Trabalho de Conclusão de Curso

Comparação de técnicas para aumentar a eficácia e eficiência de Análise de Sentimento por meio da redução do número de *features*

Gabriel Fabrício B. Freire
gfbf@ic.ufal.br

Orientadores:

Bruno Almeida Pimentel
Rafael de Amorim Silva

Maceió, Setembro de 2019

Gabriel Fabrício B. Freire

Comparação de técnicas para aumentar a eficácia e eficiência de Análise de Sentimento por meio da redução do número de *features*

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientadores:

Bruno Almeida Pimentel

Rafael de Amorim Silva

Maceió, Setembro de 2019

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

F866c Freire, Gabriel Fabrício B.

Comparação de técnicas para aumentar a eficácia e eficiência de análise de sentimento por meio da redução do número de *features* / Gabriel Fabrício B Freire. – 2019.

37 f. : il.

Orientador: Bruno Almeida Pimentel e Rafael de Amorim Silva.

Monografia (Trabalho de conclusão de curso em Ciência da Computação) - Universidade Federal de Alagoas, Instituto de Computação. Maceió.

Bibliografia: f. 32-37.

1. Aprendizado do computador. 2. Modelo de sentimento . 3. Processamento de linguagem natural (Computação). I. Título.

CDU: 004.81:159.942.52

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Prof. Dr. Bruno Almeida Pimentel - Orientador
Instituto de Computação
Universidade Federal de Alagoas

Prof. Dr. Rafael de Amorim Silva - Coorientador
Instituto de Computação
Universidade Federal de Alagoas

Prof. MSc Lucas Benevides Viana de Amorim - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Prof^{a}. Dr^{a}. Roberta Vilhena Vieira Lopes - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Maceió, Setembro de 2019

Agradecimentos

Inicialmente gostaria de agradecer ao professor Bruno Pimentel, meu orientador, pela atenciosidade e paciência ao me ajudar a idealizar e compor partes deste trabalho. Ao professor Rafael Amorim, meu coorientador, por sempre ser bastante prestativo, incentivador e companheiro.

Gostaria também de agradecer aos demais professores que me deram oportunidades e me serviram de inspiração durante o curso. Em especial aos professores Ibsen Bittencourt e Leandro Dias, orientadores de projetos de pesquisa que participei, por terem acreditado no meu potencial.

Além disso, agradeço também à minha família e meus amigos por todo o apoio e incentivo que sempre me deram ao longo do curso, e principalmente nessa etapa final que se fez bem longa.

Resumo

Análise de sentimentos é uma das mais prevalentes áreas do Aprendizado de Máquina, por conseguir converter textos em classes. Porém essa conversão vem com alguns inconvenientes, dentre eles estão o alto tempo de treinamento e a baixa acurácia em geral, os quais são agravados por conta dos modelos de definição de *features* mais comuns, que geram *features* baseadas em cada uma das palavras contidas no conjunto de dados. Assim, esse trabalho tem por objetivo comparar meios de reduzir o número de *features* considerando tanto o aumento de eficiência, na forma de tempo de treinamento e predição, quanto o aumento de eficácia, na forma de melhores parâmetros e resultados para as predições.

Palavras-chave: Aprendizado de Máquina, Análise de Sentimento, Processamento de Linguagem Natural, Redução de Dimensionalidade

Abstract

Sentiment Analysis is one of the most prevalent areas of Machine Learning, for being able to convert texts into classes. Though that conversion comes with its own drawbacks, among them are the high training cost and low general accuracy, which are compounded by the most common feature definition models, that generate features based on every word contained in the training datasets. Thus, this research has the goal of comparing means of reducing the number of features to considering both the increase efficiency, in the form of shorter training and prediction times, and the increase in efficacy, in the form of better parameters and results for predictions.

Key-words: Machine Learning, Sentiment Analysis, Natural Language Processing, Dimensionality Reduction

Lista de Figuras

Conteúdo

1

Introdução

A análise de sentimentos tem por objetivo definir técnicas automatizáveis para se extrair informações subjetivas de textos e classificá-los adequadamente de acordo com o sentimento que o autor passa, com objetivo de transformar conjuntos de textos em dados mais simplesmente acopláveis a sistemas de predição, recomendação ou tomada de decisão tradicionais [6].

Este capítulo conterá uma introdução do presente trabalho; primeiramente mostrando sua motivação, em seguida serão mostrados o objetivo do trabalho e, por fim, os avanços que este pode gerar no cenário de análise de sentimentos.

1.1 Motivação

Com os avanços da tecnologia, a *internet* tem ficado cada vez mais populada por opiniões e análises de conteúdos dos mais diversos tipos. Desde comentários de vídeos no *Youtube* a análises de restaurantes no *Yelp* a opiniões sobre ações que podem afetar o mercado financeiro. Contudo, grande parte dessas informações estão em forma de texto, o que torna necessária a aplicação de análise de sentimentos para que se possa melhor inferir sobre o que está sendo avaliado, seja este um vídeo, restaurante ou qualquer outro tipo de conteúdo ou serviço. Dessa forma, temos que é fundamental para diversos tipos de análise social e de mercado o uso de tal análise.

Dentre as principais aplicações da Análise de Sentimentos estão atendimento ao consumidor, pesquisas de mercado, monitoramento de reputação de empresas e de satisfação de clientes [39].

Parte da etapa de pré-processamento para a análise de sentimentos envolve processamento de linguagem natural, como pode ser visto na Figura 1.1 nas etapas de *Tokenization* a *Stemming*, a qual resulta na geração de uma matriz de caraterísticas extraídas dos textos disponíveis, onde cada característica (ou *feature*) é relacionada a um termo, que pode ser composto de uma

Figura 1.1: Etapas da Análise de Sentimentos [57]

palavra (unigrama) ou um conjunto de palavras (bigrama, trigrama, etc.) que ocorrem em um ou mais textos. É fácil ver que isso tende a resultar em um alto número de *features* que não necessariamente explicam bem a variável alvo, ou seja, o sentimento do texto.

1.2 Objetivo

Neste trabalho, iremos comparar meios de reduzir a complexidade computacional de se aplicar aprendizado de máquina aos resultados do processamento de linguagem natural adicionando passos intermediários para redução de *features* geradas pelo processo.

1.3 Estrutura do documento

Este documento foi organizado em seis capítulos, sendo o primeiro esta introdução. Essa seção tem por objetivo descrever brevemente os próximos cinco capítulos.

O capítulo 2 é relacionado à etapa de pré-processamento da análise de sentimentos, incluindo a descrição dos processamentos de linguagem natural necessários, como *Stemming* e Remoção de *Stopwords*, e dos tipos de redução de dimensionalidade utilizados.

O capítulo 3 apresenta breve descrições dos quatro algoritmos de predição de dados utilizados na comparação feita nesse trabalho.

Já no capítulo 4 é apresentada a metodologia desse trabalho, onde são melhor descritos as ferramentas utilizadas e suas limitações, as métricas utilizadas, a redução feita sobre os dados e o problema utilizado para comparação dos métodos de redução de *features*.

No capítulo 5 são apresentados resultados e discussões, dentre eles gráficos e tabelas relacionados às execuções dos algoritmos de aprendizado sobre os dados já reduzidos, tanto em dimensionalidade quanto em numerosidade.

E, por fim, no capítulo 6 será mostrada a conclusão desse trabalho, resumindo o conteúdo e as principais observações e limitações desse trabalho, além de apresentar possíveis trabalhos futuros.

2

Análise de Sentimentos

O presente capítulo apresentará uma breve revisão de literatura em relação à etapa de pré-processamento da análise de sentimento, incluindo processamento de linguagem natural e redução de dimensionalidade.

2.1 Processamento de linguagem natural

Como já citado na introdução, o processamento de linguagem natural faz parte da etapa de pré-processamento da análise de sentimento. Esse processamento será usado para transformar uma coleção de documentos em uma coleção de dados sobre cada documento, de forma a ter-se um meio de categorizar tais documentos com algoritmos de aprendizado de máquina.

2.1.1 *Stemming*

Como pode ser visto na Figura 2.1, se trata do processo de extrair a raiz de uma palavra e substituir esta por sua raiz, isso é feito pois palavras com mesma raiz - como "*waited*", "*waiting*"e "*waits*"que possuem a raiz "*wait*"em comum - indicam um mesmo sentimento, logo não devem ser diferenciadas no processo de análise [20].

2.1.2 Remoção de *stopwords*

É o processo de remoção das palavras que não serão úteis à análise, exemplos comuns de *stopwords* podem ser vistos na Figura 2.2. Em geral são palavras que ocorrem muitas vezes em textos, porém não trazem um significado útil à maioria das análises feitas sobre textos [46, 58].

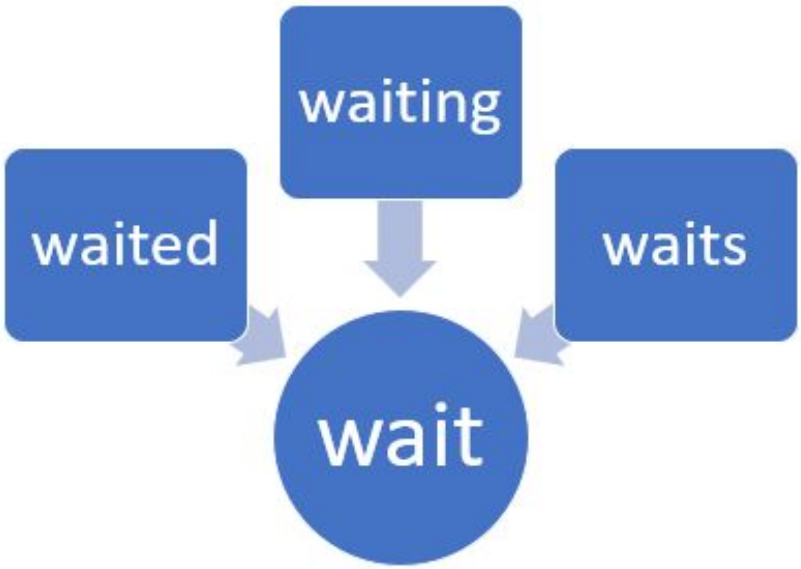

Figura 2.1: Exemplo de *Stemming* [47]

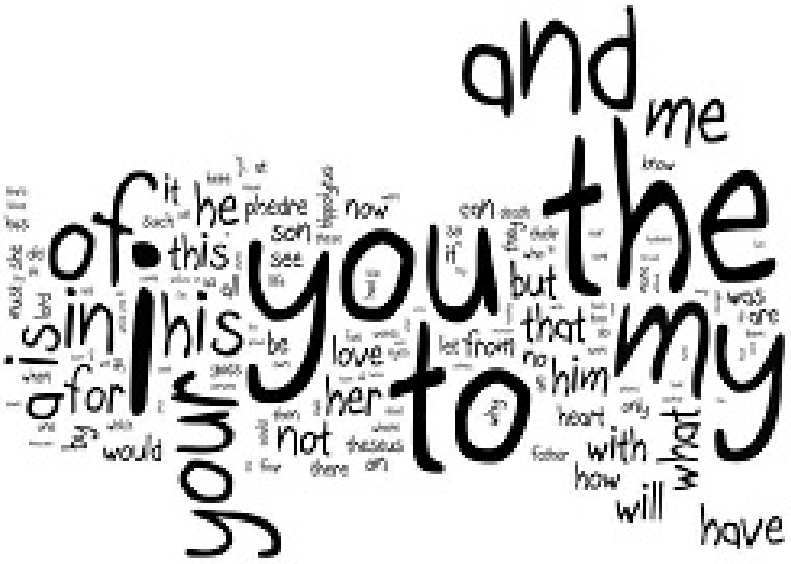

Figura 2.2: Exemplo de *Stopwords* [3]

2.1.3 Definição das *features*

Para os propósitos deste estudo, cada um dos *reviews* será visto como um documento, e dentro dos documentos utilizados encontraremos vários termos, os quais podem ser unigramas (contém apenas uma palavra) ou bigramas (contém duas palavras encontradas em sequência) definidos pelas palavras existentes nos documentos e sua disposição no texto.

Esses termos encontrados nos documentos serão utilizados como *features* na análise de sentimento. Para dar valor às *features* a partir de um texto, existem diversas abordagens, dentre elas as abordagens mais comuns são o *bag-of-words*, na qual é contabilizado o número de vezes que cada termo ocorre em cada documento, e o *TF-IDF* (Term Frequency–Inverse Document Frequency) [6], que utiliza a seguinte fórmula para gerar os valores das *features* para cada documento:

$$w_{ij} = tf_{ij} \cdot \log(\frac{N}{df_i}) \tag{2.1}$$

Onde w_{ij} é o peso, ou seja, o valor da *feature* do termo i no documento j, tf_{ij} é o número de ocorrências do termo i no documento j, N é o número de documentos e df_i é o número de ocorrências do termo i no conjunto de documentos como um todo.

Note que ambas as abordagens citadas definem uma *feature* para cada termo (unigrama, bigrama, etc) da coleção de documentos como um todo - pois todos os documentos do conjunto precisam ter as mesmas *features* para a análise - isso faz com que os dados gerados sejam altamente esparsos e o número total de *features* muito alto na a maioria dos casos.

2.2 Redução de dimensionalidade

Além das técnicas citadas anteriormente, que são específicas ao processamento de linguagem natural, existem também técnicas mais gerais para a redução de dimensionalidade. Dentre essas técnicas, existem abordagens baseadas em testes de hipótese, como o uso dos resultados do chi-quadrado para determinar *features* menos relacionadas [25]. Existem também abordagens mais complexas, como análise discriminante [23] ou análise de componentes principais [1], que são métodos que fazem análise das variáveis do modelo e realizam transformações sobre ele com objetivo de reduzir drasticamente o número de *features*.

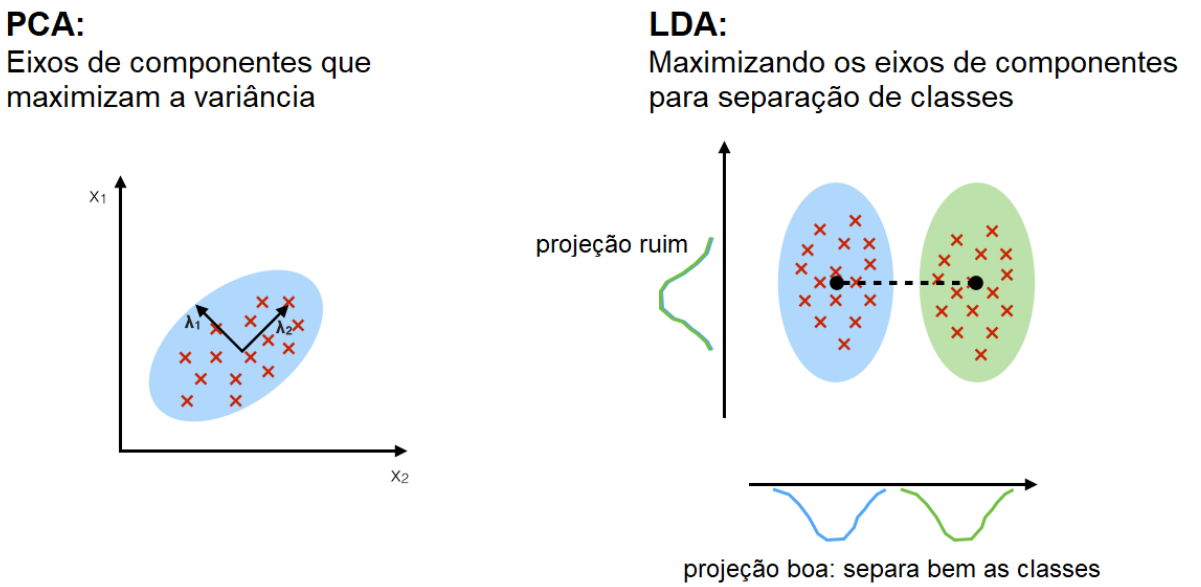

Figura 2.3: Comparação entre Principal Component Analysis e Linear Discriminant Analysis [44]

2.2.1 Redução baseada em correlação

Essa técnica é a mais simples dentre as escolhidas, pois leva em conta apenas o coeficiente de correlação de Pearson entre as *features* e a variável alvo com base no conjunto de dados. Quanto à eliminação das *features*, a ideia é estabelecer um valor de corte, como a média ou mediana das correlações, e remover as *features* com correlação inferior a esse valor.

2.2.2 Redução baseada em chi-quadrado

O teste de hipótese chi-quadrado utiliza a seguinte fórmula para cada combinação de classe e *feature*:

$$\chi^2 = \sum \frac{(f_o - f_e)^2}{f_e} \tag{2.2}$$

Onde f_o é a frequência observada de uma determinada classe no conjunto de dados (número de vezes que a classe aparece no conjunto), f_e é a frequência esperada da classe caso não haja relacionamento entre a *feature* e a variável alvo.

A redução envolve o uso dos resultados desse teste de hipótese para cada *feature* e a remoção de *features* com p-valor abaixo de 0.05, visto que essas não tem relacionamento claro com a variável alvo de acordo com o teste, uma vez que o p-valor indica a probabilidade de que a *feature* em questão explica os valores da variável alvo por coincidência.

2.2.3 Redução utilizando análise de componentes principais

Esse método para redução de dimensionalidade, conhecido como PCA, funciona de uma maneira um pouco diferente dos anteriores, pois ele não simplesmente remove termos, mas gera componentes, os quais substituem as *features* do conjunto de dados e em geral são menos numerosos.

O PCA tem por objetivo encontrar o subespaço vetorial com base contendo os vetores que representem as direções de variância máxima no espaço vetorial original [33]. Após isso, as *features* antigas são transformadas para esse subespaço vetorial, e o resultado são os componentes principais.

Para a geração dos componentes, é utilizada uma matriz W, que serve para converter pontos em um espaço t-dimensional para um espaço f-dimensional, e tem suas colunas compostas por autovalores e_i, os quais são obtidos a partir da seguinte fórmula:

$$\lambda_i e_i = Q e_i \tag{2.3}$$

Onde $Q = XX^T$ é a matriz de covariância entre as variáveis do modelo e λ_i é o autovalor relacionado ao autovetor e_i. Dessa forma, após obter a matriz W, para obtenção dos novos vetores de *features* $y_i \in \Re^f$, é utilizada a seguinte equação, onde x_i são os vetores de *features* anteriores:

$$y_i = W^T x_i \tag{2.4}$$

2.2.4 Redução utilizando análise discriminante linear

Esse método, também conhecido como LDA, é bastante similar ao PCA, uma vez que ele gera novas *features* com o objetivo de substituir as do conjunto de dados. Porém a principal diferença entre os dois métodos é que este método busca o subespaço que melhor discrimina dentre as classes presentes no conjunto de dados, e não necessariamente o que melhor descreve os dados, como é feito pelo PCA [33]. Essa diferença entre os dois métodos é exemplificada na Figura 2.3.

Ou seja, o LDA busca combinar linearmente as *features* do conjunto de dados de forma a obter a maior diferenciação inter-classe e a menor intra-classe. Para isso, são usadas duas fórmulas, as quais são aplicadas em todos os elementos de todas as classes para gerar duas matrizes de dispersão.

A primeira é relacionada à diferenciação intra-classe, e é conhecida como a matriz *within-class*, definida pela seguinte equação:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (x_i^j - \mu_j)(x_i^j - \mu_j)^T \tag{2.5}$$

Onde c é o número de classes, N_j é o número de elementos da classe j, x_i^j é o i-ésimo elemento da classe j e μ_j é a média da classe j.

Já a segunda matriz está relacionada com a diferenciação inter-classe, e é conhecida como *between-class*, dada pela seguinte equação:

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu)(\mu_j - \mu)^T \tag{2.6}$$

Onde c é o número de classes, μ_j é a média da classe j e μ é a média de todas as classes.

Dessa forma, o objetivo é maximizar a diferenciação inter-classe e minimizar a diferenciação intra-classe, o que pode ser descrito pela fórmula:

$$\frac{det|S_b|}{det|S_w|} \tag{2.7}$$

3

Predição de Dados

O presente capítulo apresentará uma breve descrição sobre aprendizado de máquina e de cada um dos algoritmos de aprendizado analisados neste trabalho. Dentre eles estão os algoritmos *Random Forest*, *Decision Tree*, KNN (*K-Nearest Neighbors*) e a Análise Discriminante Linear (*Linear Discriminant Analysis*).

3.1 Aprendizado de Máquina

Essa é uma das partes mais prominentes do campo de inteligência artificial, por ter como objetivo fazer com que programas possam tirar conclusões e perceber padrões com uma capacidade de acerto, em alguns casos, maior do que especialistas humanos. Isso faz com que a falta ou o alto custo de especialistas de diversas áreas não sejam mais um impedimento para muitas empresas.

O Aprendizado de Máquina pode ser dividida em dois principais tipos de aprendizado, supervisionado e não-supervisionado, os quais são exemplificados na Figura 3.1, e serão descritos nas subseções a seguir.

3.1.1 Aprendizado Não-Supervisionado

No aprendizado não-supervisionado não é esperado que, no conjunto de dados de treinamento, as respostas para o problema tenham sido pré-calculadas. Isso torna os algoritmos desse tipo de aprendizado ferramentas muito poderosas, pois podem ser aplicados a problemas sem respostas claras ou aos quais não há um conhecimento prévio dos padrões.

Alguns algoritmos desse tipo de aprendizado (como o *K-means Clustering*) precisam do conhecimento do número de classes, ou grupos, existentes no conjunto de dados, porém outros (como o *Mean-Shift Clustering*) podem detectar automaticamente o número de grupos presentes no conjunto de dados [45].

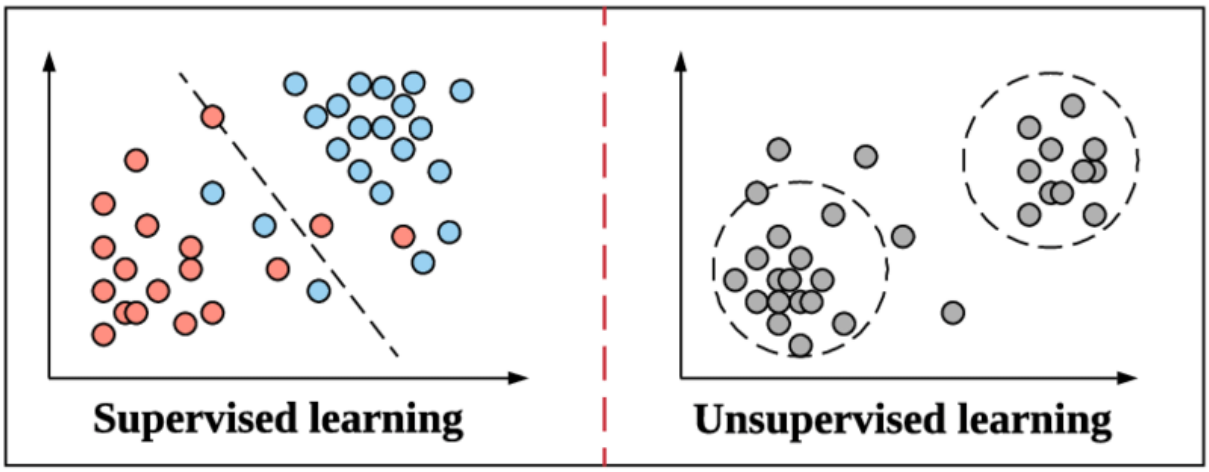

Figura 3.1: Comparação gráfica de aprendizado supervisionado e não-supervisionado [56]

3.1.2 Aprendizado Supervisionado

Já no aprendizado supervisionado, o qual é utilizado neste trabalho, a fase de aprendizado necessita de um conjunto de dados com respostas pré-calculadas para que os algoritmos possam detectar os padrões baseados nesses dados e utilizar essas conclusões retiradas deles para categorizar ou predizer respostas para novos dados.

Esse tipo de aprendizado pode ser subdividido em algoritmos de classificação, os quais tem como resultado valores discretos (que podem também representar variáveis categóricas), e de regressão, os quais tem como resultado valores contínuos.

Muitos dos algoritmos de aprendizado supervisionado podem ser adaptados para ser utilizados tanto como algoritmos de classificação, quanto de regressão, com diferentes graus de sucesso a depender das características do algoritmo em si.

3.1.2.1 *Decision Tree*

Também conhecido como Árvore de Decisão, tem por objetivo gerar uma árvore que pode ser utilizada para classificar as instâncias de um conjunto de dados. Nessa árvore, exemplificada pela Figura 3.2, nós representam comparações que podem ser aplicadas a uma instância do conjunto de dados, e arestas representam os possíveis resultados das comparações de forma que, para descobrir a classe de uma dada instância é necessário apenas seguir a árvore da raiz até uma de suas folhas.

Dentre as demais aplicações desse algoritmo, temos:

- Detecção de Isquemia Cardíaca [31]
- Mapeamento da Cobertura da Terra [19]
- Predição de Resultado de Testes de Tuberculose [21]

- Estimação de Perigo de Intoxicação em Ambientes [9]
- Predição de Consumo de Energia Elétrica [54]
- Previsão Meteorológica [40]
- Diagnóstico de Diabetes Tipo II [4]

3.1.2.2 *Random Forest*

O algoritmo *Random Forest* gera várias Árvores de Decisão para um mesmo conjunto de dados, como pode ser visto na Figura 3.3, de forma que para classificar uma dada instância, todas as Árvores de Decisão são consultadas para obtenção de resultados parciais, os quais são utilizados por um critério de decisão para determinar a classificação da *Random Forest* para a instância.

O funcionamento de uma *Random Forest* pode ser visto como um processo de votação, onde cada uma das Árvores de Decisão vota em seu resultado e a *Random Forest* decide (por meio de algum algoritmo de escolha) qual será o resultado final baseado nos votos das Árvores de Decisão.

Outras aplicações para esse algoritmo são:

- Classificação de Faces [22]
- Seleção de Genes [11]
- Classificação de Dados Ecológicos [10]
- Identificação de Genes de Susceptibilidade a Doenças Complexas [8]
- Predição de Toxicidade Aquática [41]
- Mapeamento de Potencial de Água Subterrânea [37]

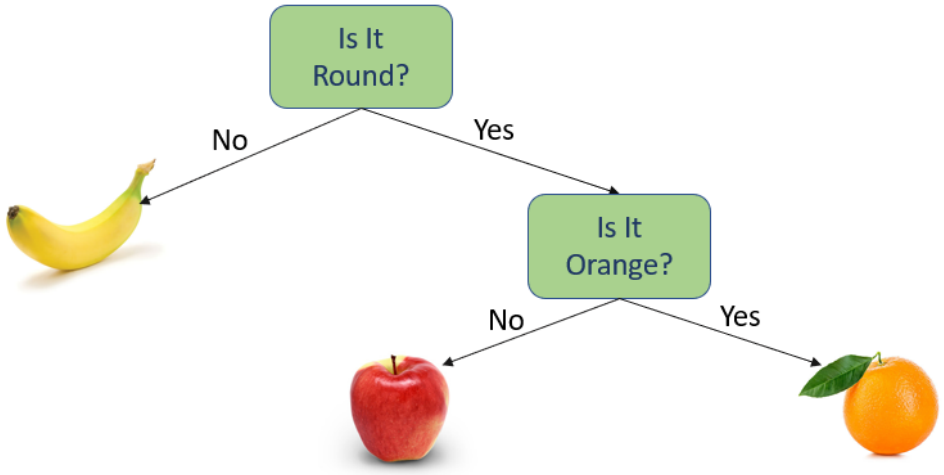

Figura 3.2: Exemplo de Árvore de Decisão [55]

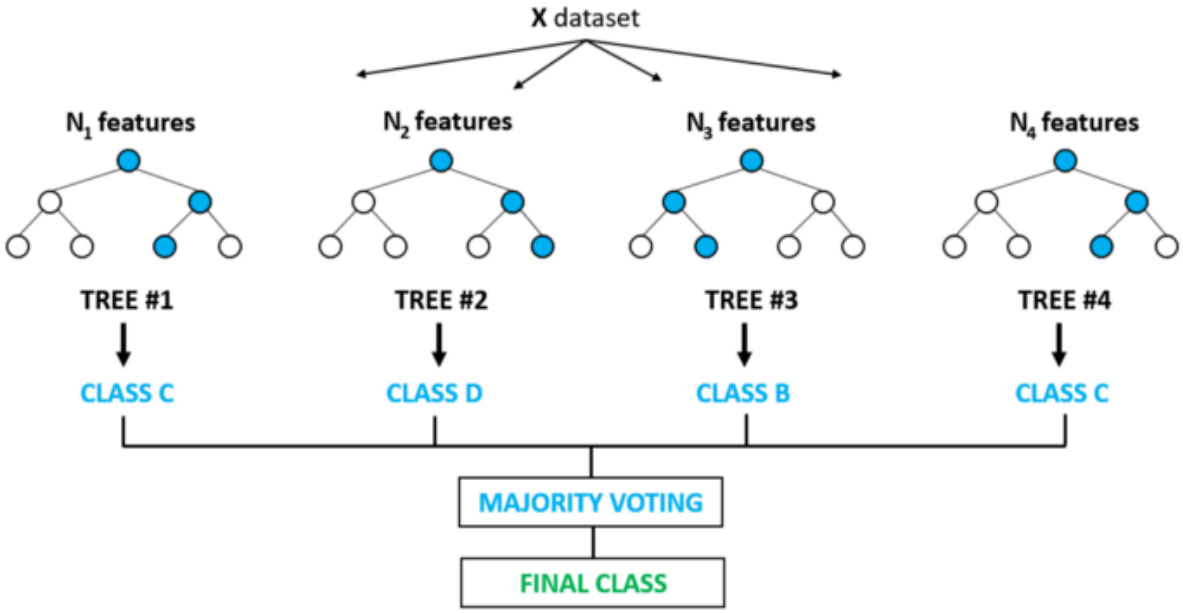

Figura 3.3: Exemplo de *Random Forest* [43]

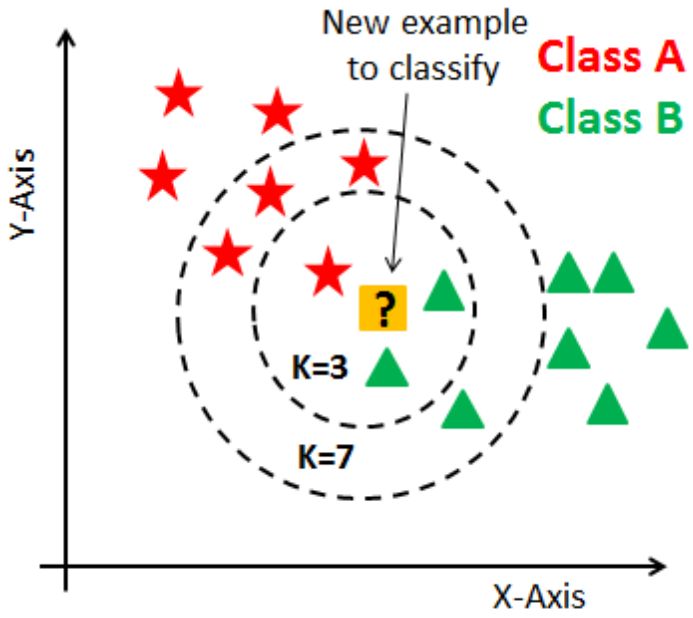

Figura 3.4: Exemplo de KNN [27]

3.1.2.3 *K-Nearest Neighbors*

Mais conhecido como KNN, esse algoritmo utiliza as instâncias do conjunto de dados como parte de um hiperplano. Novas instâncias são classificadas de acordo com os vizinhos próximos destas instâncias no hiperplano, geralmente utilizando a distância euclidiana para o calculo de distância e com número K de vizinhos a serem checados para a classificação. Esse processo pode ser visto na Figura 3.4.

Esse algoritmo também tem diversas outras aplicações, como:

- Ranqueamento de Páginas da *Web* [38]
- Otimização de Data Centers [12]
- Detecção de Falhas na Produção de Semicondutores [24]
- Sistemas de Recomendação [51]
- Detecção de Anomalia de Rede [28]

3.1.2.4 *Linear Discriminant Analysis*

Apesar de ser um dos algoritmos utilizados para redução de dimensionalidade, o algoritmo de Análise Discriminante Linear disponível na biblioteca *Scikit Learn*, utilizada neste trabalho, também pode ser utilizado como classificador [52].

O teorema de Bayes, o qual indica a probabilidade de um evento (o valor da variável alvo ser k) acontecer dado que outro evento (os valores das *features* para a amostra serem $x_1, x_2, ..., x_n$) aconteceu, é utilizado como base para a classificação, o que nos leva à seguinte fórmula:

$$P(y = k|x) = \frac{P(x|y = k)P(y = k)}{P(x)} = \frac{P(x|y = k)P(y = k)}{\sum_l P(x|y = l) \cdot P(y = l)} \tag{3.1}$$

Onde k e l representam classes do problema, x representa uma amostra do conjunto de dados, e y é uma possível classificação da amostra. A classificação de x feita pelo algoritmo é dada pela classe k que maximiza o valor da fórmula acima.

Como esse algoritmo também serve para redução de *features*, ele é mais presente que os demais algoritmos listados nesse capítulo, pois na maioria de suas aplicações ele é acoplado a outros algoritmos de predição. Dentre suas aplicações, temos:

- Redução de *Features* para Classificação de Sinais de Eletroencefalograma [50]
- Classificação de Gasolina por Origem e Tipo [5]
- Classificação Geográfica de Vinhos Vermelhos [30]

4

Metodologia

Neste capítulo será detalhado como foram realizados os experimentos deste estudo, incluindo descrição das ferramentas utilizadas no desenvolvimento, do problema escolhido para os testes, dos algoritmos de aprendizado de máquina utilizados nas comparações, das métricas utilizadas para avaliação e, por fim, detalhes sobre as reduções feitas sobre o conjunto de dados.

4.1 Ferramentas utilizadas

Para este trabalho foi usada a linguagem de programação *Python* e as bibliotecas NLTK (*Natural Language Toolkit*), *Scikit Learn*, *Numpy* e *Pandas*. Como plataforma de programação foi usado o site *Kaggle*, para que o programa fosse mantido e executado em nuvem, o que elimina alguns dos problemas que a execução local pode causar, como problemas com falta de energia interrompendo a execução do programa e a necessidade de se configurar todo sistema em que o programa precisasse ser executado.

Vale ressaltar que versões de bibliotecas são gerenciadas pelo próprio *Kaggle*, então não é possível saber quais versões exatas foram utilizadas, isso provavelmente se dá por conta de que diferentes execuções de um mesmo *Kaggle Notebook* podem utilizar versões levemente diferentes da mesma biblioteca caso haja alguma atualização na mesma. A única informação que a plataforma dá em relação a isso é a versão da linguagem utilizada, que foi o *Python* 3.7.

Dentre as bibliotecas utilizadas, as bibliotecas *Scikit Learn*, que contém diversos algoritmos e técnicas de pré-processamento utilizadas para aprendizado de máquina, e NLKT (*Natural Language Toolkit*), que é composta por diversas ferramentas para facilitar o processamento de linguagem natural, foram as mais importantes, pois estão envolvidas diretamente com os processos necessários para a análise de sentimentos.

Porém, a biblioteca *Numpy*, que contém estruturas de dados e funções para lidar com operações com números, *arrays* e matrizes numéricas, e a biblioteca *Pandas*, que está relacionada

com o conjunto de dados e operações feitas sobre ele como um todo; também foram fundamentais ao processo, pois abstraem grande parte das operações que de outra forma precisariam ser codificadas manualmente e estariam mais vulneráveis a erro e menos otimizadas em geral.

O site *Kaggle*, mesmo sendo uma ferramenta extremamente útil por conter além de um ambiente de programação, uma grande biblioteca de conjuntos de dados que servem para os mais diversos tipos de aplicações; possui algumas limitações em seu modo gratuito, especialmente quanto ao tempo de execução, o qual não pode passar de 9 horas, e ao consumo de memória, o qual não pode passar de 16 gigabytes.

Essas limitações da plataforma limitaram os algoritmos de aprendizado de máquina que puderam ser analisados, pois alguns dos algoritmos tomam muito tempo de execução ou gastam muita memória durante sua execução, como o algoritmo SVM (*Support Vector Machine*), que foi utilizado apenas em fases muito iniciais do estudo e foi retirado por conta dessas limitações, para permitir um estudo mais aprofundado.

4.2 Descrição do problema

O problema escolhido para a comparação dos algoritmos provém de uma base de dados disponível no *Kaggle* que se trata de avaliações de cursos do portal *Coursera*, onde temos o texto da avaliação relacionado ao número de estrelas - o qual vai de um a cinco - escolhido pelo usuário que fez a avaliação [2]. Dessa forma, podemos ver que se trata de um problema de classificação multi-classe, onde cada classe é representada por um número de estrelas.

4.3 Algoritmos de aprendizado de máquina analisados

O algoritmo mais usado para análise de sentimento é o SVM (*Support Vector Machine*) [36, 35], porém, como citado anteriormente, não foi viável o seu uso nessa análise por conta das restrições da ferramenta escolhida. No entanto, os algoritmos utilizados neste trabalho também veem bastante uso para análise de sentimento em geral [48, 34, 42, 26, 7].

Dentre os escolhidos temos os algoritmos *Random Forest*, *Decision Tree*, KNN (*K-Nearest Neighbors*) e a Análise Discriminante Linear (*Linear Discriminant Analysis*), todos classificadores, visto que o problema escolhido é de classificação. Esses algoritmos estão disponíveis na biblioteca *Scikit Learn* citada anteriormente.

4.3.1 *Cross-validation*

A execução dos algoritmos de aprendizado foi feita utilizando *cross-validation*, uma técnica bastante conhecida que separa um conjunto de dados em partes de mesmo tamanho e executa os algoritmos várias vezes usando uma dessas partes como conjunto de teste e o restante como conjunto de treinamento.

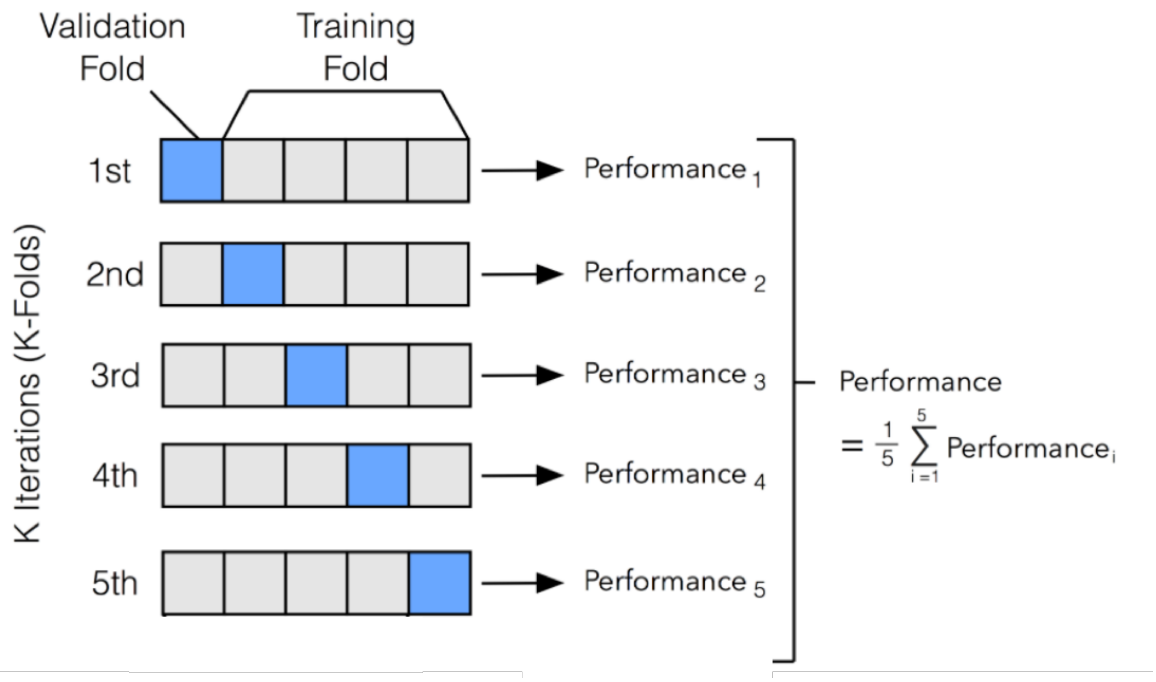

Figura 4.1: Exemplo de *Cross-validation* [29]

Cada uma dessas execuções utiliza separações (mais comumente conhecidas como *splits*) diferentes do conjunto de dados, onde a parte utilizada para teste é sempre diferente para cada *split*, como pode ser visto na Figura 4.1. Nesse trabalho foram utilizados 5 *splits*, dessa forma, obteremos cinco resultados para cada combinação de métrica, algoritmo e método de redução de *features*, uma vez que cada um desses resultados será de uma execução diferente do algoritmo com um *split* diferente do conjunto de dados.

4.4 Métricas utilizadas

Dentre as métricas utilizadas nesse trabalho estão Acurácia, Acurácia Balanceada, *Precision*, *Recall* e *Negative Logarithmic Loss*. A métrica *F1-score*, apesar de ser uma métrica interessante, foi removida pois ela pode ser facilmente calculada com os resultados de Precision e Recall.

4.4.1 Acurácia e Acurácia Balanceada

Possivelmente a métrica mais conhecida entre as escolhidas, a acurácia indica a proporção de acertos do modelo em relação ao número total de predições. Porém em certos casos, onde se há uma base de dados desbalanceada - como o caso do conjunto de dados utilizado, que tem uma classe com cerca de duas mil instâncias e outra com mais de setenta mil - é necessário o uso de uma métrica como a acurácia balanceada, a qual basicamente realiza o calculo da acurácia para

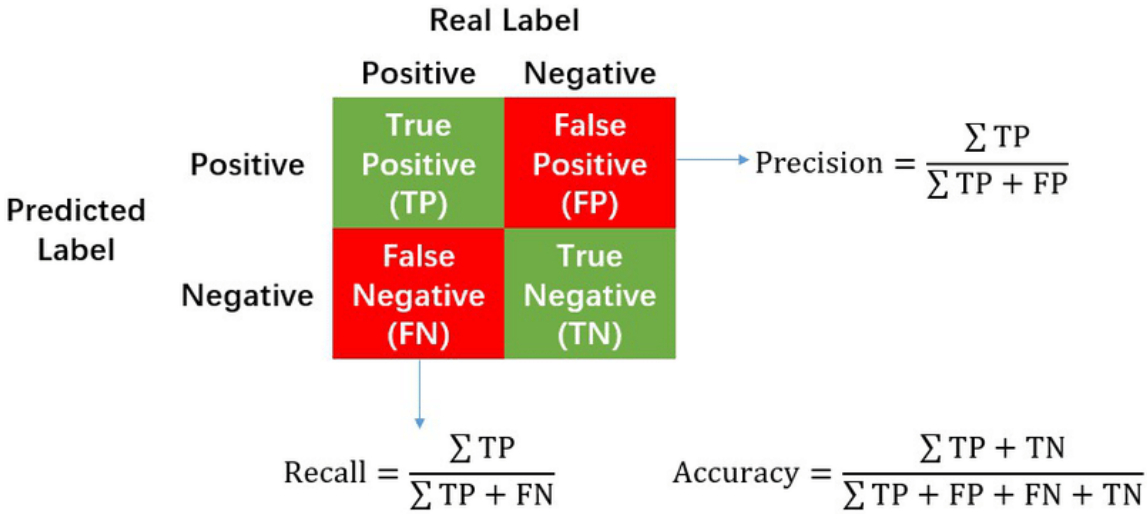

Figura 4.2: Fórmulas de Acurácia, *Precision* e *Recall* [32]

cada classe do conjunto de dados separadamente e depois faz a média entre esses resultados.

4.4.2 *Precision* e *Recall*

Apesar de serem métricas menos conhecidas em geral, *precision* e *recall* são muito utilizadas para problemas ligados à medicina. *Precision* indica a proporção entre o número de acertos de uma dada classe e o número total de predições feitas daquela classe, ou seja, acertos somados aos erros. Já o *recall* indica a proporção do número de acertos para a dada classe e o número total de elementos dessa classe no conjunto de dados - note que esse número total não é relacionado às predições feitas, mas sim aos dados já disponíveis na base de dados. Calculos dessas métricas, assim como da acurácia, podem ser vistos na Figura 4.2

4.4.3 *Negative Logarithmic Loss*

Essa é uma métrica menos conhecida em relação às outras utilizadas neste estudo, porém ela é útil pois leva em consideração o quanto de certeza o modelo teve de que a predição estava correta. No entanto, seu uso está mais relacionado a comparações dentro de um mesmo mesmo problema ou dentro de uma mesma classe de problemas, visto que essa métrica não é limitada, como vamos ver adiante.

Dessa forma, essa métrica foi escolhida para que em trabalhos futuros possa haver uma comparação não apenas dos acertos e erros do modelo, mas também levando em consideração a certeza do modelo em relação às predições. Para calculo dessa métrica em problemas multi-classe - como o escolhido para essa análise - a seguinte fórmula é utilizada [53]:

$$NegLogLoss = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k}\log p_{i,k} \tag{4.1}$$

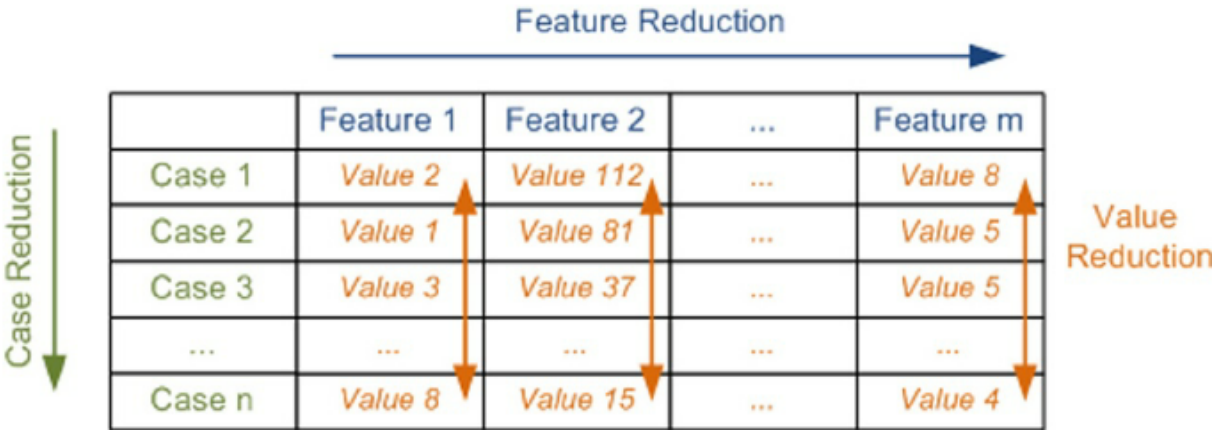

Figura 4.3: Tipos de redução de dados [49]

Onde N e K são, respectivamente, o número de instâncias e o número de classes diferentes presentes no conjunto de dados, $y_{i,k}$ é um valor booleano que indica se a instância i pertence à classe k e $p_{i,k}$ indica a probabilidade dada pelo modelo de predição - que também pode ser vista como a certeza do modelo - de que $y_{i,k}$ seja verdadeiro.

Essa métrica é calculada em escala logarítmica para evitar erros de precisão. A versão negativa foi a utilizada pois é a presente na biblioteca *Scikit Learn*, o motivo dela ser negativa é para que identificar o melhor resultado se torne um problema de maximização em vez de minimização, visto que o algoritmo de *Logarithmic Loss* padrão retorna valores entre $[0, \infty)$, onde zero seria o melhor resultado.

4.5 Redução de dados

A redução de dados é um processo fundamental para o Aprendizado de Máquina, uma vez que pode tanto reduzir o tempo de execução quanto aumentar a qualidade das predições dos algoritmos de aprendizado quando aplicado corretamente. Os diferentes tipos de redução são exemplificados na Figura 4.3.

4.5.1 Redução de dimensionalidade

Dentre os métodos comparados para a redução de dimensionalidade neste estudo, estão o algoritmo de Análise Discriminante Linear (também conhecido como LDA), o qual serve tanto como algoritmo de classificação, quanto como método de redução de *features*; o algoritmo de Análise de Componentes Principais (também conhecido como PCA) que, similarmente ao LDA, gera *features* novas baseadas nas inicialmente existentes e as utiliza no lugar das iniciais; o método de redução por correlação, o qual remove as *features* que possuem menor correlação com a variável alvo; a redução pelo método chi-quadrado, a qual remove *features* com menos de 0.05 de p-valor de acordo com o teste de hipótese feito pelo mesmo.

4.5.2 Redução de numerosidade

O conjunto de dados escolhido possui cerca de 100 mil instâncias, porém mais de 70% desses *reviews* são classificados como 5 estrelas e cerca de 17% são classificados como 4 estrelas, com os 13% restantes sendo distribuídos entre os *reviews* classificados como 1, 2 e 3 estrelas.

Assim, foi necessária uma redução do número de instâncias no conjunto, tanto por conta das limitações da plataforma quanto por conta do grande desbalanceamento entre as classes. Após essa redução, tivemos pouco mais de 48 mil *reviews* no conjunto de dados, com cerca de 55% com 5 estrelas e 25% com 4 estrelas.

Essa redução foi feita primeiramente para reduzir o tamanho do conjunto de dados, visto que por limitações da plataforma seria inviável utilizar todas as 100 mil instâncias, mas também para a obtenção de uma melhor proporção para que uma maior parte dos *splits* a serem realizados no processo de *cross-validation* tivessem membros de todas as classes presentes no conjunto de dados.

5

Resultados e Discussões

Neste capítulo serão descritos e discutidos os resultados obtidos a partir dos experimentos realizados para este trabalho. Estes resultados estão, tanto em forma de quatro figuras, 5.1, 5.2, 5.3 e 5.4, sendo uma para cada algoritmo de predição, quanto em forma de Tabelas 5.1, 5.2, 5.3, 5.4 e 5.5, sendo uma para cada uma das métricas avaliadas.

Ainda como resultados deste trabalho, temos o código que foi utilizado para geração desses gráficos e tabelas, o qual está separado em seis Kaggle Notebooks: pré-processamento [18], redução por correlação [15], redução por chi-quadrado [13], redução pelo método PCA [17], redução pelo método LDA [16] e, por fim, comparação dos métodos de redução [14].

5.1 Organização dos resultados

Como descrito na metodologia deste trabalho, quatro algoritmos de aprendizado de máquina foram utilizados, dentre eles estão *Decision Tree*, *Random Forest*, LDA e KNN, para cada um destes algoritmos tivemos uma imagem como resultado.

Em cada imagem há cinco gráficos, cada um deles relacionado com uma métrica diferente. Para cada métrica, temos os resultados do algoritmo para cada um dos métodos de redução de *features* utilizados. Dentre eles, como mencionado no capítulo anterior, estão: redução por correlação, por chi-quadrado, pelo método LDA e pelo método PCA. O melhor resultado de cada gráfico está marcado com um azul mais escuro.

Além disso, há cinco tabelas, uma para cada métrica, onde os resultados estão agrupados por algoritmo de redução de features (*Reducers*) e por algoritmo de classificação (*Classifiers*).

5.2 Discussões

Como pode-se ver nas imagens abaixo, os resultados para os métodos de redução por correlação e por chi-quadrado foram bastante similares, em geral melhores que os resultados produzidos

Reducer	Classifier	Accuracy
Correlation	Random Forest	0.597863 ± 0.001407
	Decision Tree	0.544549 ± 0.001746
	KNN	0.468257 ± 0.002013
	LDA	0.632092 ± 0.001426
Chi-square	Random Forest	0.596265 ± 0.001947
	Decision Tree	0.548698 ± 0.002061
	KNN	0.466908 ± 0.003148
	LDA	0.636552 ± 0.001606
PCA	Random Forest	0.576763 ± 0.001430
	Decision Tree	0.486825 ± 0.003102
	KNN	$\mathbf{0.319190 \pm 0.002264}$
	LDA	0.633298 ± 0.002959
LDA	Random Forest	0.692874 ± 0.015014
	Decision Tree	0.602634 ± 0.007989
	KNN	0.676589 ± 0.014119
	LDA	$\mathbf{0.716493 \pm 0.018630}$

Tabela 5.1: Acurácia agrupada por algoritmos de redução e de classificação

pela redução pelo método PCA.

Porém dentre os métodos testados o maior destaque foi para a redução pelo método LDA, que teve resultados melhores do que os demais métodos em todas as métricas e para todos os algoritmos de aprendizado testados. Mesmo que o LDA tenha sido usado tanto como método de redução de *features* quanto como algoritmo de classificação, seus resultados para os demais algoritmos de classificação demonstram que esse não é um fator que tornaria menos válidos seus bons resultados, quando comparados com os resultados dos demais métodos de redução de *features*.

Mas ainda com esses resultados, para alguns dos algoritmos de classificação o LDA obteve uma taxa de erro padrão na métrica *Negative Log Loss* bastante alta, o que pode indicar que ele tem uma alta chance de levar os algoritmos de classificação a errarem ou acertarem com alto grau de certeza. Um possível motivo é a redução muito grande de *features* feita por ele, visto que ele reduz esse número para ser no máximo igual $C - 1$, onde C é o número de classes do problema. Essa redução drástica é bastante evidente, pois antes da redução o número de features era cerca de três mil e após a redução o problema passou a ter apenas quatro *features*

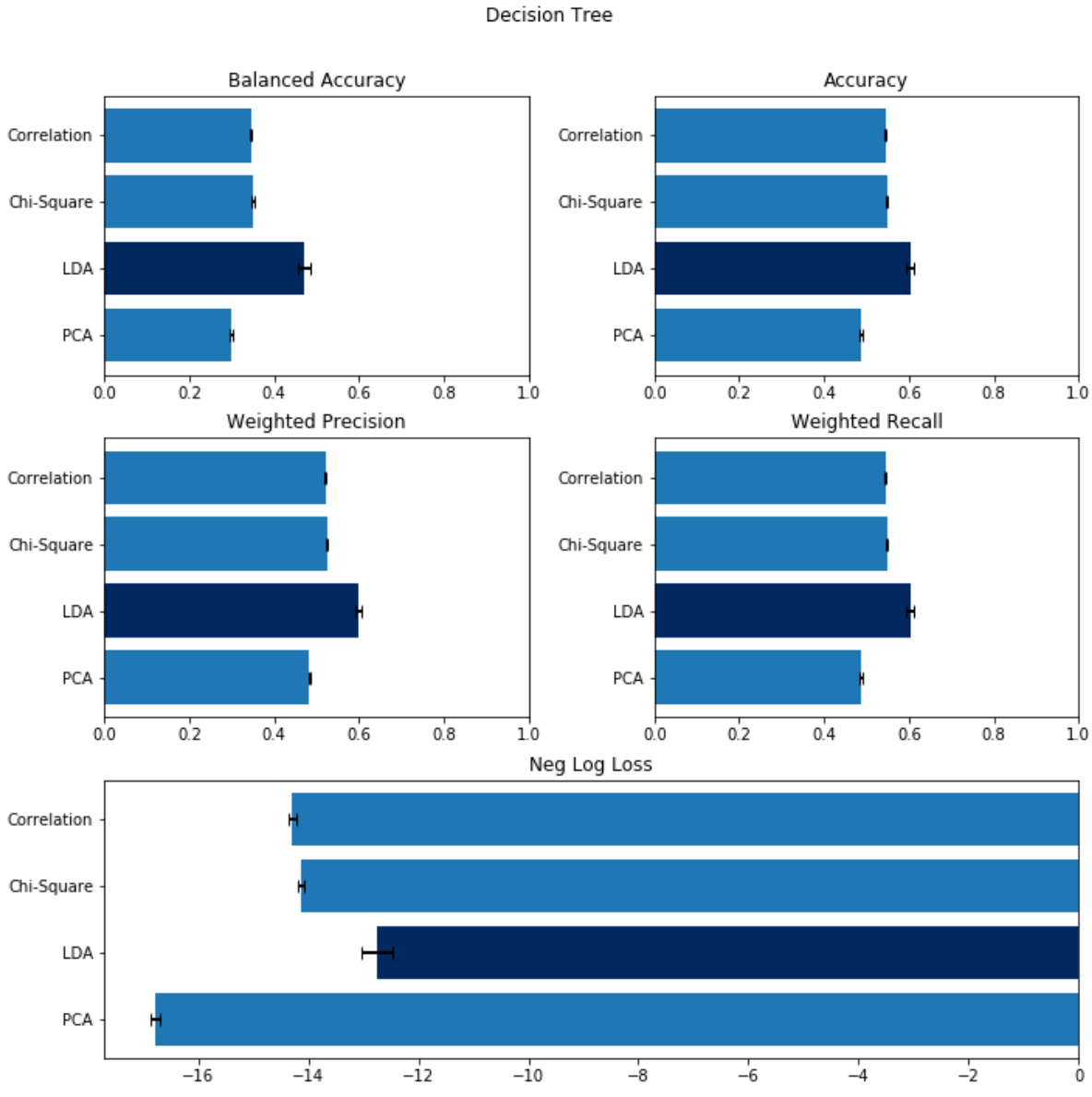

Figura 5.1: Gráficos dos resultados das execuções da Decision Tree

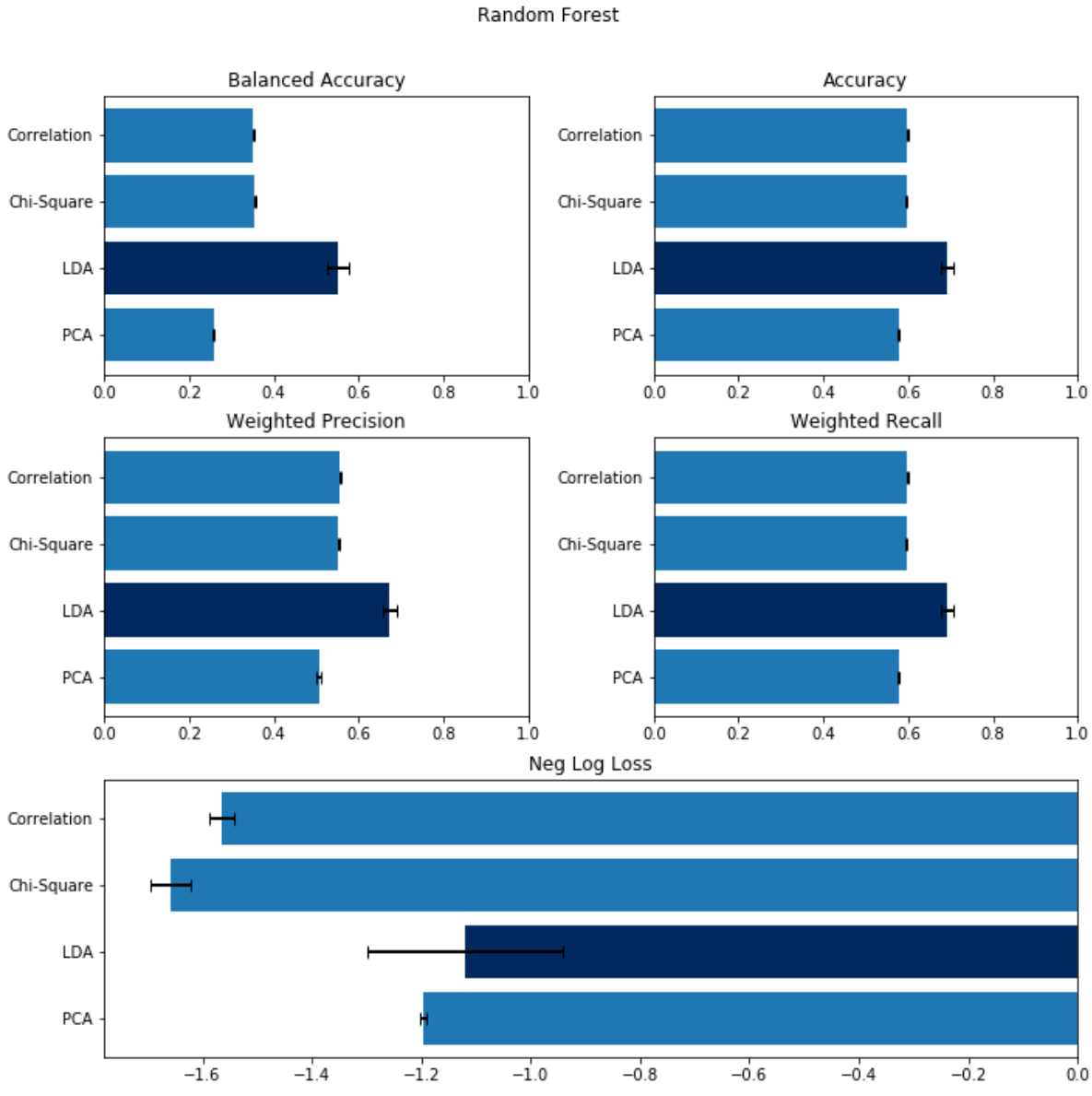

Figura 5.2: Gráficos dos resultados das execuções do Random Forest

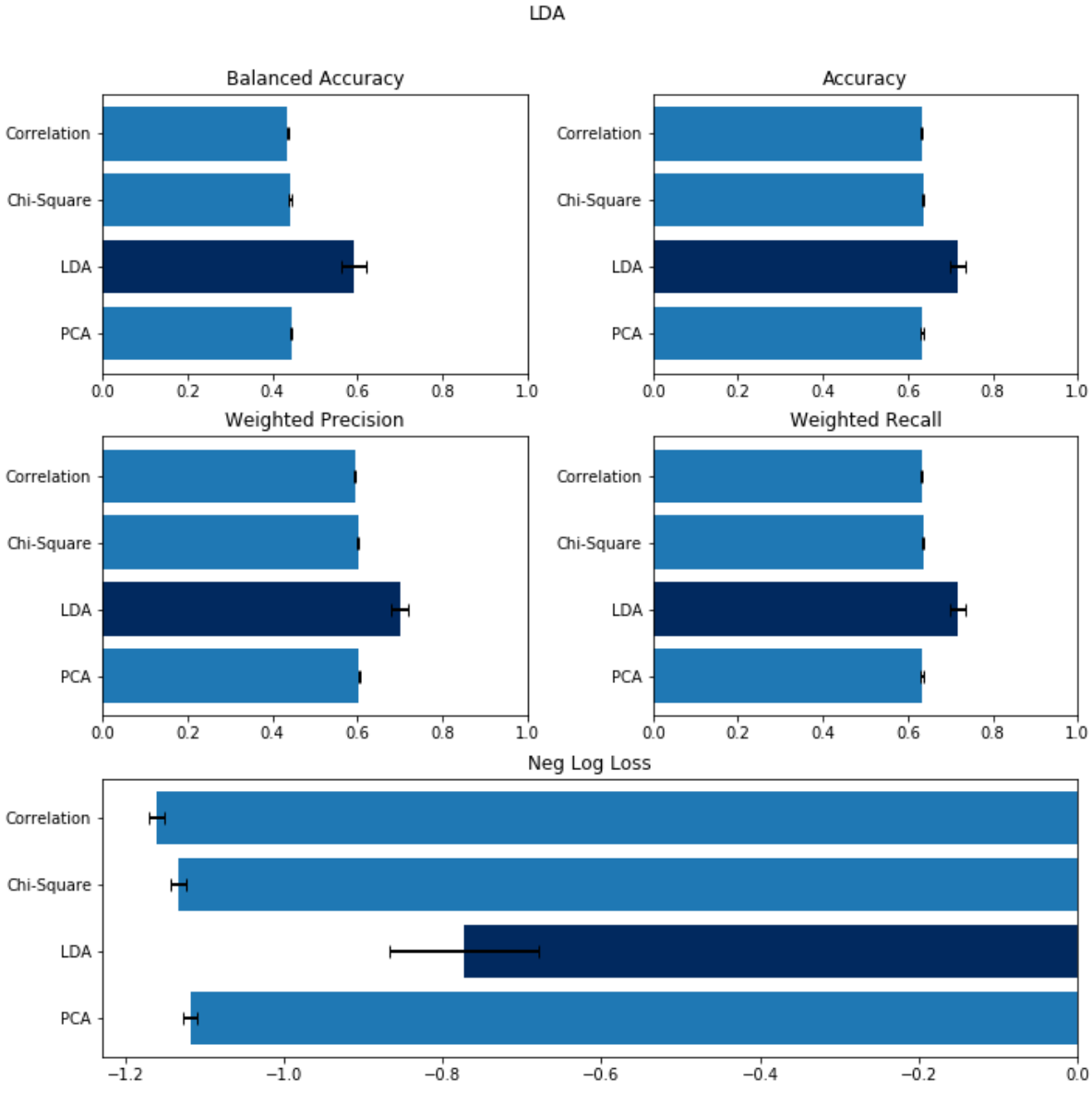

Figura 5.3: Gráficos dos resultados das execuções do LDA

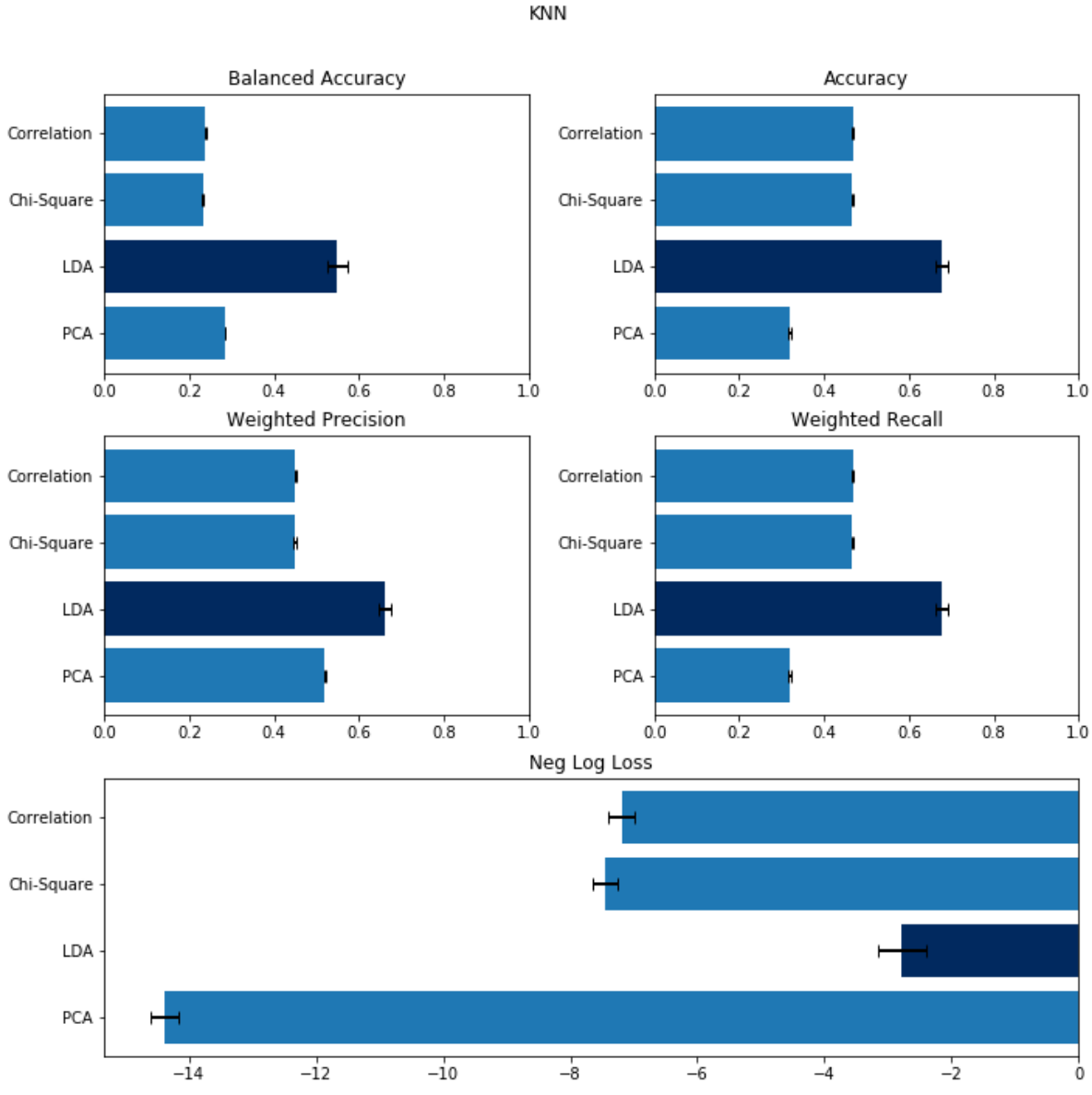

Figura 5.4: Gráficos dos resultados das execuções do KNN

Reducer	Classifier	Balanced Accuracy
Correlation	Random Forest	0.352137 ± 0.001742
	Decision Tree	0.345002 ± 0.001956
	KNN	0.238617 ± 0.002321
	LDA	0.435583 ± 0.002601
Chi-square	Random Forest	0.355418 ± 0.002053
	Decision Tree	0.350496 ± 0.002907
	KNN	$\mathbf{0.232239 \pm 0.002341}$
	LDA	0.441410 ± 0.002434
PCA	Random Forest	0.258335 ± 0.001786
	Decision Tree	0.300412 ± 0.003114
	KNN	0.284797 ± 0.001272
	LDA	0.444812 ± 0.001604
LDA	Random Forest	0.551326 ± 0.024050
	Decision Tree	0.472285 ± 0.014562
	KNN	0.548258 ± 0.023804
	LDA	$\mathbf{0.591876 \pm 0.030271}$

Tabela 5.2: Acurácia Balanceada agrupada por algoritmos de redução e de classificação

Reducer	Classifier	Weighted Precision
Correlation	Random Forest	0.555452 ± 0.001836
	Decision Tree	0.521332 ± 0.001382
	KNN	0.450101 ± 0.001801
	LDA	0.594329 ± 0.001711
Chi-square	Random Forest	0.552067 ± 0.002111
	Decision Tree	0.525625 ± 0.001845
	KNN	$\mathbf{0.447944 \pm 0.003011}$
	LDA	0.602637 ± 0.001825
PCA	Random Forest	0.506028 ± 0.003953
	Decision Tree	0.483571 ± 0.002282
	KNN	0.519106 ± 0.001713
	LDA	0.604450 ± 0.003326
LDA	Random Forest	0.673840 ± 0.016142
	Decision Tree	0.598945 ± 0.007917
	KNN	0.659882 ± 0.014624
	LDA	$\mathbf{0.699639 \pm 0.020276}$

Tabela 5.3: Precisão Ponderada agrupada por algoritmos de redução e de classificação

Reducer	Classifier	Weighted Recall
Correlation	Random Forest	0.597863 ± 0.001407
	Decision Tree	0.544549 ± 0.001746
	KNN	0.468257 ± 0.002013
	LDA	0.632092 ± 0.001426
Chi-square	Random Forest	0.596265 ± 0.001947
	Decision Tree	0.548698 ± 0.002061
	KNN	0.466908 ± 0.003148
	LDA	0.636552 ± 0.001606
PCA	Random Forest	0.576763 ± 0.001430
	Decision Tree	0.486825 ± 0.003102
	KNN	$\mathbf{0.319190 \pm 0.002264}$
	LDA	0.633298 ± 0.002959
LDA	Random Forest	0.692874 ± 0.015014
	Decision Tree	0.602634 ± 0.007989
	KNN	0.676589 ± 0.014119
	LDA	$\mathbf{0.716493 \pm 0.018630}$

Tabela 5.4: Recall Ponderado agrupado por algoritmos de redução e de classificação

Reducer	Classifier	Negative Log Loss
Correlation	Random Forest	-1.5646 ± 0.0229
	Decision Tree	-14.2883 ± 0.0627
	KNN	-7.1940 ± 0.2090
	LDA	-1.1602 ± 0.0095
Chi-square	Random Forest	-1.6573 ± 0.0373
	Decision Tree	-14.1329 ± 0.0669
	KNN	-7.4504 ± 0.2016
	LDA	-1.1323 ± 0.0097
PCA	Random Forest	-1.1953 ± 0.0059
	Decision Tree	$\mathbf{-16.7757 \pm 0.0870}$
	KNN	-14.3827 ± 0.2191
	LDA	-1.1176 ± 0.0089
LDA	Random Forest	-1.1199 ± 0.1782
	Decision Tree	-12.7518 ± 0.2830
	KNN	-2.7779 ± 0.3855
	LDA	$\mathbf{-0.7732 \pm 0.0941}$

Tabela 5.5: Perda Logarítmica Negativa agrupada por algoritmos de redução e de classificação

6

Conclusão

Um algoritmo de redução de *features* pode ser visto como apenas uma maneira mais rápida de fazer os processamentos, mas em muitos casos pode ser essencial para um aumento na eficácia dos algoritmos de predição, especialmente quando se trata de dados como textos, os quais geram um alto número de *features*, dentre as quais poucas são realmente relevantes para a análise e podem acabar apenas "confundindo"o algoritmo de aprendizado de máquina.

6.1 Resultados e contribuições

Com esse trabalho, é possível comparar os quatro métodos de redução de *features* utilizados e, para minimização de viés em relação ao algoritmo de aprendizado, quatro diferentes foram utilizados e múltiplas métricas foram analisadas para os resultados.

Dessa forma, como pode ser visto no capítulo anterior, o algoritmo de Análise Discriminante Linear (LDA) foi o algoritmo mais eficiente dentre os escolhidos para redução de *features* e, portanto, é a melhor escolha dentre os algoritmos analisados, pois tanto reduz o tempo computacional de predição quanto obtém melhores resultados que os demais algoritmos de redução comparados nesse estudo.

6.2 Limitações deste estudo

Como já mencionado no capítulo 4, este trabalho teve algumas limitações relacionadas às ferramentas utilizadas, dentre elas está a inviabilidade do uso de alguns algoritmos por conta do alto custo computacional necessário, tanto em tempo quanto em espaço de memória.

Outra limitação está relacionada à base de dados escolhida, pois pôde ser visto que nem todos os *reviews* existentes na base de dados estavam escritos na mesma língua, o que pode ter

causado um grande aumento na quantidade de *features* reconhecidas pelo processo de reconhecimento de termos nos documentos, dos quais alguns possivelmente descrevessem apenas um pequeno subconjunto dos documentos (os relacionados à lingua do termo), e assim podem ter sido removidos na etapa de redução de *features*, assim possivelmente reduzindo a chance de que os algoritmos de aprendizado tivessem resultados positivos em relação a alguns dos textos.

6.3 Trabalhos futuros

O principal objetivo deste trabalho é de apresentar comparações claras entre algoritmos de redução de *features* tentando minimizar o viés possivelmente gerado pela escolha do algoritmo de predição. Trabalhos futuros poderiam comparar outros algoritmos de redução de *features* com o LDA, e o PCA, que foram os que obtiveram melhor performance dentre os aqui comparados. Outras abordagens seriam buscar expandir a quantidade ou variedade dos algoritmos de predição utilizados ou utilizar outras métricas para comparação dos dados.

Referências bibliográficas

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Jan C M Adona. 100k coursera's course reviews dataset. URL https://www.kaggle.com/septa97/100k-courseras-course-reviews-dataset. Acessado em 14/03/2021.

[3] Elvin Aghammadzada. Stop words (removal) nlp. URL https://www.kaggle.com/learn-forum/186390. Acessado em 09/06/2021.

[4] Asma A Al Jarullah. Decision tree discovery for the diagnosis of type ii diabetes. In *2011 International conference on innovations in information technology*, pages 303–307. IEEE, 2011.

[5] Roman M Balabin and Ravilya Z Safieva. Gasoline classification by source and type based on near infrared (nir) spectroscopy data. *Fuel*, 87(7):1096–1101, 2008.

[6] Fabrício Benevenuto, Filipe Ribeiro, and Matheus Araújo. Métodos para análise de sentimentos em mídias sociais. In *Brazilian Symposium on Multimedia and the Web.(Webmedia), Manaus, Brazil*, page 30, 2015.

[7] Ameni Bouaziz, Christel Dartigues-Pallez, Célia da Costa Pereira, Frédéric Precioso, and Patrick Lloret. Short text classification using semantic random forest. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 288–299. Springer, 2014.

[8] Alexandre Bureau, Josée Dupuis, Kathleen Falls, Kathryn L Lunetta, Brooke Hayward, Tim P Keith, and Paul Van Eerdewegh. Identifying snps predictive of phenotype using random forests. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society*, 28(2):171–182, 2005.

[9] GM Cramer, RA Ford, and RL Hall. Estimation of toxic hazard—a decision tree approach. *Food and cosmetics toxicology*, 16(3):255–276, 1976.

[10] D Richard Cutler, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jacob Gibson, and Joshua J Lawler. Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007.

[11] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

[12] Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, and Juha Plosila. Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 256–259. IEEE, 2013.

[13] Gabriel F. B. Freire. Nlp chi-square, . URL https://www.kaggle.com/fabriond/nlp-chi-square. Acessado em 09/06/2021.

[14] Gabriel F. B. Freire. Nlp comparison, . URL https://www.kaggle.com/fabriond/nlp-comparison. Acessado em 09/06/2021.

[15] Gabriel F. B. Freire. Nlp correlation, . URL https://www.kaggle.com/fabriond/nlp-correlation. Acessado em 09/06/2021.

[16] Gabriel F. B. Freire. Nlp lda, . URL https://www.kaggle.com/fabriond/nlp-lda. Acessado em 09/06/2021.

[17] Gabriel F. B. Freire. Nlp pca, . URL https://www.kaggle.com/fabriond/nlp-pca. Acessado em 09/06/2021.

[18] Gabriel F. B. Freire. Nlp preprocessing, . URL https://www.kaggle.com/fabriond/nlp-preprocessing. Acessado em 09/06/2021.

[19] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[20] Carmen Galvez, Félix de Moya-Anegón, and Victor H Solana. Term conflation methods in information retrieval. *Journal of Documentation*, 2005.

[21] Lynn B Gerald, Shenghui Tang, Frank Bruce, David Redden, Michael E Kimerling, Nancy Brook, Nancy Dunlap, and William C Bailey. A decision tree for tuberculosis contact investigation. *American journal of respiratory and critical care medicine*, 166(8): 1122–1127, 2002.

[22] Vidyut Ghosal, Paras Tikmani, and Phalguni Gupta. Face classification using gabor wavelets and random forest. In *2009 Canadian Conference on Computer and Robot Vision*, pages 68–73. IEEE, 2009.

[23] Mohammad Haghighat, Mohamed Abdel-Mottaleb, and Wadee Alhalabi. Discriminant correlation analysis: Real-time feature level fusion for multimodal biometric recognition. *IEEE Transactions on Information Forensics and Security*, 11(9):1984–1996, 2016.

[24] Q Peter He and Jin Wang. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE transactions on semiconductor manufacturing*, 20(4):345–354, 2007.

[25] Samuel H Huang. Dimensionality reduction in automatic knowledge acquisition: a simple greedy search approach. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1364–1373, 2003.

[26] M Ikonomakis, Sotiris Kotsiantis, and V Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[27] Vinit Kumar. Vinit kumar's answer to "how does the k- nearest neighbour algorithm work?". URL https://www.quora.com/How-does-the-K-Nearest-Neighbour-algorithm-work/answer/Vinit-Kumar-244. Acessado em 09/06/2021.

[28] Yang Li, Binxing Fang, Li Guo, and You Chen. Network anomaly detection based on tcm-knn algorithm. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 13–19, 2007.

[29] Ethen Liu. Machine learning model selection. URL http://ethen8181.github.io/machine-learning/model_selection/model_selection.html. Acessado em 09/06/2021.

[30] L Liu, D Cozzolino, WU Cynkar, M Gishen, and CB Colby. Geographic classification of spanish and australian tempranillo red wines by visible and near-infrared spectroscopy combined with multivariate analysis. *Journal of agricultural and food chemistry*, 54(18): 6754–6759, 2006.

[31] William J Long, John L Griffith, Harry P Selker, and Ralph B D'agostino. A comparison of logistic regression to decision-tree induction in a medical domain. *Computers and Biomedical Research*, 26(1):74–97, 1993.

[32] Jun Ma. Analyzing the leading causes of traffic fatalities using xgboost and grid-based analysis: A city management perspective - scientific figure on researchgate. URL https://www.researchgate.net/figure/Calculation-of-Precision-Recall-and-Accuracy-in-the-confusion-matrix_fig3_336402347. Acessado em 09/06/2021.

[33] Aleix M. Martínez and Avinash C. Kak. Pca versus lda. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:228–233, 2001.

[34] Aiman Moldagulova and Rosnafisah Bte Sulaiman. Using knn algorithm for classification of textual documents. In *2017 8th International Conference on Information Technology (ICIT)*, pages 665–671. IEEE, 2017.

[35] Rodrigo Moraes, João Francisco Valiati, and Wilson P Gavião Neto. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633, 2013.

[36] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 412–418, 2004.

[37] Seyed Amir Naghibi, Kourosh Ahmadi, and Alireza Daneshi. Application of support vector machine, random forest, and genetic algorithm optimized random forest models in groundwater potential mapping. *Water Resources Management*, 31(9):2761–2775, 2017.

[38] Charul Nigam and AK Sharma. Experimental performance analysis of web recommendation model in web usage mining using knn page ranking classification approach. *Materials Today: Proceedings*, 2020.

[39] Federico Pascual. Twitter sentiment analysis with machine learning. URL https://monkeylearn.com/blog/sentiment-analysis-of-twitter/. Acessado em: 12/07/2020.

[40] Elia Georgiana Petre. A decision tree for weather prediction. *PP*, 77:82, 2009.

[41] Pavel G Polishchuk, Eugene N Muratov, Anatoly G Artemenko, Oleg G Kolumbin, Nail N Muratov, and Victor E Kuz'min. Application of random forest approach to qsar prediction of aquatic toxicity. *Journal of chemical information and modeling*, 49(11): 2481–2488, 2009.

[42] Tomas Pranckevičius and Virginijus Marcinkevičius. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221, 2017.

[43] Abilash R. Applying random forest (classification) — machine learning algorithm from scratch with real datasets. URL https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-w Acessado em 09/06/2021.

[44] Sebastian Raschka. Linear discriminant analysis – bit by bit. URL https://sebastianraschka.com/Articles/2014_python_lda.html. Acessado em 09/06/2021.

[45] George Seif. The 5 clustering algorithms data scientists need to know. URL https://towardsdatascience.com/linear-discriminant-analysis-in-python-76b8b17817c2. Acessado em: 01/11/2020.

[46] Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1661–1666. IEEE, 2003.

[47] Sachin Singh. Stemming in natural language processing. URL https://www.c-sharpcorner.com/blogs/stemming-in-natural-language-processing. Acessado em 09/06/2021.

[48] Pascal Soucy and Guy W Mineau. A simple knn algorithm for text categorization. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 647–648. IEEE, 2001.

[49] Marco Spruit. Improving short-term demand forecasting for short-lifecycle consumer products with data mining techniques - scientific figure on researchgate. URL https://www.researchgate.net/figure/Three-types-of-data-reduction-techniques_fig1_282564978. Acessado em 09/06/2021.

[50] Abdulhamit Subasi and M Ismail Gursoy. Eeg signal classification using pca, ica, lda and support vector machines. *Expert systems with applications*, 37(12):8659–8666, 2010.

[51] V Subramaniyaswamy and R Logesh. Adaptive knn based recommender system through mining of user preferences. *Wireless Personal Communications*, 97(2):2229–2247, 2017.

[52] Scikit Learn Team. Linear and quadratic discriminant analysis, . URL https://scikit-learn.org/stable/modules/lda_qda.html. Acessado em: 28/06/2020.

[53] Scikit Learn Team. Metrics and scoring: quantifying the quality of predictions, . URL https://scikit-learn.org/stable/modules/model_evaluation.html. Acessado em: 28/06/2020.

[54] Geoffrey KF Tso and Kelvin KW Yau. Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. *Energy*, 32(9): 1761–1768, 2007.

[55] Jon Wayland. Jon wayland's answer to "what is the intuitive concept of ensembling in machine learning?". URL https://www.quora.com/What-is-the-intuitive-concept-of-ensembling-in-machine-learning/answer/Jon-Wayland. Acessado em 09/06/2021.

[56] Zhenyu Wen. Orchestrating the development lifecycle of machine learning-based iot applications: A taxonomy and survey - scientific figure on researchgate. URL https://www.researchgate.net/figure/Examples-of-Supervised-Learning-Linear-Regression-and-Unsupervised-Learning_fig3_336642133. Acessado em 09/06/2021.

[57] Tom Yuz. A sentiment analysis approach to predicting stock returns. URL https://medium.com/@tomyuz/a-sentiment-analysis-approach-to-predicting-stock-returns-d5ca8b75a42. Acessado em 09/06/2021.

[58] George Kingsley Zipf. Human behavior and the principle of least effort. 1949.