UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIEL HENRIQUE DE ALENCAR MORTARI

ANÁLISE DE DADOS RELACIONADOS A PARTIDAS DE FUTEBOL UTILIZANDO TÉCNICAS DE MINERAÇÃO

PATO BRANCO

2022

GABRIEL HENRIQUE DE ALENCAR MORTARI

ANÁLISE DE DADOS RELACIONADOS A PARTIDAS DE FUTEBOL UTILIZANDO TÉCNICAS DE MINERAÇÃO

ANALYSIS OF DATA RELATED TO FOOTBALL MATCHES USING MINING TECHNIQUES

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Dalcimar Casanova

Coorientador: Prof. Dr. Pablo Gauterio Cavalcanti

PATO BRANCO

2022

4.0 Internacional

Esta licença permite remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es) e que licenciem as novas criações sob termos idênticos. Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

GABRIEL HENRIQUE DE ALENCAR MORTARI

ANÁLISE DE DADOS RELACIONADOS A PARTIDAS DE FUTEBOL UTILIZANDO TÉCNICAS DE MINERAÇÃO

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 27/junho/2022

Prof. Dr. Dalcimar Casanova
Doutorando em Física Computacional
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Prof. Dr. Ives Renê Venturini Pola
Doutorado em Ciências da Computação e Matemática Computacional
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Prof. Dr. Marcelo Teixeira
Doutorado em Engenharia de Automação e Sistemas
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

PATO BRANCO
2022

RESUMO

Diversos dados podem ser analisados sobre partidas de futebol, como o desempenho das equipes no campeonato, histórico contra a equipe adversária, jogadores lesionados ou suspensos. Porém, ao analisar o futebol no Brasil, outros dados devem ser levados em consideração, como a distância que as equipes precisam viajar e os diferentes climas aqui presente. Esse trabalho realizou a aquisição de dados do campeonato brasileiro, juntamente com as distâncias percorridas pelas equipes e do tempo no dia dos jogos. A partir dessas informações, foi construída uma base com vários dados de desempenho, distância e tempo sobre cada partida disputada no campeonato brasileiro e foi desenvolvido um classificador que realiza a predição dos resultados finais dos jogos, o qual obteve uma acurácia de 50%. Com isso, foi possível realizar uma análise do real impacto de atributos relacionados a tempo e distância no resultado das partidas que constatou que tais atributos não são substancialmente relevantes para determinar o placar final.

Palavras-chave: futebol; mineração de dados; análise de dados.

ABSTRACT

Several data can be analyzed about football matches, such as the performance of teams in the championship, history against the opposing team, injured or suspended players. However, when analyzing football in Brazil, other data must be taken into account, such as the distance teams need to travel and the different climates present here. This work carried out the acquisition of data from the Brazilian championship, along with the distances traveled by the teams and the weather on the day of the matches. Based on this information, a data set was built with several performance, distance and weather data about each match played in the Brazilian championship and a classifier that performs the prediction of the final results of the games was developed, which obtained an accuracy of 50%. With this, it was possible to perform an analysis of the real impact of attributes related to weather and distance on the result of the matches, which found that such attributes are not substantially relevant in determining the final score.

Keywords: football; data mining; data analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O esporte com o decorrer dos anos foi se profissionalizando cada vez mais. Equipes e atletas de alto nível passaram a procurar suporte em outras áreas para melhoria de desempenho, como a área de nutrição e aperfeiçoamento físico. Clubes começaram a construir centros de treinamento mais sofisticados, com academias completas, dormitórios e centros médicos.

Nesse cenário de avanço, a computação vem se tornando um fator cada vez mais importante no esporte. Atletas, clubes, instituições esportivas vêm aumentando seus investimentos nessa área em razão do retorno que ela entrega. Hoje, o desempenho esportivo não se resume apenas ao resultado final, diversos outros fatores estão sendo levados em consideração como análises individuais dos atletas com relação a sua condição física para prevenção de lesões, área de atuação no campo, análises de perfomance para eliminar erros de execução e novos equipamentos para recuperação acelerada dos atletas. Em uma pesquisa realizada por Rossi *et al.* (2018), é proposta uma abordagem voltada para prever lesões em jogadores de futebol, utilizando rastreamento com GPS. Dados como distância percorrida nos treinos, distância percorrida acima de determinada velocidade, assim como idade e função do jogador são consideradas para a análise do modelo.

Esse investimento na computação vêm gerando retorno financeiro em diversas áreas esportivas, melhorando o desempenho de equipes e tornando possível que tenham melhores resultados em seus campeonatos. Um exemplo disso, é a equipe de baseball Oakland A's, que com um orçamento limitado para contratações, fez um estudo em cima de centenas de atletas e passou a analisar informações que não eram normalmente avaliadas em contratações de novos jogadores. No artigo publicado por Wassermann *et al.* (2005), mostra-se que as características principais anteriormente analisadas eram velocidade, força no braço, rapidez, habilidade de rebater e resistência mental e que a nova teoria utilizada pelo clube se baseava na capacidade dos jogadores de correr entre as bases e rebater muitas bolas, gerando a partir disso outras estatísticas. Com isso, eles conseguiram formar uma equipe competitiva que foi capaz de bater recorde de vitórias na Liga Americana no ano de 2002.

Nessa área, há uma tendência de tentar prever os resultados das partidas, principalmente no futebol, que possui um mercado milionário e um alcance relevante mundialmente, inclusive já foram desenvolvidos diversos estudos sobre essa temática.

Dentre os trabalhos que tentam realizar essa tarefa, temos o Miljković *et al.* (2010), que se propõe a prever os resultados de partidas da NBA, onde existem apenas duas possibilidades, vitória ou derrota. Nesse estudo, são utilizados dados referentes a estatísticas padrões de uma partida, como quantidade de cestas de três pontos, lances livres, faltas, entre outros. Foram testados vários classificadores, sendo o Naive Bayes o com melhor desempenho, tendo uma acurácia de 67%. Esse resultado foi considerado bastante safistatório tendo em vista que o sistema de predição de jornalistas da CBS obteve 68,3% na temporada anterior.

Em uma outra pesquisa, Buursma (2011), no qual o trabalho tinha intuito de criar um sistema de predição de resultados de partidas do futebol holandês que fosse melhor do que previsões feitas por casas de aposta, foram utilizados dados referente a partidas anteriores, como vitórias recentes, gols feitos e gols sofridos. Os melhores resultados encontrados foram utilizando a Regressão Logística, com uma acurácia de 54,84%. Por fim, ele conclui que utilizando algumas estratégias, é possível conseguir lucro nas casas de aposta.

Porém, faltam pesquisas desse tipo no futebol brasileiro. No Brasil, possuímos alguns outros dados que podem vir a ser relevantes. A dinâmica do futebol brasileiro também é diferente, já que aqui os principais clubes brasileiros costumam disputar muito mais jogos do que equipes de outros países. O Brasil, pelo fato de ser um país de dimensões continentais, faz com que os clubes precisem realizar muitras viagens, assim como implica aos times diversas condições climáticas diferentes. Equipes que jogam boa parte do ano em regiões quentes, precisam se deslocar até o sul do país para disputar partidas em temperaturas próximas a zero.

Nesse sentido, o objetivo desse trabalho foi construir um modelo de regressão utilizando diversos dados de partidas do futebol brasileiro e verificar se os atributos de distância percorridas pelas equipes e condições climáticas tem um impacto significativo na previsão dos resultados finais dos jogos.

Para isso, diversos dados como condições meteorológicas dos locais onde foram realizadas as partidas, dados geográficos a respeito do deslocamento das equipes, foram extraidos de sites e depois combinados em uma base de dados completa para serem treinados com modelos inteligentes. Para verificar se a acurácia do modelo é boa, foi realizada uma comparação com os resultados dos sites de aposta.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Foi realizada uma análise de dados para averiguar qual a relevância/influência de dados referentes as condições climáticas e distâncias percorridas pelas equipes nos resultados finais de partidas de futebol, utilizando técnicas e ferramentas próprias para a execução de todo esse processo de análise.

1.1.2 Objetivos Específicos

- Montar uma base de dados com jogos do futebol brasileiro, dados do tempo no dias das partidas disputadas e de deslocamento das equipes para realização dos jogos, realizando transformações e criação de novas características.
- Coletar dados de casas de aposta para comparativo com modelos desenvolvidos.

- Construir um classificador com a base de dados desenvolvida e averiguar a taxa de acerto do modelo na predição dos resultados finais.
- Analisar o impacto dos atributos de tempo e distância no resultado final dos jogos.

2 REVISÃO DE LITERATURA

Neste capítulo serão apresentados os conceitos fundamentais para o processo de mineração de dados que serão necessários para a solução da proposta. Na seção 2.1 é apresentado o conceito de aquisição de dados. Na seção 2.2 serão abordadas as técnicas utilizadas para realização do pré-processamento dos dados. Por fim, na seção 2.3 serão apresentadas algumas das técnicas de mineração mais utilizadas em análises esportivas.

2.1 Aquisição de dados

A etapa de aquisição é onde serão coletados todos os dados que possam vir a ser úteis no processo de mineração que virá depois. Possuir bons dados é de extrema importância porque será a partir deles que se irá tirar conclusões e caso os dados sejam ruins, mal preparados ou incompletos, toda a informação adquirida e analisada será errada, tendenciosa ou não será condizente com a realidade.

Os dados podem ser adquiridos de várias fontes e de diversas maneiras. A internet é sem dúvidas a maior fonte, porém esses dados estão em diversos formatos diferentes e na maioria das vezes não é fácil fazer essa coleta. Quando queremos extrair determinados dados de um *website* por exemplo, é muito comum a criação de programas conhecidos como *Crawl Agents*, que são feitos exatamente para ir a fundo na página e adquirir os dados lá armazenados. Esse processo é chamado de *Web Crawling*, e é usualmente utilizado quando se trabalha com um grande conjunto de dados.

Porém, os *websites* não são a única fonte de dados, pois podemos extrair informações de uma máquina, de bancos de dados, dados coletados à mão. Então, é necessário ter bem determinado qual o problema ou situação que se está querendo analisar para saber quais informações pesquisar, evitando o armazenamento de contéudo irrelevantes que só ocasionam no desperdício de tempo, espaço e processamento.

Em uma pesquisa realizada por Buursma (2011), os dados coletados foram extraídos de uma base de dados que possuia 15 anos de informações armazenadas a respeito do futebol Holandês. Essa base já possuía dados como todos os eventos do jogo por inteiro assim como de cada tempo separadamente, e muitos outros tipos de informações poderiam ser extraídas dessa base de dados, como gols marcados, vitórias e etc.

Já em pesquisa feita por Zdravevski e Kulakov (2009), para a extração dos dados da NHL (*National Hockey League*), NFL (*National Football League*) dentre outras ligas nacionais americanas, seria necessário o desenvolvimento de um *Crawl Agent*, para obtenção e armazenamento dos dados. Primeiramente, foi verificado de forma manual que o formato dos dados era consistente nas páginas onde se encontravam os resultados finais e suas estatísticas.

Sabendo o formato, qualquer URL de uma data específica que era desejada poderia ser automaticamente produzida. Caso não existissem jogos naquela data, a página do site não

existiria ou iria mostrar um aviso. Além disso, sendo o formato de como os dados são publicados nas páginas também consistente, com tabelas que contém o resumo do jogo e cada feito dos jogadores e também um número constante de colunas em um formato específico. Isso permite que o HTML da página seja analisado e os dados requeridos sejam armazenados na base de dados (ZDRAVEVSKI; KULAKOV, 2009).

Dados incompletos, inconsistentes e ruídos são propriedades comuns em grandes conjuntos de dados e *data warehouses*. Esses dados incompletos podem acontecer por diversos motivos. Atributos que são de interesse podem nem sempre estar a disposição ou dados podem não ser incluídos simplesmente porque não foi considerado importante em determinado momento. Dados relevantes podem não ter sido armazenados por mal entendido ou pelo mal funcionamento de determinado equipamento (HAN; PEI; KAMBER, 2011).

Depois da coleta dos dados requeridos, esses dados precisam ser pré-processados. A preparação dos dados irá gerar um conjunto menor que o original, mas que pode melhorar a eficiência da mineração de forma significativa (ZHANG; ZHANG; YANG, 2003). O pré-processamento de dados é um passo importante no processo de descoberta de conhecimento, porque decisões de qualidade devem ser baseadas em dados de qualidade (HAN; PEI; KAMBER, 2011).

2.2 Pré-processamento dos dados

Existem diversas técnicas para a realização do pré-processamento dos dados e para executar essas técnicas é importante se ter em mente que tipo de dado está sendo tratado, porque existem certos algoritmos que não trabalham com dados númericos como é o caso de regras de associação. Então é importante saber com que tipo de variável se irá trabalhar para que se saiba qual a entrada em um determinado algoritmo de mineração.

Em grandes conjuntos de dados é comum que existam dados faltantes, incompletos, inconsistentes e também redundantes, para esse tipo de falha se utiliza o processo de *Data Cleaning*. Já para a transformação das variáveis para que sejam manipuladas de maneira correta pelos algoritmos de mineração, aplica-se o processo de *Data Transformation* e para a redução da quantidade de dados para que se possa ter um processamento mais rápido mas que não se perca informações importantes, utiliza-se o processo de *Data Reduction*. Todos esses conceitos serão explicados a seguir.

2.2.1 *Data Cleaning*

A rotina de *Data Cleaning* trabalha em realmente limpar os dados seja preenchendo os valores faltantes, diminuindo os ruídos, identificando ou removendo valores atípicos e resolvendo inconsistências. Dados que estão incompletos e inconsistentes podem causar confu-

são no processo de mineração, resultando em resultados não confiáveis (HAN; PEI; KAMBER, 2011).

2.2.1.1 Valores faltantes

Existem algumas maneiras de tratar esses valores que estão faltando no conjunto de dados. Todos os registros que contém dados faltantes podem ser eliminados por inteiro. Porém, essa abordagem pode não ser prática quando na maioria dos registros possuem dados inexistentes (AGGARWAL, 2015).

Os valores faltantes podem ser estimados ou atribuídos. No entando, podem ser criados erros por causa do processo de atribuição que podem vir a afetar os resultados do algoritmo de mineração (AGGARWAL, 2015).

Como apresentado por Han, Pei e Kamber (2011), podemos preencher esses espaços vazios com uma constante global como "Desconhecido". Pode-se também utilizar a média dos atributos, usar o valor médio para todas as amostras que pertencem a mesma classe ou até usar regressão para preencher com o valor mais provável.

2.2.1.2 Valores corrompidos

O dado é dito corrompido quando existe um erro aleatório ou uma variação em uma variável medida. Para solucionar esses dados corrompidos é possível utilizar um método *Binning*, que suaviza um valor de dados classificados realizando uma consulta na sua vizinhança, ou seja, nos valores que estão em volta do dado corrompido (HAN; PEI; KAMBER, 2011).

O valor corrompido pode ser suavizado ajustando o dado a uma função, como a regressão. A regressão linear envolve encontrar a melhor linha que encaixe dois atributos ou variáveis, para que um atributo possa ser usado para prever o outro. Valores atípicos podem ser detectados utilizando o agrupamento de dados, onde valores similares são organizados em grupos. Intuitivamente, valores que ficam fora desses agrupamentos são considerados valores atípicos (HAN; PEI; KAMBER, 2011).

2.2.2 *Data Transformation*

Os processos de *Data Transformation*, são para que eles se adequem às técnicas de mineração específicas e para que possam ajudar o algoritmo a ser melhor executado, fazendo criação de novos atributos ou alterando sua forma de apresentação por exemplo.

2.2.2.1 Construção de atributos

Novos atributos são construídos a partir de atributos já existentes e adicionados com intuito de ajudar a melhorar a acurácia e entendimento da estrutura em dados de alta dimensão. Por exemplo, pode ser desejável adicionar o atributo *área* baseado nos atributos de altura e largura. Com a combinação de atributos, a construção de novos dados pode descobrir informações faltantes sobre as relações entre os dados que podem ser úteis para o descobrimento de conhecimento (HAN; PEI; KAMBER, 2011).

Na pesquisa realizada por Zdravevski e Kulakov (2009), foi possível gerar novos dados como sequência de vitórias, fadiga do time antes de determinado jogo, dados de ataque e defesa do time. Tudo isso analisando dados anteriores à partida atual.

2.2.2.2 Normalização

Em diversos cenários, as diferentes características são representadas em diferentes escalas e portanto não podem ser comparadas umas as outras. Por isso aplicasse a normalização dos dados, que é o processo utilizado para colocar conjuntos de dados que estão em escalas diferentes em um intervalo de valores comum. Por exemplo, um atributo como idade é colocado em uma escala muito diferente de um atributo como salário. Em funções de agregação avaliadas em diferentes características, como a Distância Euclidiana, serão dominadas pelo atributo de maior magnitude (AGGARWAL, 2015).

O processo de normalização pode ser feito de varias maneiras, sendo as mais comuns a normalização *min-max*, normalização *z-score* e normalização por escala decimal.

A normalização *min-max* efetua uma transformação linear no dado original. Supondo que min_A e max_A são os valores mínimos e máximos do atributo A. Segundo Han, Pei e Kamber (2011), essa técnica mapeia um valor *v* do atributo A para um valor *v'* no intervalo [new_min_A, new_max_A] calculando da seguinte forma

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A. \quad (1)$$

Han, Pei e Kamber (2011) definem também a normalização *z-score*, onde os valores do atributo *A* são normalizados baseados na média e no desvio padrão de *A*. A forma como o novo valor *v* do atributo A é normalizado para *v'* é realizado da seguinte maneira

$$v' = \frac{v - \overline{A}}{\sigma}, \quad (2)$$

onde $\overline{A}$ e σ são a média e o desvio padrão, respectivamente, do atributo *A*.

A última forma de normalização definida por Han, Pei e Kamber (2011) é a por escala decimal, que normaliza movendo o ponto decimal dos valores do atributo *A*. O número de ca-

sas decimais movidas vai depender do valor máximo absoluto de *A*. O valor *v* do atributo *A* é normalizado para *v'* utilizando a seguinte fórmula

$$v' = \frac{v}{10^j}, \tag{3}$$

onde *j* é o menor inteiro de tal modo que $Max(|v'|) < 1$.

2.2.3 *Data Reduction*

Técnicas de redução de dados podem ser aplicadas para obter uma representação reduzida do conjunto de dados que é menor em volume mas que ainda mantém a integridade dos dados originais. Isto quer dizer que fazer a mineração no conjunto de dados reduzidos pode vir a ser mais eficiente e produzir os mesmos, ou quase os mesmos, resultados analíticos (HAN; PEI; KAMBER, 2011).

2.2.3.1 Agregação de cubo de dados

Han, Pei e Kamber (2011) definem que cubos de dados armarzenam informações agregadas multidimensionalmente. A Figura 1 apresenta um exemplo de um cubo de dados para análise de dados multidimensionais que se referem às vendas anuais por tipo de item de cada filial de uma empresa. Cada célula do cubo armazena um valor de dado agregado, correspondendo ao ponto de dados no espaço multidimensional.

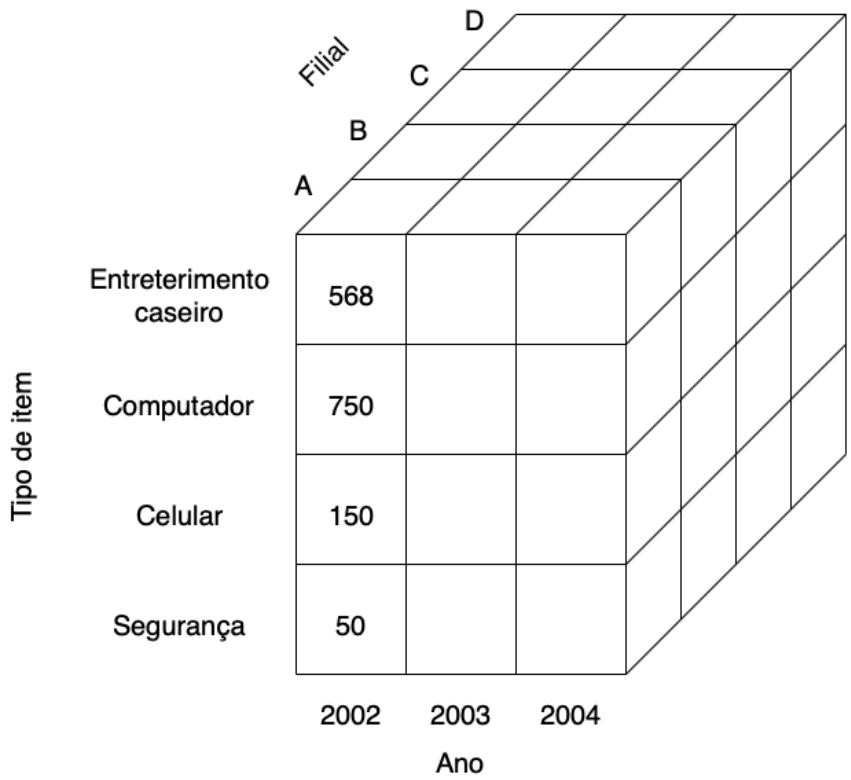
Cubos de dados proveem acesso rápido a dados pré-processados e resumidos, com isso beneficiando processamento analítico online como também a mineração de dados (HAN; PEI; KAMBER, 2011).

2.2.3.2 Seleção de atributos

Esta seção é baseada em (HAN; PEI; KAMBER, 2011).

Conjuntos de dados para análise podem conter centenas de atributos, muitos deles que podem ser irrelevantes para o processo de mineração ou até redundantes. Remover atributos relevantes ou permanecer com atributos irrelavantes pode ser prejudicial, causando confusão no algoritmo de mineração utilizado. Isso pode resultar no descobrimento de padrões de qualidade baixa e o volume de dados irrelevantes ou redundantes pode desacelerar o processo de mineração.

A seleção de um subconjunto de dados reduz o tamanho do conjunto removendo os dados irrelevantes ou redundantes. O objetivo dessa seleção de um subconjunto é achar o menor conjunto de atributos de maneira que a distribuição de probabilidade resultante das classes dos dados seja o mais próxima possível à distribuição original quando foi utilizado todos os atributos.



Fonte: adaptado de Han, Pei e Kamber (2011).

Figura 1 – Cubo de dados referente a vendas

Realizar a mineração em um conjunto de dados reduzido irá reduzir o número de atributos que aparecem nos padrões descobertos, facilitando o entendimento desses padrões.

Métodos heurísticos são comumente utilizados para a seleção de subconjuntos. Esses métodos são geralmente gananciosos, enquanto procuram no espaço dos atributos, eles sempre fazem o que aparenta ser a melhor escolha no momento. A estratégia é fazer uma escolha ótima local na esperança que isso irá guiar para uma solução ótima global.

Os melhores e piores atributos são tipicamente determinados usando testes de significancia estatística, onde se assume que os atributos são independentes dos outros.

Alguns dos métodos heurísticos são a Seleção Progressiva, Seleção Regressiva, uma combinação entre Seleção Progressiva e Regressiva e Indução de Árvores de Decisão. A diferença entre as seleções progressivas e regressivas é que a seleção progressiva irá a cada passo começar a escolher os melhores atributos e preencher um conjunto que estava vazio. Já a seleção regressiva irá sempre removendo os piores atributos do conjunto inicial. A combinação dessas duas seleções fará com que a cada passo o melhor atributo seja selecionado e adicionado em um conjunto reduzido e o pior atributo seja removido do conjunto inicial.

A indução de árvores de decisão constrói uma estrutura semelhante a um fluxograma onde cada nó interno denota um teste em um atributo, cada ramo corresponde a uma saída do teste e cada nó externo denota a previsão da classe. Em cada nó o algoritmo escolhe o melhor atributo para particionar os dados em classes individuais.

A árvore é construída a partir dos dados fornecidos e todos os atributos que não fazem parte da árvore são assumidos como irrelevantes e os que aparecem na árvore fazem parte do conjunto reduzido de dados.

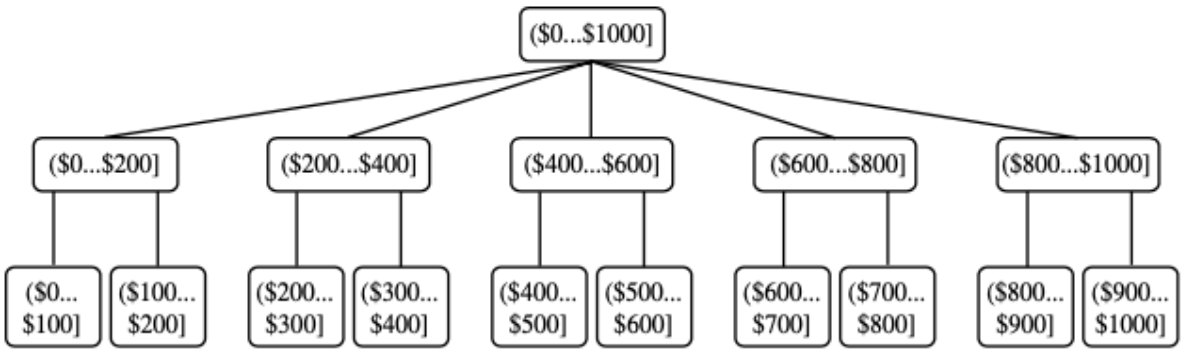
2.2.3.3 Discretização e Conceito de Geração de Hierarquia

As técnicas de discretização de dados podem ser usadas para reduzir o número de valores de um determinado atributo dividindo a extensão dos atributos em intervalos. Os rótulos dos intervalos podem ser então utilizados para substituir os atuais valores dos dados. Substituindo um grande número de atributos contínuos por um pequeno número de intervalos irá reduzir e simplificar os dados originais. Isso irá levar a um nível de representação de conhecimento conciso e de fácil uso dos resultados da mineração (HAN; PEI; KAMBER, 2011).

Essas técnicas podem ser categorizadas baseado em como a discretização é realizada, seja se ela usa informações das classes ou em qual direção ela processa. Se a discretização processa utilizando informação, então ela é dita *discretização supervisionada*, caso contrário ela é *não supervisionada* (HAN; PEI; KAMBER, 2011).

Se o processo começa primeiro encontrando um ou mais pontos para fazer a divisão total do intervalo dos atributos e então repete esse processo recursivamente nos intervalos resultantes, é então chamado de *discretização top-down*. Isso contrasta com a *discretização bottom-up*, onde se começa considerando todos os valores contínuos como potenciais pontos de divisão, remove alguns fazendo uma fusão com valores vizinhos para formar os intervalos e então recursivamente aplica esse processo para encontrar os intervalos resultantes (HAN; PEI; KAMBER, 2011).

A hierarquia de conceito para um dado atributo numérico, define a discretização do atributo. Ela pode ser usada para reduzir o dado ao coletar e substituir um conceito de baixo nível, como um atributo como idade, por um conceito de alto nível, como atributos do tipo jovem, meia idade e idoso (HAN; PEI; KAMBER, 2011). A Figura 2 mostra um exemplo da hierarquia de conceito.



Fonte: Han, Pei e Kamber (2011).

Figura 2 – Hierarquia de conceito para um atributo de preços.

Após a realização da aquisição e do pré-processamento dso dados, a etapa a seguir é a selecionar a técnica de mineração a ser utilizada no processo. A próxima seção apresenta alguns dos principais métodos.

2.3 Técnicas de mineração

2.3.1 Regressão Linear

Esta seção apresenta os fundamentos da Regressão Linear e ela é fundamentada sobretudo na literatura de Han, Pei e Kamber (2011).

A análise por regressão linear envolve uma variável de resposta, *y*, e uma única variável de predição, *x*. Essa é a forma mais simples de regressão que modela *y* como uma função linear de *x*. Isto é

$$y = b + wx, \tag{4}$$

onde a variação de *y* é assumidamente constante, e *b* e *w* são os coeficientes de regressão especificando a interceptação no eixo Y e a inclinação da reta, respectivamente. Os coeficientes de regressão podem também ser analisados como pesos, então de forma equivalente podemos escrever da seguinte maneira,

$$y = w_0 + w_1 x. \tag{5}$$

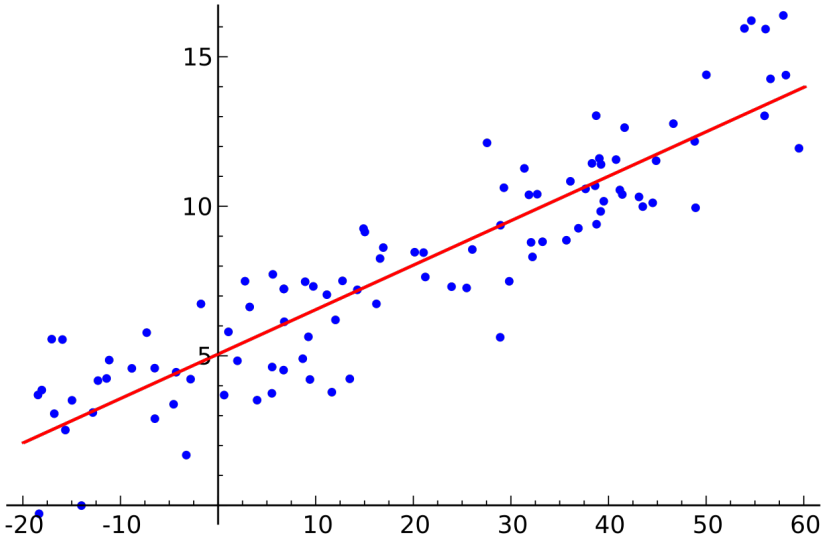
Esses coeficientes podem ser resolvidos pelo método dos mínimos quadrados, onde se estima a linha que melhor minimiza o erro entre os dados reais e os estimados pela linha. Considerando *D* como o conjunto de treinamento consistente de valores para a variável de predição, *x*, e seus valores associados à variável de resposta, *y*, o conjunto de dados *D* contém pares de dados no formato $(x_1,y_1),(x_2,y_2),...,(x_{|D|},y_{|D|})$. Os coeficientes de regressão podem ser estimados usando esse método com as seguintes equações:

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \tag{6}$$

$$w_0 = \bar{y} - w_1 \bar{x}, \tag{7}$$

onde $\bar{x}$ é o valor médio de $x_1,x_2,...,x_{|D|}$, e $\bar{y}$ é a média dos valores $y_1,y_2,...,y_{|D|}$.

O exemplo da Figura 3 mostra a reta que melhor se encaixa em um conjunto de dados e que possui o menor erro entre os dados reais e os previstos.



Fonte: Raschka e Mirjalili (2017).

Figura 3 – Exemplo de Regressão Linear.

Regressão linear múltipla é uma extensão da regressão linear simples, já que envolve mais de uma variável para predição. Ela permite que a váriavel de resposta *y* seja modelada como uma função linear de *n* variáveis ou atributos para predição que descrevem uma tupla $\boldsymbol{X}$.

O conjunto de treinamento, *D*, contém dados no formato $(X_1,y_1),(X_2,y_2),...,(X_{|D|},y_{|D|})$, onde X_i são as tuplas de treinamento com *n* dimensões, com os rótulos das classes associados, y_i. Um exemplo de um modelo de múltipla regressão linear baseada em dois atributos preditores, A_1 e A_2, é o seguinte

$$y = w_0 + w_1 x_1 + w_2 x_2, \tag{8}$$

em que x_1 e x_2 são os valores dos atributos A_1 e A_2, respectivamente em $\boldsymbol{X}$. O método dos mínimos quadrados mostrado anteriormente pode ser extendido para resolver w_0, w_1 e w_2.

2.3.2 Redes Neurais

Redes Neurais são uma forma de simular o sistema nervoso humano. Neurônios biológicos são conectados entre sim em pontos de contato, os quais são referidos como sinapses. Em organismos vivos, o aprendizado é desempenhado pela mudança da força das conexões das sinapses entre os neurônios (AGGARWAL, 2015).

Assim como no caso das redes neurais biológicas, os nós individuais em redes neurais artificiais também são chamadas de neurônios. Esses neurônios são unidades de computação que recebem entradas de outros neurônios, fazem cálculos sobre essas entradas e alimentam outros neurônios. A função para computar em um neurônio é definida pelos pesos nas conexões de entrada desse neurônio. Esse peso pode ser analogamente visto como a força de uma

conexão sináptica. Ao se alterar esses pesos de forma apropriada, a função de computação pode ser aprendida (AGGARWAL, 2015).

O estímulo externo em redes neurais artificiais para aprender esses pesos é fornecido pelos dados de treinamento. A ideia é incrementalmente modificar os pesos sempre que predições incorretas são feitas pelos conjunto atual de pesos (AGGARWAL, 2015).

A chave para a eficácia das redes neurais é a arquitetura usada para organizar as conexões entre os nós. Existem vários tipos de arquiteturas, começando com um simples *perceptron* de uma camada até complexas redes com multicamadas (AGGARWAL, 2015).

2.3.2.1 *Perceptron*

Esta seção apresenta os conceitos do *Perceptron* e ela é baseada sobretudo nas literaturas de Aggarwal (2015) e Raschka e Mirjalili (2017).

O *perceptron* contém duas camadas de nós, que correspondem aos nós de entrada e a um único nó de saída. O número de nós de entrada é exatamente igual a dimensionalidade *d* dos dados subjacentes. Cada nó de entrada recebe e transmite um único atributo numérico para o nó de saida. Portanto, os nós de entrada apenas transmitem valores de entrada e não realizam nenhum cálculo nesses valores. No modelo básico do *perceptron* o nó de saída é o único que realiza funções matemáticas nas entradas.

Podemos dizer que a ideia por trás de neurônios artificiais é uma classificação binária onde nos referimos às duas classes como 1 (classe positiva) e -1 (classe negativa). Define-se então uma função de decisão (ϕ(z)) que faz uma combinação linear de certos valores de entrada $\boldsymbol{x}$ e um vetor de pesos correspondente $\boldsymbol{w}$, onde *z* é a entrada da rede e é definida da seguinte maneira:

$$z = w_1 x_1 + ... + w_m x_m \tag{9}$$

Se a entrada da rede de uma certa amostra $x^{(i)}$ for maior do que um limite θ, a predição será da classe 1, caso contrário a classe será -1. No algoritmo do *perceptron*, a função de decisão $\phi()$ é uma variante da função degrau:

$$\phi(z) = \begin{cases} 1 & \text{se } z \geq \theta \\ -1 & \text{caso contrário} \end{cases} \tag{10}$$

Para simplicidade, definimos o peso zero como $w_0 = -\theta$ e $x_0 = 1$. Na literatura de aprendizado de máquina, o limite negativo, ou peso zero, é usualmente chamado de unidade *bias*.

As regras inicias do *perceptron* são simples e podem ser resumidas da seguinte maneira:

1. Inicializar os pesos com 0 ou pequenos valores aleatórios.

2. Para cada amostra de $x^{(i)}$:

    a) Calcular o valor de saída $\hat{y}$.

    b) Atualizar os pesos.

Com relação ao valor de saída e os pesos, se define que $\hat{y}$ é a classe prevista pela função degrau e que a atualização simultânea de cada peso w_j no vetor de pesos $\boldsymbol{w}$ pode ser escrita da seguinte forma:

$$w_j := w_j + \Delta w_j, \tag{11}$$

onde Δw_j pode ser calculado pela regra de aprendizado do *perceptron*:

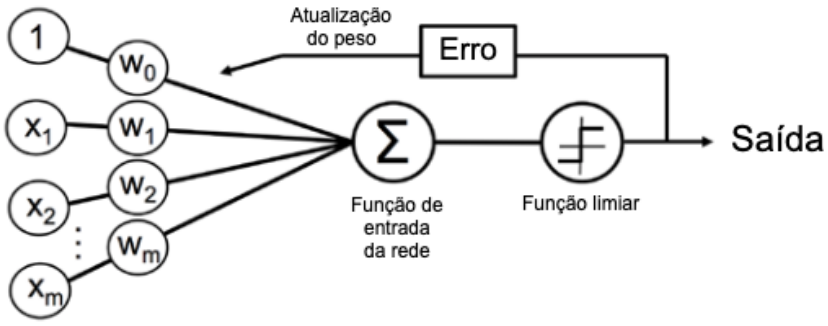
$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \tag{12}$$

A variável η é a taxa de aprendizagem (tipicamente um valor constante entre 0.0 e 1.0), $y^{(i)}$ é a verdadeira classe da i-ésima amostra de treinamento, e $\hat{y}$ é a classe prevista. É de importância notar que todos os pesos no vetor de pesos estão sendo atualizados simultaneamente, o que significa que não se recalcula $\hat{y}$ antes que todos os pesos sejam atualizados.

Os pesos são atualizados sempre com base no valor do erro entre a classe verdadeira, $y^{(i)}$, e a classe prevista, $\hat{y}$, também chamada de $output^{(i)}$. Os pesos permanecem inalterados caso a classe prevista esteja correta, caso contrário, os pesos serão alterados na direção da classe alvo de forma positiva ou negativa.

O objetivo do algoritmo de rede neural é aprender o vetor de pesos, até que as previsões, se aproximem o máximo possível das classes verdadeiras que foram utilizadas para treinamento.

A Figura 4 exemplifica a estrutura do *perceptron*.



Fonte: adaptado de Raschka e Mirjalili (2017).
Figura 4 – Rede neural de uma camada: *Perceptron*

O modelo *perceptron* é a forma mais básica de redes neurais, contendo apenas uma camada de entrada e uma de saída. Pelo fato da camada de entrada apenas transmitir os valores dos atributos sem realmente realizar cálculos nessas entradas, a função aprendida por

esse modelo é um simples modelo linear baseado em um único nó de saída. Na prática, modelos mais complexos podem ser necessários de serem aprendidos com redes neurais multicamadas.

2.3.2.2 Redes Neurais Multicamadas

Esta seção apresenta os fundamentos de Redes Neurais Multicamadas (HAN; PEI; KAMBER, 2011).

Uma rede neural *feed-forward* multicamada consiste em uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída. Cada camada é feita de unidades. As entradas da rede correspondem aos atributos medidos para cada tupla de treinamento e essas entradas são simultaneamente inseridas nas unidades, fazendo assim a camada de entrada. Os dados de entrada passam pela camada de entrada e são então ponderados e passados simultaneamente para a segunda camada de unidades, a camada escondida (*hidden layer*). As saídas da camada escondida podem ser passadas para outra camada escondida, porém, em prática, é usualmente utilizada apenas uma. As saídas ponderadas da última camada escondida são então inseridas em unidades que fazem a camada de saída, que emite então a previsão da rede para as tuplas dadas.

A rede é *feed-forward*, porque nenhum dos pesos retorna para uma unidade de entrada ou para a unidade de saída de uma camada anterior. Essa rede é completamente conectada, pois cada unidade provê entradas para as unidades da camada seguinte. Cada unidade de saída recebe como entrada, uma soma ponderada das saídas da camada anterior. Redes *feed-forward* multicamadas conseguem modelar a classe de previsão como uma combinação não-linear das entradas.

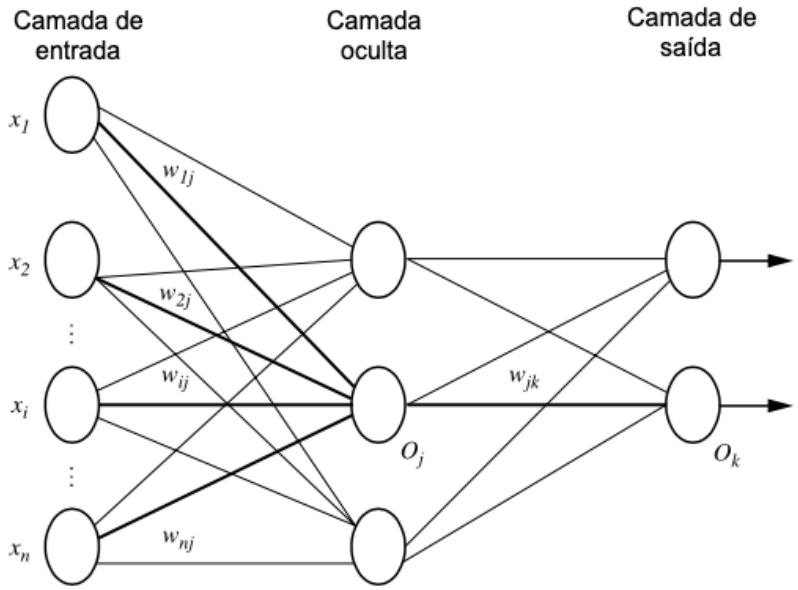
A Figura 5 demonstra o esquema de uma rede neural com camadas de entrada, camada oculta e camada de saída.

Como afirma Aggarwal (2015), quando existe um erro de classificação, ou seja, a previsão foi errada, é necessário que as camadas mais adiante deem um *feedback* às camadas anteriores, e isso é feito com o uso do algoritmo de *Backpropagation*.

O *backpropagation* aprende interativamente processando o conjunto de dados das tuplas de treinamento, comparando a previsão da rede para cada tupla com o valor real do alvo. Para cada tupla de treinamento, os pesos são modificados para que se minimize o erro quadrado médio entre a previsão da rede e o valor real. Essas modificações são feitas para trás, ou seja, da camada de saída, passando por cada uma das camadas ocultas até a primeira delas.

Uma das primeiras coisas a se fazer é a inicialização dos pesos na rede. Eles são inicializados com pequenos números aleatórios. Cada unidade possui um *bias* associado, que também são inicializados com pequenos números aleatórios. Cada tupla de treinamento, $\boldsymbol{X}$, é processada com os passos seguintes.

O primeiro passo é fazer a propagação dos dados para as camadas mais a frente. Primeiramente, a tupla de treinamento é inserida na camada de entrada da rede, onde ela irá



Fonte: adaptado de Han, Pei e Kamber (2011).
Figura 5 – Rede Neural *feed-forward* multicamada.

passar sem alterações. Depois, a entrada da rede e a saída de cada unidade das camadas ocultas e de saída são calculadas. A entrada da rede para uma unidade na camada oculta ou de saída é calculada como uma combinação linear das entradas. Cada uma dessas unidades possuem várias entradas que são, de fato, as saídas das unidades conectadas a ela na camada anterior.

Cada conexão possui um peso. Para calcular a entrada da rede na unidade, cada entrada conectada à essa unidade é multiplicada pelo seu peso correspondente, e então somada. Dado uma unidade *j* em uma camada oculta ou de saída, a entrada da rede, I_j, na unidade j é

$$I_j = \sum_i w_{ij} O_j + \theta_j, \tag{13}$$

onde w_{ij} é o peso da conexão da unidade *i* da camada anterior com a unidade *j*; O_j é a saída da unidade *i* da camada anterior; e θ_j é o valor *bias* da unidade. O *bias* atua como um limite, pois serve para variar a atividade da unidade.

Cada unidade da camada oculta ou de saída, pega a entrada da rede e aplica a função de ativação à ela. As funções que podem ser utilizadas são as funções logística ou sigmóide e elas significam a ativação do neurônio representado por aquela unidade. Dada a entrada da rede I_j para a unidade *j*, então O_j, a saída para a unidade *j*, é calculada como

$$O_j = \frac{1}{1 + e^{-I_j}} \tag{14}$$

Essa função ela mapeia um grande domínio de entrada em um pequeno intervalo entre 0 e 1. A função logística é não-linear e diferenciável o que faz com que o algoritmo de *backpropagation* consiga modelar problemas de classificação que não são linearmente separáveis. Esse valor

de O_j é calculado para cada uma das camadas ocultas e de saídas, que é onde será gerada a previsão da rede.

Após a propagação para frente será feita a propagação dos erros para as camadas anteriores. O erro é propagado para trás, atualizando os pesos e os *biases* para refletir o erro na previsão da rede. Para uma unidade *j* na camada de saída, o erro Err_j é calculado como

$$Err_j = O_j(1 - O_j)(T_j - O_j), \tag{15}$$

onde O_j é a atual saída da unidade *j*, e T_j é o valor alvo conhecido para uma dada tupla de treinamento.

Para calcular o erro de uma unidade *j* de uma camada oculta, a soma ponderada dos erros das unidades conectadas à unidade *j* na camada seguinte, são considerados. O erro de uma unidade *j* de uma camada oculta é

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}, \tag{16}$$

onde w_{jk} é o peso da conexão entre a unidade *j* e a unidade *k* da camada superior seguinte, e Err_k é o erro da unidade *k*.

Os pesos são atualizados seguindo as seguintes equações, onde Δw_{ij} é a mudança no peso w_{ij}:

$$\Delta w_{ij} = (l) Err_j O_i \tag{17}$$

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{18}$$

A variável *l* é a taxa de aprendizagem, uma constante que tipicamente possui valores entre 0.0 e 1.0. *Backpropagation* aprende utilizando um método de gradiente descendente para procurar pelo conjuntos de pesos que se encaixe nos dados de treinamento de tal forma que se minimize a distância média quadrada entre a classe prevista pela rede e o valor alvo conhecido. A taxa de aprendizagem ajuda a evitar que se fique preso em mínimos locais em espaços de decisão e encoraja a encontrar o mínimo global. Se a taxa de aprendizagem for muito pequena, o aprendizado irá acontecer em um ritmo muito lento, mas caso a taxa seja muito grande, oscilações entre soluções inadequadas podem ocorrer.

O valores dos *biases* são atualizados com as seguintes equações, onde $\Delta\theta_j$ é a mudança do *bias* θ_j:

$$\Delta\theta_j = (l) Err_j \tag{19}$$

$$\theta_j = \theta_j + \Delta\theta_j \tag{20}$$

Nota-se que a atualização dos pesos e do *bias* são feitas após a apresentação de cada tupla e esse método é conhecido como atualização de caso. De forma alternativa, os incrementos dos pesos e dos *biases* podem ser acomulados em uma variável, para que eles sejam atualizados depois que todas as tuplas do conjunto de treinamento sejam apresentadas. Essa estratégia é chamada de atualização de época, onde uma iteração pelo conjunto de treinamento é uma época.

O treinamento termina quando todos os Δw_{ij} das épocas anteriores sejam tão pequenos que estão abaixo de um limite específico, ou quando a porcentagem das tuplas classificadas de maneira incorreta na época anterior seja abaixo de um limite ou quando um número de épocas especificado for ultrapassado.

2.3.3 Regressão Logística

Esta seção apresenta os fundamentos de Regressão Logística e ela é baseada na literatura de Raschka e Mirjalili (2017).

Regressão logística é um modelo de classificação que é facilmente implementável mas que tem uma boa performance com classes linearmente separavéis. De forma similar ao *perceptron*, esse modelo é também um modelo linear para classificação binária que pode ser extendido para classificação de múltiplas classes.

O conceito de *odds ratio* é imporatnte de ser inserido para que se melhor entenda a regressão logística como um modelo probabilístico. Esse conceito diz respeito às probabilidades a favor de um evento em particular. O *odds ratio* pode ser escrito como $\frac{p}{(1-p)}$ onde *p* significa a probabilidade de um evento positivo, que não necessariamente significa bom, mas se refere ao evento que quer ser previsto. Pode-se então definir melhor a função *logit*, que é simplesmente o logaritmo do *odds ratio* (*log-odds)*:

$$logit_{(p)} = log\frac{p}{(1-p)} \tag{21}$$

A função *logit* pega um valor de entrada no intervalo de 0 à 1 e o transforma em valores em todo o intervalo de números reais, no qual pode-se usar para expressar a relação linear entre valores de atributos e o *log-odd*:

$$logit(p(y=1|x)) = w_0x_0 + w_1x_1 + ... + w_mx_m = \sum_{i=0}^{m} w_ix_i = w^tx, \tag{22}$$

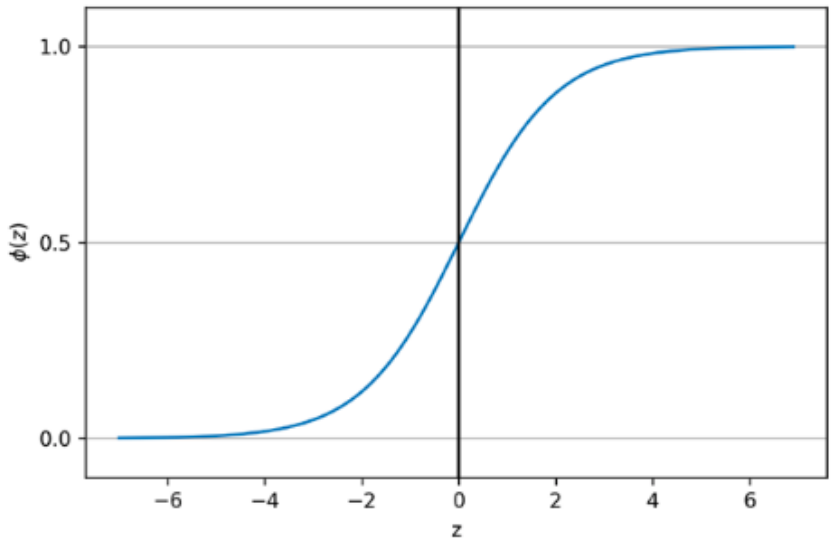
onde, $p(y=1|x)$ é a probabilidade condicional de que uma amostra particular pertence à classe 1 dado os atributos *x*.

O real interesse é prever a probabilidade que uma certa amostra pertence a uma classe específica, o que é a forma inversa da função *logit*. Também chamada de função sigmóide logística:

$$\phi(z) = \frac{1}{1 + e^{-z}}, \tag{23}$$

onde *z* é a entrada da rede, a combinação linear dos pesos com os atributos da amostra, $z = w^t x = w_0 x_0 + w_1 x_1 + ... + w_m x_m$.

Na Figura 6 pode-se ver que $\phi(z)$ se aproxima de 1 caso *z* vá para o infinito $z \to \infty$ já que e^{-z} se torna um valor muito pequeno para grandes valores de *z*. De forma similar, $\phi(z)$ se aproxima de 0 para $z \to -\infty$ como resultado de um denominador cada vez maior. Dessa maneira, pode-se concluir que a função sigmóide pega valores numéricos reais como entrada e os transforma em valores no intervalo [0,1] com uma interceptação em $\phi(z) = 0.5$.



Fonte: Raschka e Mirjalili (2017).
Figura 6 – Exemplo da função sigmóide logística.

Em regressão logística, a função de ativação simplesmente se torna a função sigmóide onde depois se interpreta a sua saída como a probabilidade de uma determinada amostra pertencer a classe 1, $\phi(z) = (P(y = 1|x; w)$, dado os atributos $\boldsymbol{x}$ parametrizados pelos pesos $\boldsymbol{w}$. A probabilidade da previsão pode ser simplesmente convertida em um resultado binário através de uma função limite:

$$\hat{y} = \begin{cases} 1 & \text{se } \phi(z) \geq 0.5 \\ 0 & \text{caso contrário} \end{cases} \tag{24}$$

Existem várias aplicações onde não se tem interesse nos rótulos de classe previstos, mas em que a estimativa da probabilidade de associação já é particulamente útil.

É importante ressaltar como se realiza o ajuste de alguns parâmetros desse modelo, como os pesos $\boldsymbol{w}$. Para isso, se utiliza a função de custo como sendo o erro quadrático, definida da seguinte maneira:

$$J(w) = \sum_i \frac{1}{2}(\phi(z^{(i)}) - y^i)^2 \tag{25}$$

Minimiza-se essa função para aprender os pesos $\boldsymbol{w}$. Para a explicação de como se derivar a função de custo por regressão logística, é necessário definir a probabilidade L que se quer maximizar quando se constrói o modelo de regressão logística, assumindo que as amostras individuais no conjunto de dados são independentes das outras. A fórmula é a seguinte:

$$L(w) = P(y|x;w) = \prod_{i=1}^{n} P(y^i|x^i;w) = \prod_{i=1}^{n} (\phi(z^i))^{y^i}(1-\phi(z^i))^{1-y^i} \tag{26}$$

Na prática, é mais fácil maximizar o log dessa equação, que é chamada de função log-verossimilhança:

$$l(w) = logL(w) = \sum_{i=i}^{n} [y^{(i)} log(\phi(z^{(i)})) + (1-y^{(i)}) log(1-\phi(z^{(i)}))] \tag{27}$$

Primeiramente aplica-se a função log que irá reduzir o potencial de subfluxos numéricos que podem vir a ocorrer caso as probabilidades sejam muito pequenas, e em seguida converte-se o produto dos fatores em uma soma de fatores, tornando assim mais fácil o processo de derivar essa função.

Usa-se um algoritmo de otimização como o gradiente ascendente para maximizar a função log-verossimilhança. De forma alternativa, pode-se reescrever a log-verossimilhança como uma função de custo *J* que pode ser minimizada pelo uso do gradiente descendente.

$$J(w) = \sum_{i=i}^{n} [-y^{(i)} log(\phi(z^{(i)})) - (1-y^{(i)}) log(1-\phi(z^{(i)}))] \tag{28}$$

Para facilitar a visualização e o entendimento da função de custo, utilizamos o cálculo do custo para uma instância de treinamento de apenas uma amostra:

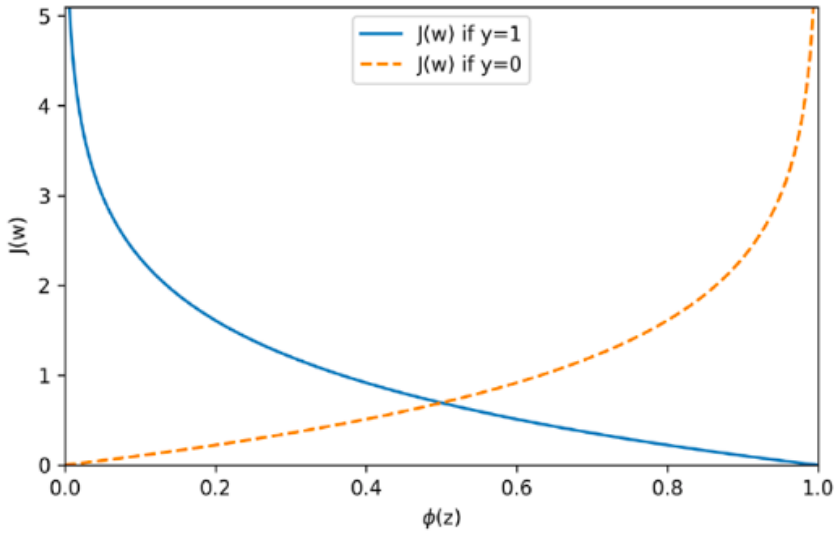
$$J(\phi(z),y;w) = -y log(\phi(z)) - (1-y) log(1-\phi(z)) \tag{29}$$

Analisando a equação, pode-se ver que o primeiro termo se torna zero se $y = 0$, e o seguno termo se torna zero caso $y = 1$:

$$J(\phi(z),y;w) = \begin{cases} -log(\phi(z)) & \text{se } y = 1 \\ -log(1-\phi(z)) & \text{se } y = 0 \end{cases} \tag{30}$$

Na Figura 7 pode-se notar que o custo se aproxima de 0 (linha contínua) quando a previsão de que a amostra pertence à classe 1 é feita corretamente e que o custo também se aproxima de 0 se a previsão para $y = 0$ (linha tracejada) for feita de maneira correta. Entretanto,

se a previsão for errada, o custo tende ao infinito, ou seja, o ponto principal é que as previsões que são feitas erradas são penalizadas com um enorme crescimento do custo.



Fonte: Raschka e Mirjalili (2017).

Figura 7 – Função de custo.

A regressão logística é um dos métodos que utilizam probabilidade para realizar classificação, porém, existem outras técnicas como o Naive Bayes, que é baseado no Teorema de probabilidade do Bayes.

2.3.4 Naive Bayes

Esta seção é baseada em (HAN; PEI; KAMBER, 2011).

Classificadores Bayesianos podem prever a probabilidade de associação a uma classe, como a probabilidade que uma certa tupla pertença a uma classe particular. Esses classificadores são baseados no teorema de Bayes que será descrito a seguir.

Seja *X* a tupla de dados. Em termos Baysianos, *X* é considerado a evidência, que usualmente é descrita por medições em um conjunto de *n* atributos. *H* é a hipótese de que certa tupla *X* pertence a uma classe específica e nós problemas de classificação, queremos determinar $P(H|X)$, que é a probabilidade que a hipótese *H* mantém dada a evidência *X*. Essa probabilidade é chamada de probabilidade posterior (probabilidade *a posteriori*) de *H* condicionada em *X*.

De forma contrastante a probabilidade posterior, existem as probabilidades anteriores (probabilidade *a prior*), que são a $P(H)$ que é a probabilidade anterior de *H* independentemente de *X*, $P(X)$ que é a probabilidade anterior de *X* independentemente de *H* e $P(X|H)$ que é a probabilidade de *X* condicionada à *H*.

Para melhor entendimento, vamos supor que temos dados de consumidores separados pelos atributos *idade* e *renda*, e que a tupla de dados *X* é um consumidor de 35 anos com renda

de $\$40{,}000$. Sendo a hipótese *H* que o consumidor irá comprar um computador, a probabilidade posterior $P(H|X)$ reflete a probabilidade de que um consumidor *X* irá comprar um computador dados os atributos informados.

As probabilidades $P(H), P(X)$ e $P(X|H)$ são estimadas a partir do conjunto de dados, onde $P(H)$ é a probabilidade de qualquer consumidor comprar um computador sem levar em conta nenhum atributo, $P(X)$ é a probabilidade de um consumidor se encaixar nos atributos de ter 35 anos e renda de $\$40{,}000$ e $P(X|H)$ é a probabilidade do consumidor ter 35 anos e possuir renda de $\$40{,}000$ sabendo que ele irá comprar um computador.

O teorema de Bayes é útil no fato de que ele prove uma maneira de calcular a probabilidade posterior, $P(H|X)$, a partir das probabilidades $P(H), P(X)$ e $P(X|H)$, utilizando a seguinte fórmula:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}. \tag{31}$$

Estudos de comparação de algoritmos de classificação encontraram um classificador Bayesiano simples, o *Naive Bayes*, que pode ser comparado em performance com árvores de decisão e redes neurais selecionadas. Esse classificador também exibiu uma alta acurácia e velocidade quando aplicado em grandes conjuntos de dados.

O classificador *Naive Bayes* utiliza o teorema de Bayes e assume que o efeito do valor de um atributo em uma classe dada é independente aos valores dos outros atributos. Essa suposição é chamada de independência condicional de classe e é feita para simplificar os cálculos envolvidos.

Primeiramente assumimos que *D* é o conjunto de tuplas com seus rótulos de classe associados. Usualmente, cada tupla é representada por um vetor de atributos de dimensão *n*, $X = (x_1,x_2,...,x_n$, representando *n* medidas feitas na tupla a partir de *n* atributos $(A_1,A_2,...,A_n)$.

Supondo que existem *m* classes, $C_a,C_2,...,C_m$. Dada uma tupla *X*, o classificador irá prever que *X* pertence à classe que obtiver a maior probabilidade posterior, condicionada à *X*. O classificador Bayesiano preve que a tupla *X* pertence à classe C_i, se e somente se

$$P(C_i|X) > P(C_j|X) \text{ para } 1 \leq j \leq m, j \neq i. \tag{32}$$

Portanto, se maximiza $P(C_i|X)$. Como $P(X)$ é constante para todas as classes, apenas $P(X|C_i)P(C_i)$ precisa ser maximizada. Caso as probabilidades anteriores das classes não sejam conhecidas, é comum que se assuma que as classes sejam igualmente prováveis, ou seja, $P(C_1) = P(C_2) = ... = P(C_m)$, e só seria necessário maximizar $P(X|C_i)$. As probabilidades anteriores das classes podem ser estimadas por $P(C_I) = |C_{i,D}|/|D|$, onde $C_{i,D}|$ é o número de tuplas de treinamento da classe C_i no conjunto de tuplas *D*.

Em um conjunto de dados com muitos atributos, seria extremamente caro no ponto de vista computacional calcular $P(X/C_i)$, então para reduzir esses cálculos utilizasse a suposi-

ção de independência condicional de classes, que assume que não existem relacionamentos dependentes entre os atributos. Portanto,

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i) \quad (33)$$

As probabilidades $P(x_1|C_i), P(x_2|C_i)$,...,$P(x_n|C_i)$ podem ser facilmente estimadas a partir das tuplas de treinamento.

Portanto, para prever o rótulo da classe para $\boldsymbol{X}$, $P(\boldsymbol{X}|C_i)P(C_i)$ é avaliado para cada classe C_i, então a previsão do classificador para o rótulo da classe da tupla $\boldsymbol{X}$ será C_i, se e somente se,

$$P(\boldsymbol{X}|C_i)P(C_i) > P(\boldsymbol{X}|C_j)P(C_j) \text{ para } 1 \leq j \leq m, j \neq i. \quad (34)$$

Ou seja, a previsão da classe será C_i quando $P(\boldsymbol{X}|C_i)P(C_i)$ for o valor máximo.

2.4 Trabalhos Relacionados

Em uma pesquisa realizada por Buursma (2011), que tinha como objetivo desenvolver um sistema para predição de resultados do futebol holandês, o conjunto inicial de atributos era composto pelos gols marcados pelo time mandante nas últimas partidas, gols marcados pela equipe visitante nos jogos anteriores, gols sofridos pelo time mandante e pelo time visitante, quantidade de pontos ganhos pelos dois times. Esses dados foram testados com a ferramenta Weka, que usa aprendizado de máquina para calcular a correlação entre certos tipos de dados.

Para a divisão dos dados em base de teste e base de treinamento, a ferramenta utilizada oferece a validação cruzada como solução, onde a base de dados é dividida em dez partes iguais e se usam nove como base de treinamento e a parte restante como base de teste. Esse processo é repetido dez vezes, sendo cada vez selecionado uma parte diferente para ser a base de testes. Os dados foram importados na ferramenta Weka e vários algoritmos de aprendizado de máquina foram executados e classificaram os dados inseridos, informando como saída a porcentagem de predições feitas corretamente.

Como dito anteriormente, os atributos possuem características como os gols marcados nas últimas X partidas como mandante e esse valor de X se referia ao número de jogos analisados. Buursma fez testes com o X variando entre 4 e 75 jogos e notou que depois de um certo número de jogos a mudança no resultado final começava a ser muito pequena, então ele decidiu por usar o número de 20 partidas e sempre que alguma das equipes envolvidas não tivesse completado 20 jogos, a partida em si seria descartada e não entraria nas bases de teste e treinamento.

Após definir o número de partidas que seriam utilizadas como base a serem analisadas e feita toda a divisão de base de treinamento e base de testes, foi feita a análise com seis

algoritmos de aprendizado, obtendo os melhores resultados sempre com *ClassificationViaRegression*, que utiliza regressão linear e *MultiClassClassifier* que utiliza uma regressão logística, com acurácias de 54,86% e 54,84%, respectivamente.

Em uma outra pesquisa, Miljković *et al.* (2010), propuseram um software que tinha como finalidade predizer resultados de partidas da NBA e calcular os *spread points* para jogos de basquete. A predição dos resultados foi definida como um problema de classificação, onde os resultados das partidas são separados em duas classes, uma onde o mandante vence e a outra onde é o visitante que sai vitorioso.

Para cada time foi separado dois grupos de atributos. O primeiro grupo consiste em estatísticas padrões do basquete, como cestas de três pontos, lances livres, rebotes, faltas, entre outros. O segundo grupo é composto por informações sobre o campeonato em geral, como número de vitórias e derrotas, vitórias como mandante e como visitante e sequência de vitórias.

Várias técnicas de classificação foram testadas sendo o classificador Naive Bayes o que obteve melhor desempenho. Para a melhoria desse classificador foi utilizado também a seleção e a normalização das características. A ferramenta *RapidMiner* foi a utilizada para o desenvolvimento do classificador.

O conjunto de dados era composto por 778 partidas da temporada 2009/2010, retiradas do site oficial da NBA e de um site dedicado a NBA League. Nessa pesquisa também foi utilizada a validação cruzada e a acurácia foi calculada pela divisão entre número de predições corretas pelo número total de predições. O resultado obtido foi de uma acurácia de 67%, o que foi considerado bastante satisfatório quando comparado com outros sistemas de predição como o de jornalistas da CBS que obtiveram resultado de 68.3% na temporada anterior à analisada nessa pesquisa.

2.4.1 Sumário

Neste capítulo foi apresentada toda a fundamentação teórica que é utilizada no capítulo seguinte, onde será realizado o processo de mineração desde a coleta dos dados até a apresentação dos resultados obtidos.

3 METODOLOGIA

3.1 Aquisição de dados

A base inicial é um arquivo adquirido de uma publicação feita no Medium, FARAH, E (2021), com todos os jogos da primeira divisão do Campeonato Brasileiro entre os anos de 2003 e 2018. Nesse arquivo temos as datas da realização das partidas, os dois times envolvidos, os gols marcados e a posição de cada equipe na tabela de classificação após o resultados destes jogos. Para complementar essa base, foram adquiridos dados referentes as condições climáticas nos dias dos jogos, informações essas retiradas do site do Instituto Nacional de Meteorologia.

Os dados geográficos foram adquiridos a partir da latitude e longitude dos estádios onde cada clube mandava suas partidas. Essa posição foi adquirida a partir da API do *Wikipedia* que permite coletar dados das páginas existentes no site. Contendo essas duas informações, é possível então realizar o cálculo de distância ou deslocamento das equipes para cada partida. Com a criação de novos atributos a partir dos dados já existentes, a base final possui 5681 registros e 89 atributos. A Tabela 1 mostra a base inicial montanda após a aquisição dos dados descritos anteriormente.

ATRIBUTOS	VALORES
ANO	2003
DATA	03-05-2003
POSIÇÃO MANDANTE	6
MANDANTE	FLAMENGO
GOLS MANDANTE	2
VISITANTE	VITÓRIA
GOLS VISITANTE	1
POSIÇÃO VISITANTE	11
DESLOCAMENTO VISITANTE	1221.49
TEMP MÁXIMA MANDANTE	25.5
TEMP MÍNIMA MANDANTE	19.1
UMIDADE RELATIVA MANDANTE	76.5
VELOCIDADE VENTO MANDANTE	0.4
TEMP MÁXIMA VISITANTE	31.1
TEMP MÍNIMA VISITANTE	24.1
UMIDADE RELATIVA VISITANTE	85.75
VELOCIDADE VENTO VISITANTE	1.3

Tabela 1 – Dados disponíveis como base inicial.

Também foram adquiridos dados referentes a sites de apostas, que serão utilizadas para validar o modelo final, comparando a acurácia do modelo e dos sites. Os dados que são informados são os dois times que se enfrentaram, o placar final do jogo e a probabilidade de vitória do mandante, empate ou vitória do visitante, como mostra a Tabela 2.

MANDANTE	GOLS VISITANTE	VISITANTE	GOLS VISITANTE	ODDS VITÓRIA MANDANTE	ODDS EMPATE	ODDS VITÓRIA VISITANTE
Athletico-PR	1	Botafogo	1	1.35	4.25	6.35
Flamengo	6	Cruzeiro	2	1.33	4.3	6.65

Tabela 2 – Exempo de odds do site de aposta

3.1.1 Pré-processamento

3.1.1.1 Limpeza dos dados

Foram removidos todos os jogos de times no qual as cidades não possuem aeroportos ou caso não existam cidades próximas com as mesmas características para que os dados possam ser levados em consideração. Também foram removidas as primeiras cinco partidas de cada time, porque os novos atributos gerados vão utilizar como base sempre os últimos cinco jogos de cada equipe, para gerar atributos históricos.

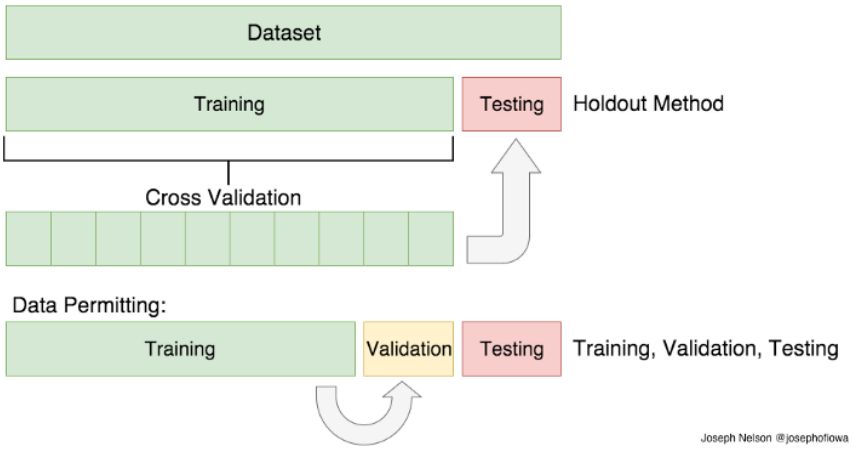
3.1.1.2 Engenharia de atributos

A partir dos atributos iniciais, foram criados diversas outras características e foram usados sempre os últimos 5 jogos de cada equipe como base. Os dados de gols de mandantes e visitantes geraram informações como total de gols feitos nas últimas partidas, gols sofridos, gols feitos como mandante, gols feitos como visitante, dentre outros. Com os gols feitos também foi possível gerar informações de vitórias, empates e derrotas de cada time. Com as informações de latitute e longitude foram criados informações sobre o deslocamento das equipes visitantes para cada partida e também um histórico com as distâncias acumuladas do mandante e visitante nas suas últimas partidas.

3.1.2 Seleção do modelo

A seleção do modelo foi realizada em etapas. A primeira foi a divisão da base em 3 partes: treinamento, validação e teste, conforme mostra a Figura 8. Depois foi utilizado o *GridSearch* com validação cruzada do tipo *k-fold time series*, onde foram testados diversos valores de hiperparâmetros e analisadas as performances dos modelos. Após isso, foi feito o treinamento do melhor modelo já com os hiperparâmetros definidos com a base de treinamento. Por último foi realizada a validação do modelo com a base de teste. A Figura 9 mostra todo o processo de seleção do modelo.

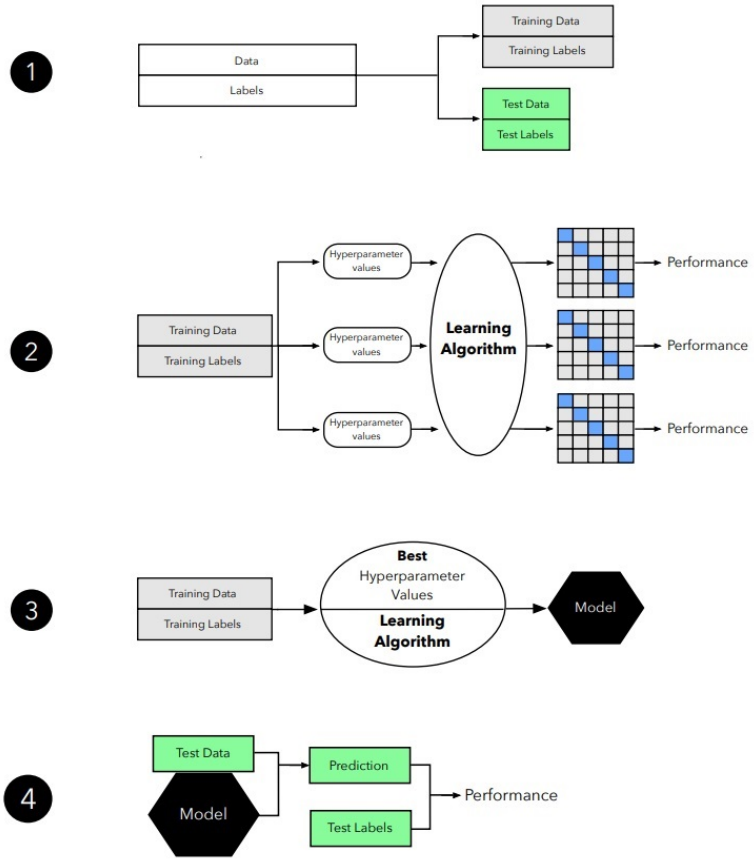
O conjunto de validação será sempre um ano inteiro do campeonato, enquanto treino e teste serão os anos anteriores ao ano selecionado para ser validação, utilizando sempre no mínimo três anos de campeonato e no máximo dez, por exemplo, se usarmos o ano de 2018 para validação, faremos o treinamento e teste do modelo com os anos de 2008 até 2017.



Fonte: BRONSHTEIN, A. (2017).

Figura 8 – Divisão da base em treino, teste e validação.

Na validação cruzada serão inseridas variações dos hiperparâmetros para encontrar os que desempenham melhor junto ao modelo. Dessa forma iremos em conjunto escolher o modelo e seus melhores atributos.



Fonte: Sebastian Raschka.

Figura 9 – Processo de seleção do modelo.

3.1.3 Avaliação do modelo

Para avaliar o melhor modelo será refeito o treinamento com os melhores parâmetros encontrados e utilizaremos a acurácia do modelo em comparação com a acurácia de casas de apostas. A acurácia das casas de aposta será calculada partindo das *odds* de vitória do mandante, empate ou vitória do visitante conforme exemplificado na Tabela 2. As *odds* são a probabilidade de tal evento acontecer, então com essa informação será possível verificar a quantidade de acertos pelos sites de apostas por ano e assim tirar a informação de acurácia. Além disso, para verificar o impacto que os atributos de clima e distância tem no modelo final iremos fazer dois experimentos, sendo o primeiro deles variar o hiperparâmetro C para verificar a importância desses atributos e o segundo é retreinar o modelo sem esses atributos de clima e distância para analisar o seu desempenho.

4 RESULTADOS FINAIS E DISCUSSÕES

4.1 Pré-processamento

Foram removidos todos os jogos dos times Santos e Joinville pelo fato de não possuírem aeroportos em suas cidades e nem cidades próximas que tivessem condições climáticas parecidas que pudessem ser utilizadas. Tínhamos no total 43 equipes e passamos a ter 41. Além dos jogos dessas duas equipes, foram removidas as primeiras 5 rodadas do campeonato em todos os anos, pelo fato de utilizarmos sempre os 5 jogos anteriores das equipes para geração de atributos históricos. Com isso, a base final possui 5681 amostras.

4.2 Engenharia de atributos

Assim como explicado na subseção 3.1.1.2, a partir dos atributos de gols feitos pelo mandante e pelo visitante, foram construidos atributos de quantidade de gols feitos, gols sofridos, vitórias, empates e derrotas nos últimos jogos. Também foram criados atributos de desempenho como média de gols feitos e sofridos e desempenho dos times como mandante e visitante. Já com os dados de latitude e longitude, foram criados atributos de deslocamento do time visitante para aquela partida específica e atributos históricos como a distância total percorrida pelas equipes nos últimos jogos. Com os atributos de clima foi possível criar informações da variação de temperatura entre as cidades do mandante e do visitante. Inicialmente eram apenas 17 atributos, com a criação dessas novas características passamos a ter 86 atributos como mostra a Tabela 4, onde os atributos com sufixo man representam os dados referentes ao time mandante e o sufixo vis os dados do time visitante.

4.3 Seleção do modelo

Como já mencionado na subseção 3.1.2, foi realizado o *GridSearch* com validação cruzada na base e o melhor modelo encontrado foi a regressão logística com os seguintes hiperparâmetros:

- fit intercept: True
- max iter: 100
- penalty: l1
- solver: liblinear
- C: 0.048329

ANO	ACURÁCIA			INTERVALO DE TREINAMENTO
	ODDS	MODELO 1	MODELO 2	
2004	50,70	-	-	-
2005	49,22	-	-	-
2006	50,13	49,09	49,09	(2003-2005)
2007	53,31	51,98	51,06	(2003-2006)
2008	55,78	53,94	55,15	(2003-2007)
2009	52,89	51,47	52,12	(2003-2008)
2010	48,15	46,97	46,36	(2003-2009)
2011	47,37	48,79	48,79	(2003-2010)
2012	49,21	48,94	49,24	(2003-2011)
2013	49,73	47,88	48,48	(2003-2012)
2014	52,36	53,68	54,29	(2004-2013)
2015	54,21	53,03	52,73	(2005-2014)
2016	53,56	55,35	54,09	(2006-2015)
2017	45,26	41,82	41,82	(2007-2016)
2018	56,31	52,29	52,67	(2008-2017)
Média	**51,41**	**50,40**	**50,45**	-

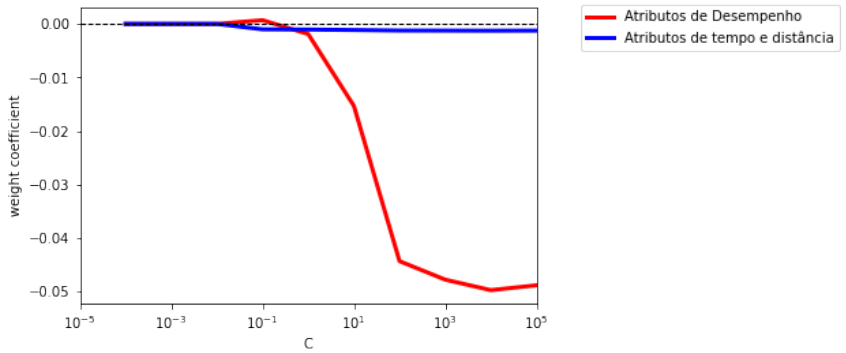
Tabela 3 – Resultados finais.

4.4 Avaliação do modelo

Na Tabela 3 estão dispostos todos os resultados finais. Na segunda coluna, chamada de *Odds*, podemos ver a acurácia dos sites de aposta em cada ano, que foi construída conforme explicado na subseção 3.1.3. Ao compararmos a segunda coluna com a terceira coluna (Modelo 1), que se refere aos resultados finais do melhor modelo, podemos perceber que os resultados são relativamente próximos. Em uma análise comparativa, notam-se pequenas variações, a exemplo do ano de 2006 em que a acurácia das casas de aposta foi 1,04% maior, enquanto no ano de 2016 o meu modelo obteve melhor resultado.

Já a terceira coluna, Modelo 2, apresenta a acurácia do modelo sem os atributos de tempo e distância. Nota-se que tais dados não alteram substancialmente os resultados finais, tendo em vista que os resultados permaneceram sempre muito próximos. Também foi possível notar que o impacto desses atributos não foi significavo ao realizarmos a alteração do hiperparâmetro C, ou seja, quando alteramos a penalização que o modelo faz às caracterísicas.

Na Figura 10, onde temos a linha vermelha representando os atributos de desempenho (sem negrito, conforme a Tabela 4) e a linha azul representando os atributos de tempo e distância (em negrito, conforme a Tabela 4), podemos notar que quando a penalização é alta, com valor de C menor que 0.1, as características ficam zeradas ou próximas de zero, mas que quando a penalização é suavizada os atributos de desempenho se tornam mais importantes, enquanto os atributos de tempo e distância permanecem muito próximos de zero.



Fonte: Autoria própria..

Figura 10 – Variação do hiperparâmetro C.

Ano	Vitorias_Como_Mandante_Man	Derrotas_UltimosJogos_Vis
Data	Vitorias_Como_Visitante_Man	Vitorias_Como_Mandante_UltimosJogos_Man
Posicao_Man	Empates_Como_Mandante_Man	Vitorias_Como_Visitante_UltimosJogos_Man
Mandante	Empates_Como_Visitante_Man	Empates_Como_Mandante_UltimosJogos_Man
Gols_Man	Derrotas_Como_Mandante_Man	Empates_Como_Visitante_UltimosJogos_Man
Visitante	Derrotas_Como_Visitante_Man	Derrotas_Como_Mandante_UltimosJogos_Man
Gols_Vis	Vitorias_Como_Mandante_Vis	Derrotas_Como_Visitante_UltimosJogos_Man
Posicao_Vis	Vitorias_Como_Visitante_Vis	Vitorias_Como_Mandante_UltimosJogos_Vis
Pontos_Ganhos_Man	Empates_Como_Mandante_Vis	Vitorias_Como_Visitante_UltimosJogos_Vis
Pontos_Ganhos_Vis	Empates_Como_Visitante_Vis	Empates_Como_Mandante_UltimosJogos_Vis
Diferenca_dias	Derrotas_Como_Mandante_Vis	Empates_Como_Visitante_UltimosJogos_Vis
GolsMarcados_Man	Derrotas_Como_Visitante_Vis	Derrotas_Como_Mandante_UltimosJogos_Vis
GolsSofridos_Man	GolsMarcados_UltimosJogos_Man	Derrotas_Como_Visitante_UltimosJogos_Vis
GolsMarcados_Vis	GolsSofridos_UltimosJogos_Man	**Quantidade_Dias_entre_jogos_man**
GolsSofridos_Vis	SaldoGols_UltimosJogos_Man	**Quantidade_Dias_entre_jogos_vis**
SaldoGols_Man	SaldoGols_UltimosJogos_Vis	**Deslocamento_vis**
SaldoGols_Vis	GolsMarcados_UltimosJogos_Vis	**Deslocamento_UltimosJogos_man**
Media_Gols_Man	GolsSofridos_UltimosJogos_Vis	**Deslocamento_UltimosJogos_vis**
Media_Gols_Como_Mandante_Man	Media_Gols_UltimosJogos_Man	**Temperatura_Maxima_Man**
Media_Gols_Como_Visitante_Man	Media_Gols_Como_Mandante_UltimosJogos_Man	**Temperatura_Minima_Man**
Media_Gols_Vis	Media_Gols_Como_Visitante_UltimosJogos_Man	**Umidade_Relativa_Man**
Media_Gols_Como_Mandante_Vis	Media_Gols_UltimosJogos_Vis	**Velocidade_Vento_Man**
Media_Gols_Como_Visitante_Vis	Media_Gols_Como_Mandante_UltimosJogos_Vis	**Temperatura_Maxima_Vis**
Vitorias_Man	Media_Gols_Como_Visitante_UltimosJogos_Vis	**Temperatura_Minima_Vis**
Empates_Man	Vitorias_UltimosJogos_Man	**Umidade_Relativa_Vis**
Derrotas_Man	Empates_UltimosJogos_Man	**Velocidade_Vento_Vis**
Vitorias_Vis	Derrotas_UltimosJogos_Man	**Variacao_Temperatura_Maxima**
Empates_Vis	Vitorias_UltimosJogos_Vis	**Variacao_Temperatura_Minima**
Derrotas_Vis	Empates_UltimosJogos_Vis	

Tabela 4 – Todos atributos da tabela final.

Dito isso, o resultado do melhor modelo possui uma acurácia média de 50,40%, um bom resultado, tendo em vista que um resultado randômico teria cerca de 33% de acurácia, considerando os três resultados possíveis, vitória, empate e derrota.

5 CONCLUSÃO

Esse trabalho tinha a intenção de construir um classificador para prever resultados de partidas de futebol com foco no impacto que atributos de tempo e distância, relativos as equipes, podiam ter. Para isso, foi construída uma base de dados com diversas informações sobre cada partida realizada. A base possui informações climáticas referente ao jogo realizado, como temperatura máxima e mínima registradas no mesmo dia, possui também informações de deslocamento da equipe visitante para realização da partida e um histórico de deslocamento nos últimos jogos dos dois times. Informações sobre desempenho recente de ambos os times também exisitam, por exemplo a quantidade de vitórias como mandante nos últimos jogos, ou gols marcados fora de casa.

Para fazer a validação do modelo criado, foram coletados dados de casas de aposta e verificada a acurácia que esses sites possuíam para cada ano. Com isso, foi possível comparar com o nosso modelo se o desempenho era próximo do que se espera. Um classificador utilizando a Regressão Logística foi desenvolvido a partir dessa base de dados criada. Foram realizados diversos testes manipulando o classificador para obter o melhor resultado possível.

Com o classificador montado e os dados de casas de aposta coletados para comparação, pudemos analisar que o desempenho do nosso modelo foi aceitável, tendo uma taxa de acerto muito próxima e em alguns casos até superior. Porém, foi possível notar que os atributos de tempo e distância não tiveram um impacto significativo e que o modelo criado sem esses dados ficou muito próximo do modelo inicial.

REFERÊNCIAS

AGGARWAL, C. C. **Data mining: the textbook**. [*S.l.*]: Springer, 2015.

BRONSHTEIN, A. **Train/Test Split and Cross Validation in Python**. 2017. Disponível em: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6. Acesso em: 20 de abril de 2022.

BUURSMA, D. Predicting sports events from past results towards effective betting on football matches. In: **Conference Paper, presented at 14th Twente Student Conference on IT, Twente, Holland**. [*S.l.*: *s.n.*], 2011. v. 21.

FARAH, E. **Um tabelão de pontos corridos**. 2021. Disponível em: https://medium.com/@p14nt40/um-tabel~ao-de-pontos-corridos-2d3e8643ae98. Acesso em: 30 de dezembro de 2021.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. [*S.l.*]: Elsevier, 2011.

MILJKOVIĆ, D. *et al.* The use of data mining for basketball matches outcomes prediction. In: IEEE. **IEEE 8th International Symposium on Intelligent Systems and Informatics**. [*S.l.*], 2010. p. 309–312.

RASCHKA, S.; MIRJALILI, V. **Python machine learning**. [*S.l.*]: Packt Publishing Ltd, 2017.

ROSSI, A. *et al.* Effective injury forecasting in soccer with gps training data and machine learning. **PloS one**, Public Library of Science San Francisco, CA USA, v. 13, n. 7, p. e0201264, 2018.

WASSERMANN, E. *et al.* An examination of the moneyball theory: a baseball statistical analysis. **The Sport Journal**, United States Sports Academy, v. 8, n. 1, 2005.

ZDRAVEVSKI, E.; KULAKOV, A. System for prediction of the winner in a sports game. In: SPRINGER. **International Conference on ICT Innovations**. [*S.l.*], 2009. p. 55–63.

ZHANG, S.; ZHANG, C.; YANG, Q. Data preparation for data mining. **Applied artificial intelligence**, Taylor & Francis, v. 17, n. 5-6, p. 375–381, 2003.