento avançado em Python, SQL e Spark. Desejável: Hive, Hadoop, Tableau, Power BI.

Vaga	6
Setor	Materiais esportivos
Tamanho da empresa	+ de 10.001 funcionários
País	Alemanha
Habilidades para a vaga	<ul style="list-style-type: none"> Experiência prática em aprendizado de máquina, inteligência artificial e estatística; Talentos que podem guiar projetos de ciência dos dados de aplicação de Pesquisa e Desenvolvimento até colocar em produção; Experiência na solução de problemas complexos de negócios por meio de algoritmos Talentos que promovem a colaboração entre equipes, projetos e comunicação