UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
CURSO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

LUIS GUSTAVO FERREIRA LOPES

APRENDIZADO DE MÁQUINA APLICADO À ANÁLISE DE SENTIMENTOS

CARAÚBAS
2023

LUIS GUSTAVO FERREIRA LOPES

APRENDIZADO DE MÁQUINA APLICADO À ANÁLISE DE SENTIMENTOS

Trabalho de conclusão de curso apresentado a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Orientador: Prof.ª Dr. Zenner Silva Pereira

CARAÚBAS

2023

L864a Lopes, Luis Gustavo Ferreira.
Aprendizado de máquina aplicado à análise de sentimentos / Luis Gustavo Ferreira Lopes. - 2023.
69 f. : il.

Orientador: Zenner Silva Pereira.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência e Tecnologia, 2023.

1. Análise de sentimentos. 2. Classificação de textos. 3. Aprendizado de máquina. 4. Processamento de texto. I. Pereira, Zenner Silva, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Caraúbas
Bibliotecária: Dalvanira Brito Rodrigues
CRB: 15/700

LUIS GUSTAVO FERREIRA LOPES

APRENDIZADO DE MÁQUINA APLICADO À ANÁLISE DE SENTIMENTOS

Trabalho de conclusão de curso apresentado a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Defendida em: 26/09/2023

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
ZENNER SILVA PEREIRA
Data: 11/10/2023 16:42:47-0300
Verifique em https://validar.iti.gov.br

Profa. Dr. Zenner Silva Pereira (UFERSA)
Presidente

gov.br Documento assinado digitalmente
ANA TEREZA DE ABREU LIMA
Data: 11/10/2023 11:28:54-0300
Verifique em https://validar.iti.gov.br

Prof. Dra. Ana Tereza de Abreu Lima (UFERSA)
Membro Examinador

gov.br Documento assinado digitalmente
MAURICIO ZULUAGA MARTINEZ
Data: 11/10/2023 16:36:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Mauricio Zuluaga Martinez (UFERSA)
Membro Examinador

Dedico este trabalho aos meus pais, Anaci e Lusimar que tanto se esforçaram para que eu pudesse terminar minha primeira graduação.

AGRADECIMENTOS

À Deus, pela existência das pessoas que nomeio a seguir.

Agradeço a toda a minha família, mas em especial aos meus pais Anaci Lopes e Lusimar Ferreira por acreditarem em meu sonho e me incentivarem nos momentos difíceis enfrentados, sobretudo por nunca ter deixado eu passar necessidades mesmo nos momentos mais desafiadores. Sou profundamente grato por sua confiança inabalável em meu potencial e pelo apoio incansável que sempre me ofereceram. Sua dedicação e amor incondicional são verdadeiramente inestimáveis, e sou privilegiado por tê-los como pais.

Agradeço ao meu orientador professor Dr. Zenner Silva Pereira que não desistiu de mim nos momentos mais desafiadores e esteve sempre presente para me orientar e encorajar. Sua dedicação, paciência e apoio foram fundamentais para o meu crescimento acadêmico e pessoal. Sou imensamente grato a ele por acreditar em meu potencial e por nunca desistir de mim, mesmo diante das adversidades. Sua orientação e sabedoria foram essenciais para o sucesso desta jornada e sou grato por tê-lo como meu orientador.

Agradeço a todos os professores que contribuíram com a minha formação acadêmica, por compartilharem do seu conhecimento e acima de tudo pelas palavras de apoio e incentivo no decorrer dos últimos anos. Sem dúvidas, foram essenciais para que eu chegasse até aqui.

Agradeço aos meus colegas e amigos que conheci durante a vida acadêmica, por compartilharmos dessa experiência juntos, por dividirmos das mesmas aflições, e pela troca de conhecimento contínuo. Mas, acima de tudo pela amizade, aqui construí verdadeiros amigos, alguns que considero como irmãos.

Expresso minha gratidão a todos os servidores da Universidade Federal Rural do Semi-Árido (UFERSA) que compõem a equipe responsável pelo campus de Caraúbas. Agradeço profundamente por sua dedicação incansável em preservar e cuidar dessa estrutura, a qual tive o privilégio de desfrutar durante toda a minha graduação em Ciência e Tecnologia. Sua atuação diária é fundamental para manter um ambiente propício ao aprendizado e crescimento dos estudantes, e sou imensamente grato pelo trabalho árduo de cada um de vocês.

Nada te perturbe, Nada te espante,
Tudo passa, Deus não muda,
A paciência tudo alcança;
Quem a Deus tem, Nada lhe falta:
Só Deus basta.

Santa Tereza D’Avila

RESUMO

A inteligência artificial tem se destacado como a grande protagonista da revolução tecnológica, moldando a forma como as pessoas vivem, uma vez que possibilita a análise de grandes volumes de dados e a realização de diversas tarefas, como a análise de sentimentos. A análise de sentimentos é uma atividade que envolve a identificação dos sentimentos expressos em determinados textos, categorizando-os como positivos, negativos ou neutros. Esse processo implica no tratamento computacional das opiniões contidas nos textos. Atualmente, diversos estudos se concentram nessa área, porém a análise de sentimentos ainda está evoluindo e continuará a demandar pesquisas para aprimorar a precisão e a compreensão da comunicação humana. Diante disso, neste trabalho, foram abordados os principais métodos e processos empregados para realizar a análise de sentimentos em críticas de filmes, com o objetivo de identificar se os sentimentos presentes nos comentários sobre os filmes são positivos ou negativos e assim validar a eficiência dos classificadores. Foi utilizado o aprendizado de máquina para treinar três tipos diferentes de algoritmos, permitindo que eles aprendam padrões relevantes capazes de identificar os sentimentos presentes nos comentários de filmes. Foi utilizada uma base de dados contendo aproximadamente 50 mil dados, todos eles rotulados, uma vez que se trata de um processo de aprendizado supervisionado. Além disso, técnicas de validação e processamento de texto foram aplicadas para garantir uma eficácia aprimorada dos algoritmos, visando determinar quais são mais adequadas para a tarefa de classificação de emoções. Após terem sido treinados e testados com a base de dados, os modelos desenvolvidos foram submetidos a avaliações adicionais, nas quais novos comentários foram utilizados para testar seu desempenho e capacidade de generalização. Contudo, os três modelos alcançaram acurácias que ficaram entre 73% e 77% nos testes realizados com os dados da base de dados. Enquanto na tarefa de classificar novas frases, os algoritmos conseguiram acertar todas os novos comentários submetidos. De acordo com esses resultados, a efetividade das técnicas de processamento de texto e aprendizado de máquina foram comprovadas, mas ainda precisam ser aprimoradas, visto que estamos em constante evolução.

Palavras-chave: análise de sentimentos; classificação de textos; aprendizado de máquina; processamento de texto.

ABSTRACT

Artificial intelligence has stood out as the great protagonist of the technological revolution, shaping the way people live, as it enables the analysis of large volumes of data and the performance of various tasks, such as sentiment analysis. Sentiment analysis is an activity that involves identifying the feelings expressed in certain texts, categorizing them as positive, negative or neutral. This process implies the computational treatment of the opinions contained in the texts. Currently, several studies are focused on this area, but sentiment analysis is still evolving and will continue to require research to improve the accuracy and understanding of human communication. Therefore, in this work, the main methods and processes used to perform sentiment analysis in film reviews were addressed, with the aim of identifying whether the feelings present in comments about the films are positive or negative and thus validate the efficiency of the classifiers. Machine learning was used to train three different types of algorithms, allowing them to learn relevant patterns capable of identifying the sentiments present in movie comments. A database containing approximately 50 thousand pieces of data was used, all of them labeled, since it is a supervised learning process. In addition, validation and text processing techniques were applied to ensure an improved efficiency of the algorithms, aiming to determine which ones are most suitable for the emotion classification task. After being trained and tested with the database, the developed models were submitted to additional evaluations, in which new comments were used to test their performance and generalization capacity. However, the three models reached accuracies that were between 73% and 77% in the tests carried out with the data from the database. While in the task of classifying new phrases, the algorithms managed to get all the new comments submitted right. According to these results, the effectiveness of text processing and machine learning techniques have been proven, but still need to be improved, as we are in constant evolution.

Keywords: sentiment analysis; text classification; machine learning; text processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
PLN	Processamento de Linguagem Natural
AM	Aprendizado de Máquina
TF	*Term Frequency*
IDF	*Inverse Document Frequency*
TF-IDF	*Term Frequency-Inverse Document Frequency*
NLTK	*Natural Language Toolkit*
SVM	*Support Vector Machine*
SVC	*Linear Support Vector Classification*
CBOW	*continuous bag-of-words*

LISTA DE SÍMBOLOS

$\sum$	Somatório
%	Porcentagem
θ	Letra minúscula theta, representando o limiar de ativação

SUMÁRIO

1 INTRODUÇÃO

Desde o surgimento da humanidade, os seres humanos têm evoluído tecnologicamente de forma significativa, trazendo consigo uma série de inovações para a sociedade capaz de auxiliar pessoas, empresas e negócios. Com o passar do tempo novas tecnologias são criadas e outras são aperfeiçoadas para render um melhor desempenho possível. Dentre essas tecnologias podemos citar a inteligência artificial (IA) que está cada vez mais presente no dia-dia das pessoas e vem transformando o modo como vivemos.

Até alguns anos atrás, a área da IA era bastante teórica, com aplicações em problemas não muito complexos, mas em problemas curiosos com pouco valor prático (Facelli *et al.*, 2011). Segundo (Facelli *et al.*, 2011) a partir dos anos 1970, ocorreu uma ampla difusão do emprego de técnicas de computação fundamentadas em inteligência artificial para resolver problemas concretos, resultando em progressos notáveis em múltiplos campos.

Dessa forma, a IA tem o potencial de transformar fundamentalmente a forma como vivemos e trabalhamos, permitindo que máquinas executem tarefas complexas e aprendam com a experiência, tornando-se cada vez mais eficientes e precisas. Trata-se de uma área multidisciplinar, se dividindo em vários segmentos, cada qual com suas aplicações específicas para determinado projeto, abrangendo desde o processamento de linguagem natural (PLN) e visão computacional até a robótica e o aprendizado de máquina (AM), proporcionando soluções inovadoras para resolver problemas desafiadores em diversos domínios.

Podemos dizer que a tecnologia em si é um dos domínios que usufrui das vantagens proporcionadas pela Inteligência Artificial, com o desenvolvimento de assistentes virtuais, *chatbots*, reconhecimento de fala, processamento de linguagem natural, interfaces de usuário, entre outros. Isso se deve em grande parte à difusão da internet, que proporciona uma ampla gama de oportunidades para a integração de tecnologias inteligentes em nossa vida cotidiana.

Nesse cenário amplo e diversificado de aplicações, a inteligência artificial (IA) emerge como uma ferramenta crucial que beneficia inúmeras empresas e setores, aprimorando a eficiência operacional e contribuindo para uma tomada de decisão mais precisa. Algumas das empresas que incorporam com destaque a IA em suas operações incluem: Amazon, que emprega a IA em sua plataforma e-commerce para oferecer produtos aos clientes, e na assistente virtual Alexa. Google, que integra a IA em ferramentas como o Google Assistant, Google Tradutor e Google Fotos. Microsoft, que utiliza a IA no pacote Office 365 e na assistente virtual Cortana. E também a Netflix, que faz uso da IA para personalizar recomendações de filmes e séries aos clientes.

A IA também desempenha um papel essencial em diversas áreas, tais como: saúde, onde auxilia no diagnóstico de doenças por meio da análise de imagens provenientes de exames médicos. Segurança, com aplicações na prevenção de fraudes financeiras e na detecção de conteúdo malicioso em mensagens de spam. Marketing, onde a IA é fundamental na análise de sentimentos (um subcampo do PLN) das pessoas em relação a produtos ou serviços, além de contribuir para uma segmentação mais precisa do público-alvo. E também na agricultura, onde é usada para otimizar processos e monitorar em tempo real as condições das lavouras.

Com relação ao processamento de linguagem natural, podemos aplicá-lo para realizar análise de sentimentos presentes em diversos textos de diversas categorias, seja para identificar os sentimentos presentes em comentários do Instagram ou twitter, ou até mesmo para identificar se uma crítica presente em um filme é positiva ou negativa. Para isso, é necessário utilizar o aprendizado de máquina (AM), onde há grandes volumes de dados previamente rotulados (aprendizado supervisionado) com seus respectivos sentimentos. Desta forma, por meio de modelos de AM torna-se possível ensinar computadores a reconhecer padrões nas expressões linguísticas, permitindo a classificação do sentimento presente nos textos.

Acerca do Aprendizado de Máquina, temos que esse segmento se concentra no desenvolvimento de algoritmos e modelos permitindo que sistemas computacionais aprendam a partir de dados. Em outras palavras, o propósito do Machine Learning é criar algoritmos que possam aprender a partir dos dados de entrada e fornecer saídas precisas e relevantes.

No entanto, a área de aprendizado de máquina ainda não é muito instintiva visto que existem muitas técnicas para as variedades de problemas. Dessa forma, o trabalho consiste em uma revisão de métodos e processos utilizados para análise de sentimentos presentes em críticas de filmes, visando identificar as abordagens mais eficazes e promissoras, bem como explorar novas metodologias e ferramentas emergentes nessa área em constante evolução.

2 OBJETIVOS

2.1 OBJETIVOS GERAL

Este trabalho tem como objetivo realizar um estudo abrangente sobre diversos algoritmos, técnicas de processamento de textos e aprendizado de máquina, aplicados à análise de sentimentos em bases de dados de avaliações de filmes. Além disso, busca-se avaliar diferentes classificadores a fim de identificar o melhor desempenho na categorização e avaliação de emoções expressas em textos.

2.2 OBJETIVOS ESPECÍFICOS

- Estudar as principais técnicas para processamento de texto, como validação cruzada, remoção de stopwords e geração de vetores de atributos.
- Aplicar técnicas de processamento de texto estudadas aos conjuntos de dados usando Python
- Implementar algoritmos de aprendizado de máquina e de redes neurais artificiais, em linguagem Python.
- Pesquisar os principais algoritmos de aprendizado de máquina utilizados em tarefas de classificação.
- Avaliar os resultados obtidos em cada algoritmo implementado.

3 REFERENCIAL TEÓRICO

3.1 APRENDIZADO DE MÁQUINA

Segundo (Géron, 2019, p. 22, tradução nossa) *Machine Learning* (Aprendizado de Máquina) “é a ciência ou a arte de programar computadores para que eles possam aprender com dados”. Assim, podemos dizer que se trata de uma abordagem computacional que permite que os sistemas sejam treinados para reconhecer padrões nos dados e tomar decisões ou fazer previsões com base nesses padrões, sem a necessidade de serem explicitamente programados para cada cenário específico. Ao aprender com os dados, o AM capacita os computadores a se adaptarem e melhorarem seu desempenho à medida que são expostos a mais informações, oferecendo uma forma eficiente de lidar com problemas complexos e lidar com conjuntos de dados de grande escala.

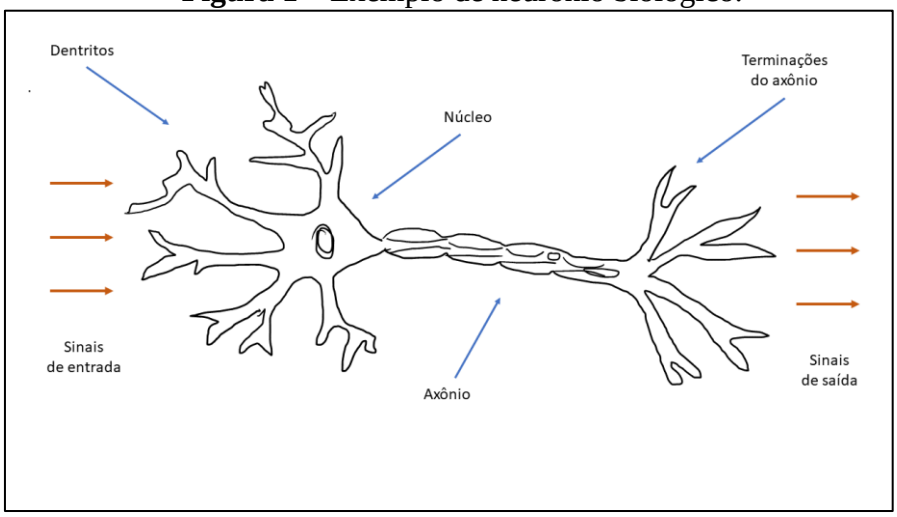
No entanto, para que isso seja possível é necessário compreender como o AM funciona, ou seja, em que ele se baseia e quais são seus fundamentos. De acordo com (Feltrin, 2020) ao longo dos primeiros estudos relacionados aos aspectos biológicos e psicológicos dos mecanismos de aprendizado, buscamos compreender nossa própria biologia, resultando na criação de modelos lógicos estruturados. Esses modelos têm como objetivo abstrair mecanismos, permitindo-nos conduzir pesquisas científicas sobre eles.

Através da evolução da neurociência, conseguimos entender cada vez mais como o nosso cérebro funciona, desde os mecanismos físicos e biológicos até aqueles que nos tornam autoconscientes e cientes de como aprendemos à medida que experimentamos o mundo. (Feltrin, 2020).

Com isso, as primeiras tentativas de desenvolvermos mecanismos de forma lógico-computacional foram baseados em nosso cérebro e assim foram recriadas estruturas anatômicas e mecanismos de aprendizado de forma artificial com redes neurais artificiais (Feltrin, 2020).

Existe vários tipos de redes neurais artificiais que servem para um propósito em específico. Todas possuem um ponto em comum, esse ponto em comum são as camadas de unidades que são conhecidas como neurônios artificiais ou perceptrons. Interações entre esses neurônios são criadas para que seja possível encontrar padrões de dados permitindo que as informações computacionais sejam processadas com um tempo muito superior aos limites humanos (Feltrin, 2020). A Figura 1 exemplifica um tipo de neurônio biológico. Podemos observar que ele é a base fundamental para as redes neurais artificiais.

Figura 1 – Exemplo de neurônio biológico.



Fonte: Svg silh (2023), adaptado.

Percebemos que esse neurônio biológico possui entradas que podemos chamar de porta lógica simples com saídas binárias; vários sinais chegam aos dendritos, são integrados ao corpo celular e quando o sinal chega a um determinado limite um sinal de saída é gerado e será repassado pelo axônio até a saída (Raschaka; Mirjalili, 2019, tradução minha). Os modelos computacionais de redes neurais artificiais seguem esse princípio fundamental. Eles adquirem conhecimento por meio de um processo de aprendizagem que requer sinais de entrada. Esses sinais, por sua vez, são processados por meio das conexões entre os neurônios, conhecidas como pesos sinápticos, que desempenham o papel crucial de armazenar o conhecimento adquirido. Após isso, entra em cena uma função que desempenha o papel de transmitir os sinais de saída. Ela realiza essa tarefa combinando os sinais de entrada com os pesos sinápticos, tudo isso em busca de alcançar um objetivo específico. A saída gerada será comparada com um limiar de ativação que determina quando o neurônio deve ser ativado ou não. Portanto, podendo ser vista como uma máquina adaptativa (Haykin, 1999).

Dessa forma, AM se concentra no desenvolvimento de algoritmos e modelos que são capazes de tomar decisões automáticas e a fazer predições a partir de dados, ou seja, pode-se utilizar redes neurais artificias para executar essas tarefas. Entretanto, outros tipos de abordagem também podem ser utilizados. Assim sendo, o campo do Aprendizado de Máquina engloba uma diversidade de estratégias para a criação e treinamento de modelos, que serão minuciosamente exploradas nas seções subsequentes.

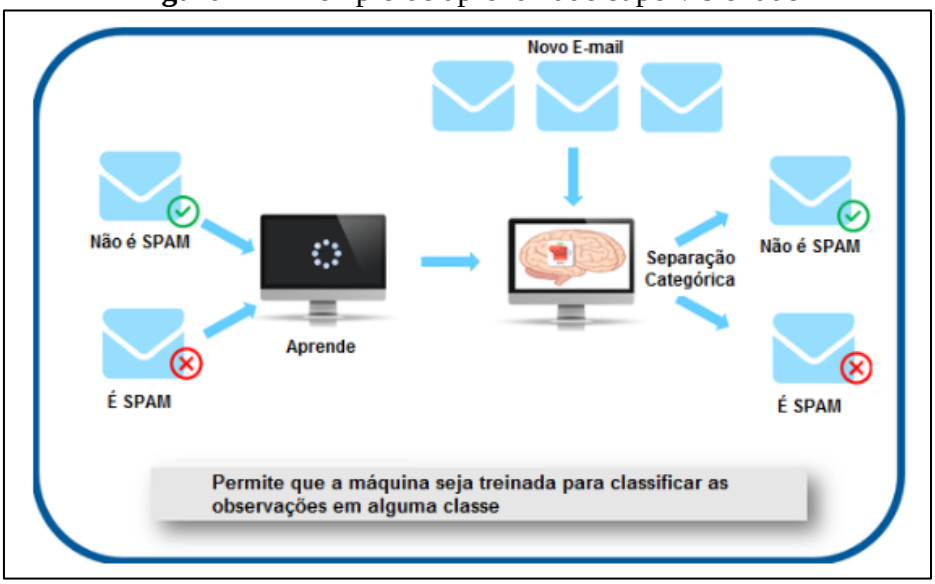
3.2 TIPOS DE APRENDIZADO

O aprendizado de máquina pode ser utilizado em diversas tarefas, que podem ser organizadas com diferentes critérios, no entanto, é necessário adotar o melhor tipo de aprendizado para lidar com essas tarefas, ou seja, é fundamental selecionar o aprendizado de máquina mais adequado para aquela determinada situação. Nesse contexto, existe três categorias fundamentais: aprendizado supervisionado, não supervisionado ou por reforço. Estes, serão abordados nas próximas seções aprofundando nossa compreensão acerca de suas características e usos específicos.

3.2.1 Aprendizado Supervisionado

No contexto do Aprendizado Supervisionado, o conjunto de treinamento fornecido ao algoritmo contém as soluções desejadas, conhecidas como rótulos, ou seja, os dados já são rotulados (Géron, 2019, tradução minha). Dessa forma, no aprendizado supervisionado os dados são rotulados com as soluções desejadas, o que permite que o algoritmo de aprendizado supervisionado aprenda a mapear corretamente as entradas para as saídas correspondentes. Essa rotulagem dos dados é essencial para o treinamento e a avaliação do modelo, fornecendo uma referência clara para a aprendizagem e permitindo a validação da precisão das previsões ou classificações feitas pelo modelo. Esse tipo de aprendizado pode ser usado para classificação, processamento de linguagem natural, reconhecimento de padrões, entre outros. Para este trabalho será usado esse tipo de aprendizado. A Figura 2 exemplifica de forma clara como ocorre o aprendizado supervisionado.

Figura 2 – Exemplo de aprendizado supervisionado



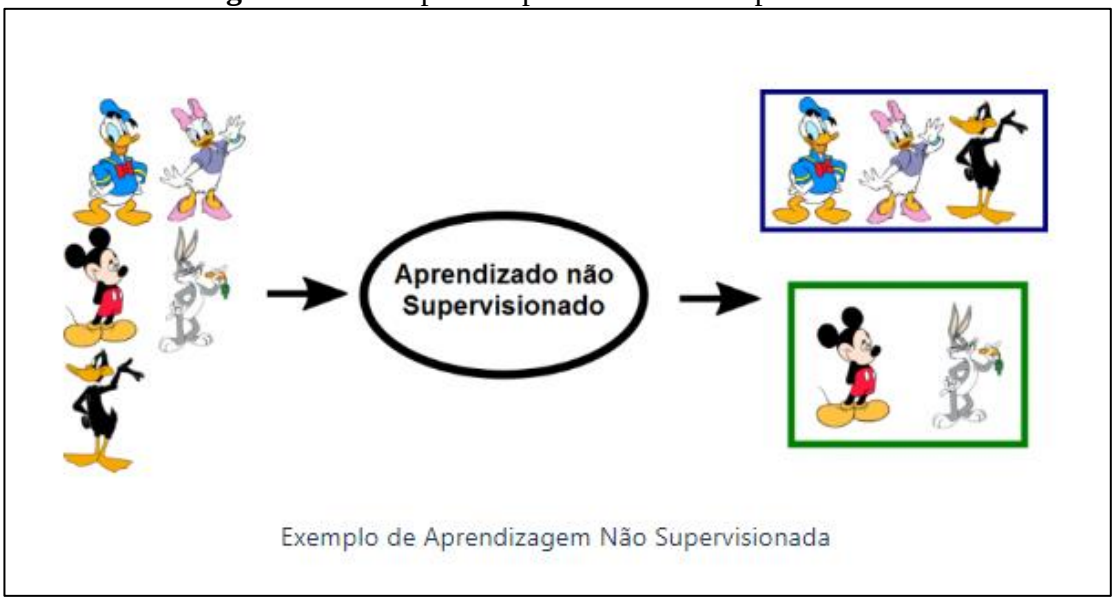
Fonte: Clavera (2019)

A Figura 2 ilustra de maneira clara o processo de aprendizado supervisionado. Neste processo, os dados de entrada, acompanhados de seus respectivos rótulos, são fornecidos ao modelo. Como resultado, o modelo é capaz de aprender com esses exemplos de treinamento. Quando novos dados, não previamente observados, são apresentados ao algoritmo de classificação, ele é competente em realizar uma categorização precisa dos dados com base nas informações de entrada. Em outras palavras, o sistema é capaz de fornecer uma saída que classifica de forma significativa os novos dados, proporcionando assim uma valiosa capacidade de tomada de decisões e análise.

3.2.2 Aprendizado Não Supervisionado

No Aprendizado Não Supervisionado os dados de treinamento não são rotulados, ou seja, não contém rótulos. O sistema tenta aprender sem um "professor", isto é, não existe um supervisor no processo de aprendizagem (Géron, 2019, tradução minha). Com isso o algoritmo tenta encontrar padrões, estruturas ou agrupamentos intrínsecos aos dados de entrada sem a orientação explicita dos dados. Esse tipo de aprendizado pode ser usado para recomendação automática, detecção de anomalias, entre outros. A Figura 3 ilustra como ocorre esse tipo de aprendizado.

Figura 3 – Exemplo de aprendizado não supervisionado



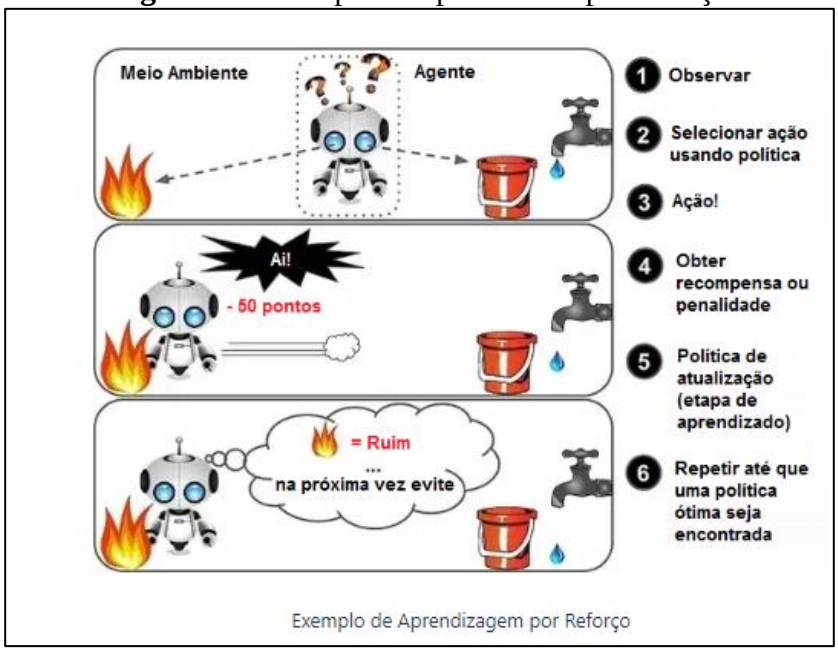
Fonte: Clavera (2019)

Nesse contexto, o modelo consegue identificar padrões comuns nos dados, o que permite efetuar uma categorização deles em grupos que compartilham essas características específicas. Ao analisarmos a Figura 3, notamos que os dados são retratados por meio de personagens, sendo que alguns compartilham características em comum, como pertencerem à mesma espécie. A partir dessa observação o modelo de aprendizado é capaz de categorizar os dados, dividindo-os em dois grupos: um que reúne dados com características similares e outro que agrupa dados que se destacam por suas diferenças.

3.2.3 Aprendizado por Reforço

O Aprendizado por Reforço é um modelo bastante diferente. Nesse contexto, o agente, que é o sistema de aprendizagem, tem a capacidade de observar o ambiente, escolher e executar ações, e receber recompensas em troca dessas ações. Assim, o agente deve adquirir autonomamente o conhecimento sobre a melhor estratégia, denominada política, para maximizar as recompensas ao longo do tempo. A política define a ação que o agente deve escolher quando se encontra em uma determinada situação (Géron, 2019, tradução minha). Esse tipo de aprendizado pode ser usado para controle de processos, jogos, robótica entre outros. A Figura 4 mostra como ocorre o processo do aprendizado por reforço.

Figura 4 – Exemplo de aprendizado por reforço



Fonte: Clavera (2019)

Ao observamos a Figura 4, percebemos que o robô age como o agente principal, sendo necessário tomar ações que resultem em recompensas ou penalidades. Nesse exemplo, o robô enfrenta duas opções de ação: dirigir-se ao fogo ou dirigir-se à água. Quando o robô opta por ir em direção ao fogo, ele imediatamente reconhece que essa ação resulta em uma consequência negativa, ou seja, o fogo é prejudicial. Esse aprendizado leva o robô a evitar essa ação da próxima vez, buscando assim maximizar seu desempenho e evitar futuras penalidades.

Para este trabalho será utilizado o aprendizado supervisionado, onde a principal tarefa abordada diz respeito à análise das críticas de filmes por meio de classificação. A questão central desse desafio está em determinar se uma crítica carrega um sentimento positivo ou negativo. Nas próximas seções, serão delineados os conceitos, parâmetros e algoritmos fundamentais empregados para abordar essa tarefa no âmbito do aprendizado de máquina.

3.3 CONCEITOS E DEFINIÇÕES

Neste trabalho, são apresentadas diversas definições e conceitos fundamentais de AM que são aplicados e utilizados para enriquecer a compreensão do projeto desenvolvido. Estas definições são apresentadas a seguir, visando proporcionar ao leitor um entendimento completo do processo.

3.3.1 Dados

Dados são informações ou valores que são coletados, armazenados e processados. Eles podem ser representados de várias formas, como números, textos, imagens, vídeos, áudios, entre outros formatos. Os dados são a matéria-prima da análise e da tomada de decisões em diversos campos, entre eles o AM. Esses dados são geralmente mantidos em bases de dados bem estruturadas, caracterizadas por diversos atributos (Bellini, 2019).

3.3.2 Atributo

Segundo (Bellini, 2019) um atributo é uma descrição de uma característica do exemplo em análise, ou seja, os atributos são as variáveis que representam informações sobre os objetos ou eventos que estamos estudando. Podendo ser classificado como nominal ou contínuo. Além disso, um atributo pode não possuir um valor quando não é aplicável a um exemplo específico. Diversos tipos de atributos podem ser considerados, como aqueles de natureza numérica,

categórica ou ordinal. A gama de atributos é detalhadamente exposta em etiquetas descritivas conhecida como rótulos.

3.3.3 Classe

Uma classe, também conhecida como rótulo, é um tipo especial de atributo de um exemplo de dado que se busca prever e/ou aprender (Bellini, 2019). No contexto deste estudo, essas classes adotam valores nominais, uma vez que as críticas extraídas da base de dados são categorizadas como positivas ou negativas. Com isso, esses rótulos são distribuídos ao longo das bases de dados utilizadas.

3.3.4 Conjunto de Dados

Um conjunto de dados refere-se a uma coleção de exemplos que é composta por seus respectivos valores de atributos e a classe associada. Esses conjuntos são usados para treinar e avaliar os modelos gerados pelos algoritmos de AM. Os conjuntos são divididos em dois subconjuntos distintos: o conjunto de treinamento e o conjunto de teste (Bellini, 2019). Desse modo, um classificador consegue assimilar informações a partir dos dados de treinamento e posteriormente aplicar esse conhecimento aos dados de teste.

3.3.5 Classificador

Um classificador, também chamado de hipótese, é o modelo criado por um algoritmo a partir de um conjunto de exemplos de treinamento. Esse classificador é então usado para prever a classe de novos exemplos fornecidos a ele (Bellini, 2019). No âmbito do Aprendizado de Máquina, uma gama diversificada de classificadores está disponível, cada um adequado a uma tarefa específica, dependendo de suas características. Assim, optar pelo classificador apropriado minimiza as chances de ocorrência de *Overfitting* e *Underfitting*.

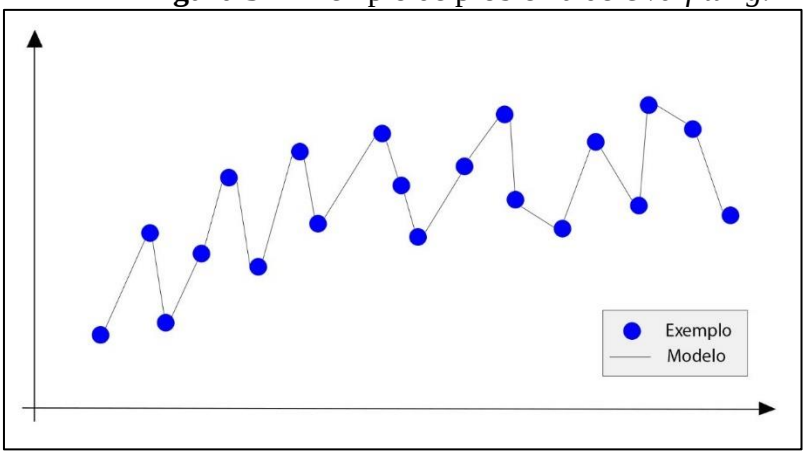
3.3.6 Underfitting e Overfitting

Durante o treinamento de um modelo usando um conjunto de exemplos, podem surgir situações em que o aprendizado é inadequado, seja porque o classificador aprendeu

excessivamente ou não aprendeu o necessário. Esses dois problemas são denominados *Overfitting* (superajuste) e *Underfitting* (sub-ajuste), respectivamente (Bellini, 2019).

No caso de *Overfitting*, o classificador se adapta excessivamente ao conjunto de treinamento fornecido ao algoritmo e falha em generalizar o aprendizado, ou seja, ele simplesmente memoriza um padrão específico presente nesse conjunto, sendo incapaz de prever novos exemplos com uma boa descrição da realidade ou capacidade de acerto (Bellini, 2019). Na Figura 5 é apresentado um problema de *Overfitting* onde o modelo "decorou" e apenas repetiu os padrões específicos dos dados de treinamento, resultando em uma falta de generalização para novos exemplos.

Figura 5 – Exemplo de problema de *Overfitting*.

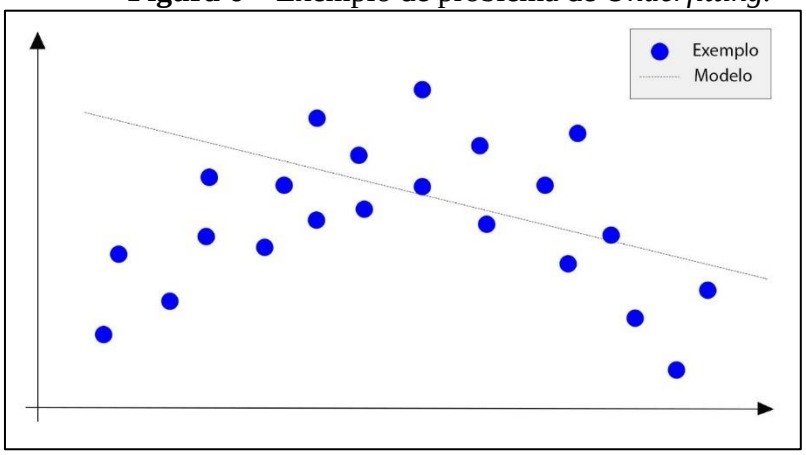


Fonte: Belline (2019), adaptado.

Como podemos observar na Figura 5, o modelo se adaptou demais aos dados, dessa forma, demostrando uma falta de generalização. Isso sugere que o modelo pode não ser capaz de performar bem em dados não vistos anteriormente, comprometendo sua capacidade de extrapolação para situações além do conjunto de treinamento.

Já o *Underfitting* acontece quando o classificador não se ajusta adequadamente ao conjunto de treinamento, ou seja, quando ocorre uma generalização excessiva (Bellini, 2019 *apud* Aalst *et al*., 2010). Isso geralmente ocorre quando o modelo é muito simples ou não possui capacidade suficiente para lidar com a complexidade dos dados. Como resultado, o modelo pode ter um desempenho insatisfatório tanto nos dados de treinamento quanto nos dados de teste. O problema de *Underfitting* é ilustrado na Figura 6.

Figura 6 – Exemplo de problema de *Underfitting.*



Fonte: Belline (2019), adaptado.

Como é visto na Figura 6, o modelo não foi capaz de escolher o melhor caminho para o conjunto de treinamento, ou seja, o modelo enfrentou dificuldades em selecionar a trajetória mais adequada para se ajustar ao conjunto de treinamento, dessa forma, comprometendo sua eficácia e consequentemente demostrando um mal desempenho.

Portanto, é necessário buscar técnicas de otimização a fim de corrigir essa questão e abordar esse problema de forma eficaz. Nas próximas seções, será apresentada uma técnica capaz de aprimorar tanto a performance quanto a capacidade de generalização de modelos de dados: a validação cruzada. No entanto, é necessário ter conhecimento prévios de outros processos que são de fundamental importância para o aprendizado de máquina aplicado à análise de sentimentos. Estes, serão abordados primeiro.

3.4 PROCESSAMENTO DE LINGUAGEM NATURAL

O aprendizado de máquina é um campo com aplicações em várias áreas, dentre elas o processamento de linguagem natural. Segundo (Motta, 2021 *apud* Seif, 2019) O processamento de linguagem natural (PLN) é uma área da inteligência artificial dedicada ao desenvolvimento de teorias e aplicações que permitem que os computadores compreendam e processem a linguagem natural humana em suas diversas formas de manifestação.

Seu surgimento data antes mesmo do surgimento do computador. As primeiras referências ao PLN datam de cerca de 1930, quando foram registradas as primeiras patentes de máquinas de tradução (Motta, 2021 *apud* Oibeau, 2017). Um dos pioneiros e influenciadores mais importantes da área de pesquisa em processamento de linguagem natural foi Alan Turing, que propôs o teste de Turing em seu artigo "*Computing Machinery and Intelligence*". O teste

consistia em avaliar a capacidade de uma máquina exibir comportamentos "inteligentes" que se assemelhem aos de um ser humano, o que impulsionou significativamente o desenvolvimento da IA (Motta, 2021 *apud* Turing, 1950).

Segundo (Motta, 2021, p. 24) “As primeiras tentativas de processar a linguagem natural se deram por meio de tradutores e decodificadores, muitos para tentar decifrar códigos durante a segunda guerra mundial.” O primeiro software de tradução chamado de “*Georgetown Experiment*” foi um dos softwares a ser documentado. O mesmo, tratou-se de um experimento que foi focado na tradução automática de mais de 60 frases de russo para língua inglesa. Contudo, esses softwares de tradução evoluíram aos *chatbots* (Motta, 2021).

Antes de 1980, a maioria dos sistemas de processamento de linguagem natural era baseada em conjuntos complexos de regras escritas manualmente. Entretanto, o aumento do poder computacional permitiu o desenvolvimento de algoritmos de AM, o que possibilitou uma evolução gradual dessa área (Motta, 2021).

Durante os anos 90, houve um aumento significativo na proposição de modelos estatísticos para processamento de linguagem natural. A evolução do aprendizado de máquina também contribuiu para o desenvolvimento do PLN, especialmente com o uso de redes neurais artificiais (Motta, 2021).

O PLN abrange várias áreas. Dentre essas áreas, podemos destacar a análise de sentimentos, que segundo (Crescencio; Gonçalves; Todesco, 2020 *apud* Liu, 2015, p. 3) “é um campo de estudos que analisa as opiniões, sentimentos das pessoas em relação a entidades como produtos, serviços, organizações, indivíduos, questões, eventos, tópicos e seus atributos”. Ela se concentra em identificar a atitude ou emoção presente em um texto, ou seja, diz se ela é positiva, negativa ou neutra.

Assim, a análise de sentimentos utiliza técnicas de aprendizado de máquina para identificar palavras e frases que indicam emoções, opiniões e atitudes expressas em um texto. Podendo ser aplicada em vários tipos de textos, incluindo avaliações de produtos, comentários em redes sociais, resenhas de filmes, notícias, entre outros. Ao analisar os sentimentos expressos em um texto, é possível obter informações valiosas sobre a percepção do público em relação a um produto, serviço, organização ou indivíduo. Isso permite que empresas e organizações possam ajustar sua estratégia de negócios com base nas informações obtidas e melhorar a experiência do cliente. Essa detecção de emoções em trechos textuais pode ser realizada através de dicionários léxicos aliados a algoritmos de aprendizado de máquina para identificar traços de sentimentos como felicidade, frustração, raiva ou tristeza, por exemplo (Motta, 2021).

Dessa forma, a análise de sentimentos é uma área de muito interesse dentro do PLN, por meio da qual é possível a partir de uma entrada de texto, identificar sentimentos expressados nele. A seção 3.5 abordará esse tópico e detalhará seus conceitos e aplicações.

3.5 ANÁLISE DE SENTIMENTOS

De acordo com (Bellini, 2019 *apud* Liu; Zhang, 2012, p. 29) “a análise de sentimentos é uma área de pesquisa dentro do processamento de linguagem natural (PLN), dada pelo estudo computacional das opiniões, avaliações e emoções das pessoas, em relação a outros indivíduos, eventos e produtos”.

(Bellini, 2019 *apud* Pang; Lee, 2008) define o termo “análise de sentimentos” como o processamento computacional das opiniões, sentimentos e subjetividade encontrados em um texto. No entanto, em outros artigos da literatura, ele é definido de forma mais específica como a classificação das avaliações em termos de polaridade, como positiva, negativa ou neutra.

Segundo (Bellini, 2019) a partir do início dos anos 2000, observou-se um aumento significativo na quantidade de pesquisas voltadas para a análise de sentimentos. Esse crescimento pode ser atribuído ao uso cada vez mais frequente de algoritmos de aprendizado de máquina em tarefas de processamento de linguagem natural e recuperação de informações. Além disso, houve uma maior disponibilidade de conjuntos de dados para o treinamento desses algoritmos. O interesse crescente da indústria também desempenhou um papel importante nesse aumento, devido às diversas aplicações potenciais da análise de opiniões, especialmente nas áreas comerciais e de inteligência de negócios.

Contudo, conduzir uma análise de sentimento não constitui uma tarefa trivial. Em outras palavras, para obter resultados precisos na análise de sentimentos, é imperativo empregar um eficiente mecanismo de processamento de texto. Isso implica na necessidade de empregar métodos que vão além da mera compreensão das palavras, incluindo a análise de estruturas sintáticas, contextos e relações semânticas, dessa maneira, extraindo informações valiosas sobre as emoções expressas. Nas seções subsequentes, serão detalhadamente discutidos o processamento de texto e as etapas cruciais que fundamentam uma análise de qualidade, isto é, métricas para enxugar a base de dados e utilizar o que realmente é necessário para a realização desse processo.

3.6 PROCESSAMENTO DE TEXTOS

O processamento de textos desempenha um papel fundamental na análise de sentimentos expressos em opiniões. Para que os algoritmos de classificação possam realizar essa tarefa, é necessário que os textos estejam em um formato compreensível pelos modelos matemáticos gerados (Bellini, 2019). Este processo abrange uma variedade de técnicas e etapas, as quais serão discutidas em detalhes a seguir.

3.6.1 Pré-processamento

Segundo (Bellini, 2019 *apud* Uysal; Gunal, 2014) o pré-processamento dos textos desempenha um papel essencial não apenas na conversão dos mesmos em formatos que os algoritmos de aprendizado de máquina possam interpretar, mas também na otimização do desempenho desses algoritmos. Isso é alcançado por meio da remoção de informações irrelevantes para a tarefa de classificação textual, o que resulta em um processo mais eficiente e preciso de análise de textos. Assim, é comum aplicar técnicas de pré-processamento para garantir a qualidade e a consistência dos dados textuais, além de otimizar a eficiência dos algoritmos de aprendizado de máquina utilizados na tarefa de classificação textual. Algumas das técnicas mais utilizadas serão apresentadas nas próximas seções. A fim de ilustrar cada técnica, utilizaremos como exemplo o poema “Quadrilha” de Carlos Drummond de Andrade. Abaixo está o poema em questão.

> João amava Teresa que amava Raimundo
> que amava Maria que amava Joaquim que amava Lili,
> que não amava ninguém.
> João foi para os Estados Unidos, Teresa para o convento,
> Raimundo morreu de desastre, Maria ficou para tia,
> Joaquim suicidou-se e Lili casou com J. Pinto Fernandes
> que não tinha entrado na história.

3.6.1.1 Conversão para *lower case* (caixa baixa) e remoção de pontuações

Estas são duas etapas comuns que são muito importantes para o pré-processamento do texto. Em textos nos quais uma mesma palavra é repetida, a transformação para letras

minúsculas (*lower case*) impede que diferentes variações dessa palavra sejam consideradas como palavras diferentes. Quanto às pontuações, elas não possuem relevância na análise de sentimentos e, portanto, devem ser eliminadas (Bellini, 2019). Abaixo está o poema "Quadrilhas" sem pontuação e em *lower case*.

> joão amava teresa que amava raimundo
> que amava maria que amava joaquim que amava lili
> que não amava ninguém
> joão foi para os estados unidos teresa para o convento
> raimundo morreu de desastre maria ficou para tia
> joaquim suicidou-se e lili casou com j pinto fernandes
> que não tinha entrado na história

3.6.1.2 Remoção de *stop words*

Existe algumas palavras que não agregam valor significativo a uma frase, ou seja, são irrelevantes e não tem significado para o processamento de linguagem natural. Essas palavras são conhecidas como *stop words* (Dias, 2021). Alguns exemplos são as palavras: "a", "em", "os", "dos", "um", "nos", entre várias outras. A seguir o poema "Quadrilha" exemplifica a remoção das *stop words*.

> João amava Teresa amava Raimundo
> amava Maria amava Joaquim amava Lili,
> não amava ninguém.
> João Estados Unidos, Teresa convento,
> Raimundo morreu desastre, Maria ficou tia,
> Joaquim suicidou-se Lili casou com J. Pinto Fernandes
> não tinha entrado história.

3.6.1.3 Lemmatization

A *Lemmatization*, também conhecida como Lematização, é um processo que consiste em remover ou substituir o sufixo de uma palavra para transformá-la em sua forma base, conhecida como lema. Esse método é utilizado para reduzir as palavras à sua forma

fundamental, o que facilita a análise e a comparação entre diferentes formas de uma mesma palavra (Dias, 2021).

Uma limitação da lematização ocorre quando lidamos com palavras homógrafas, ou seja, palavras com a mesma grafia, mas significados diferentes. Nesses casos, pode haver desafios adicionais relacionados à ambiguidade e à correta atribuição do lema correspondente. Abaixo o poema “Quadrilhas” exemplifica um texto lematizado.

> João amar Teresa que amar Raimundo
> que amar Maria que amar Joaquim que amar Lili,
> que não amar ninguém.
> João ir para o Estados Unidos, Teresa para o convento,
> Raimundo morrer de desastre, Maria ficar para tia,
> Joaquim suicidar-se e Lili casar com J. Pinto Fernandes
> que não ter entrar na história.

3.6.1.4 Stemming

Como uma alternativa à lematização, a técnica de *stemming*, conhecida como stemização em português, consiste no truncamento de palavras que tem como objetivo reduzi-las à sua raiz básica. Os sufixos são removidos para alcançar essa forma raiz, mantendo o significado semântico das diferentes formas inalterado. No entanto, essa abordagem nem sempre produz resultados ideais, já que as palavras perdem parte de seu significado original. Isso ocorre porque o stemming opera nas palavras sem levar em consideração o contexto em que estão inseridas (Dias, 2021). Abaixo, o poema “Quadrilhas” exemplifica um texto com stemização.

> joão am teres que am raimund
> que am mari que am joaquim que am lil,
> que não am ninguém.
> joão foi par os estad unid, teres par o convento,
> raimund morr de desastr, mari fic par tia,
> joaquim suicid e lil cas com j. pinto fernand
> que não tin entr na história.

3.6.1.5 N-grams

N-gramas são sequências contínuas de n palavras em um texto. Os n-gramas são usados para capturar informações sobre a estrutura e a frequência das palavras ou caracteres em um texto. São separados em três classes: unigrama, onde temos a sequência de apenas uma palavra, como "Eu"; bigrama, onde temos uma sequência de duas palavras, como "por favor, vire" ou "vire o seu"; e trigrama, onde temos uma sequência de três palavras, como "por favor, vire o seu" ou "faça o seu dever de casa". A utilização desses elementos pode melhorar ou não os resultados obtidos dependendo de sua aplicação. Sendo assim, podem ser amplamente aplicados em diversas tarefas de processamento de linguagem natural (Souza, 2021 *apud* Jurafsky; Martin, 2006).

3.6.1.6 Tokenização

A tokenização de frases, também conhecida como segmentação de frase, é o procedimento de dividir um texto em unidades de frases individuais, ou seja, a tokenização é o processo de dividir um texto em unidades menores, chamadas de tokens (Dias, 2021). Um exemplo dessa ferramenta pode ser visto na frase: “O teste de Turing, Alan Turing, é baseado em um jogo de salão, comum dos anos 50!”. Dessa forma, realizando a tokenização observamos que a frase foi dividida em vários tokens individuais.

> ["O", "teste", "de", "Turing", ",", "Alan", "Turing", ",", "é", "baseado", "em", "um", "jogo", "de", "salão", ",", "comum", "dos", "anos", "50", "!"]

3.7 VETORIZAÇÃO DE TEXTOS

A vetorização de textos é um processo fundamental na área de processamento de linguagem natural (PLN) que permite representar documentos de texto como vetores numéricos. Essa representação é necessária porque os algoritmos de aprendizado de máquina normalmente trabalham com dados numéricos.

Assim, a fim de aplicar técnicas de aprendizado de máquina em textos, é necessário converter os documentos em representações vetoriais. Esse processo é uma etapa crucial que permite a utilização de algoritmos de aprendizado de máquina numéricos. Ele constitui o primeiro passo essencial para a análise sensível ao idioma (Martins *et al*., 2020 *apud* Bengfort;

Bilbro; Ojeda, 2018). Nas próximas seções serão abordadas algumas das técnicas mais comuns para a vetorização de textos.

3.7.1 Bag-of-Words

O método *Bag-of-Words* é amplamente empregado para representar documentos em formato de texto ou imagens. No contexto de documentos de texto, essa abordagem consiste em transformá-los em vetores, onde cada posição do vetor representa o número de vezes que uma determinada palavra aparece ao longo do conteúdo do documento, ou seja, sua frequência absoluta (Belline, 2019 *apud* Boulis; Ostendorf, 2005). A Tabela 1 ilustra um exemplo de representação *bag-of-words* de algumas palavras do poema "Quadrilhas" de Carlos Drummond de Andrade. Nela é possível perceber a frequência das palavras no poema.

Tabela 1 – Exemplo de *Bag-of-words*.

Estados	João	Maria	na	não	para	que	tinha
1	2	2	1	2	3	6	1

Fonte: Autoria própria (2023).

Como podemos observar na Tabela 1, existem palavras com uma frequência maior que outras, por exemplo, a palavra "que" aparece seis vezes no texto, enquanto a palavra "Estados" aparece uma vez no texto. Assim, essa abordagem trata a importância de cada palavra de acordo com sua frequência no texto.

3.7.2 *Term Frequency-Inverse Document Frequency (TF-IDF)*

Conforme mencionado anteriormente, no método *Bag-of-Words*, os textos são representados usando a medida de Frequência do Termo (TF). No entanto, o vetor resultante pode ser normalizado ou escalonado de várias maneiras diferentes (Belline, 2019). Um exemplo disso é o *Term Frequency-Inverse Document Frequency* (TF-IDF), que leva em consideração não apenas a frequência do termo em um documento, mas também a sua importância relativa em todo o *corpus* de documentos.

Ao utilizar a medida TF-IDF, para cada termo é atribuído uma frequência relativa que é calculada como uma proporção inversa entre sua frequência absoluta em um determinado texto e a porcentagem de documentos em que esse termo aparece (Belline, 2019 *apud* Ramos, 2003).

De acordo com (Belline, 2019 *apud* Salton; Buckley, 1988) a frequência do termo, conhecida como *Term Frequency* em inglês, é calculada como a razão simples entre o número de ocorrências desse termo e o total de palavras presentes no texto em análise. A Equação 1 mostra isso.

$$tf = \frac{A}{a} \tag{1}$$

Onde:

- A = número de ocorrências em que o termo t aparece em uma frase;
- a = total de palavras na frase

Segundo (Belline, 2019 *apud* Silge; Robinson, 2017) a medida de *Inverse Document Frequency* (IDF) reduz o peso de palavras comuns, como stop words, que foram mencionadas anteriormente, e aumenta o peso de palavras menos frequentes em um conjunto de textos. A Equação 2 mostra como é realizado esse cálculo.

$$i\,df = \ln\left(\frac{N}{n}\right) \tag{2}$$

Onde:

- N = total de textos contidos no conjunto de dados;
- n = número de textos na base de dados que contêm o termo analisado

Dessa forma, com esses dois parâmetros calculados, a medida TF-IDF pode ser calculada. A Eq. 3 representa a fórmula matemática utilizada para o cálculo do TF-IDF.

$$tf * idf \; = \; tf \cdot idf \tag{3}$$

Como exemplos, vamos supor três frases diferentes contendo alguns termos semelhantes para calcular o TF-IDF.

Frase1: "O céu é azul."

Frase2: "O céu é nublado."

Frase3: "O sol é brilhante."

Primeiramente vamos calcular a frequência do termo (TF) para cada palavra em cada frase. Assim, usando a Eq. 1, temos;

$$tf = \frac{A}{a}$$

O parâmetro "A" corresponde ao número de ocorrências em que o termo t aparece em uma frase e o parâmetro "a" corresponde ao total de palavras na frase. Assim;

Para a frase1:

Palavra “O”, $A = 1$ e $a = 4$, pois a frase contém quatro palavras.

$$tf = \frac{1}{4} = 0{,}25$$

Palavra “céu”, $A = 1$ e $a = 4$, pois a frase contém quatro palavras.

$$tf = \frac{1}{4} = 0{,}25$$

De maneira semelhante, o cálculo do valor de tf para as palavras "é" e "azul" é realizado de maneira idêntica, resultando em um valor de 0,25 para ambas, uma vez que elas aparecem apenas uma vez na frase, que possui um total de quatro palavras.

Para a frase2:

Palavra “O”, $A = 1$ e $a = 4$, pois a frase contém quatro palavras.

$$tf = \frac{1}{4} = 0{,}25$$

Palavra “céu”, $A = 1$ e $a = 4$, pois a frase contém quatro palavras.

$$tf = \frac{1}{4} = 0{,}25$$

De forma análoga, para as palavras “é” e “nublado” o valor de *tf* é calculado da mesma forma e o resultado correspondente é igual 0,25, pois elas aparecem uma vez na frase, e a frase contém quatro palavras.

Para a frase3:

Palavra “O”, $A = 1$ e $a = 4$, pois a frase contém quatro palavras.

$$tf = \frac{1}{4} = 0{,}25$$

Palavra “sol”, $A = 1$ e $a = 4$, pois a frase contém quatro palavras.

$$tf = \frac{1}{4} = 0{,}25$$

Para as palavras “é” e “brilhante”, o cálculo do *tf* é realizado de forma idêntica e o valor correspondente de ambas é 0,25, visto que a frase contém quatro palavras e nenhuma delas são repetidas.

Com os valores da frequência do termo calculados, agora procederemos ao cálculo da Frequência Inversa do Documento (IDF) para cada palavra. Para isso, utilizaremos a Eq. 2, que é a seguinte:

$$i\,df = \ln\left(\frac{N}{n}\right)$$

Para este exemplo, $N = 3$, pois são três frases, ou seja, o total de textos contidos no conjunto de dados. E n é o número de frase em que um determinado termo t aparece. Logo;

Para a palavra "O", $n = 3$, pois a palavra aparece nas três frases

$$i\,df = \ln\left(\frac{3}{3}\right) = 0$$

Para a palavra "céu", $n = 2$, pois a palavra aparece em duas frases

$$i\,df = \ln\left(\frac{3}{2}\right) = 0{,}4055$$

Para a palavra "é", $n = 3$, pois a palavra aparece nas três frases

$$i\,df = \ln\left(\frac{3}{3}\right) = 0$$

Para a palavra "azul", $n = 1$, pois a palavra aparece em uma frase

$$i\,df = \ln\left(\frac{3}{1}\right) = 1{,}0986$$

De modo similar, os cálculos de $i\,df$ para as palavras "nublado", "sol" e "brilhante" seguiram o mesmo padrão, onde *N* representa as três frases no documento, isto é, *N* = 3 e *n* foi definido como 1, uma vez que essas palavras aparecem somente em uma das frases.

Dessa forma, com os valores da frequência dos termos e com os valores dos inversos da frequência dos termos, estamos aptos a calcular o TF-IDF para cada palavra em cada frase. Para isso, aplicaremos a Eq. 3, a qual é representada por:

$$tf * idf \;=\; tf \cdot idf$$

Para a frase1:

Palavra “O”

$$tf * idf \;=\; 0{,}25 \cdot 0 = 0$$

Palavra “céu”

$$tf * idf \;=\; 0{,}25 \cdot 0{,}4055 = 0{,}101375$$

Palavra “é”

$$tf * idf \;=\; 0{,}25 \cdot 0 = 0$$

Palavra “azul”

$$tf * idf \;=\; 0{,}25 \cdot 1{,}0986 = 0{,}27465$$

Da mesma maneira, os valores de TF-IDF para as demais frases foram calculados seguindo o mesmo procedimento. A Tabela 2 apresenta os valores correspondentes a cada frase, resumindo o resultado desse processo.

Tabela 2 – Valores de TF-IDF.

Frase1	TF-IDF	Frase2	TF-IDF	Frase3	TF-IDF
O	0	O	0	O	0
céu	0,101375	céu	0,101375	sol	0,27465
é	0	é	0	é	0
azul	0,27465	nublado	0,27465	Brilhante	0,27465

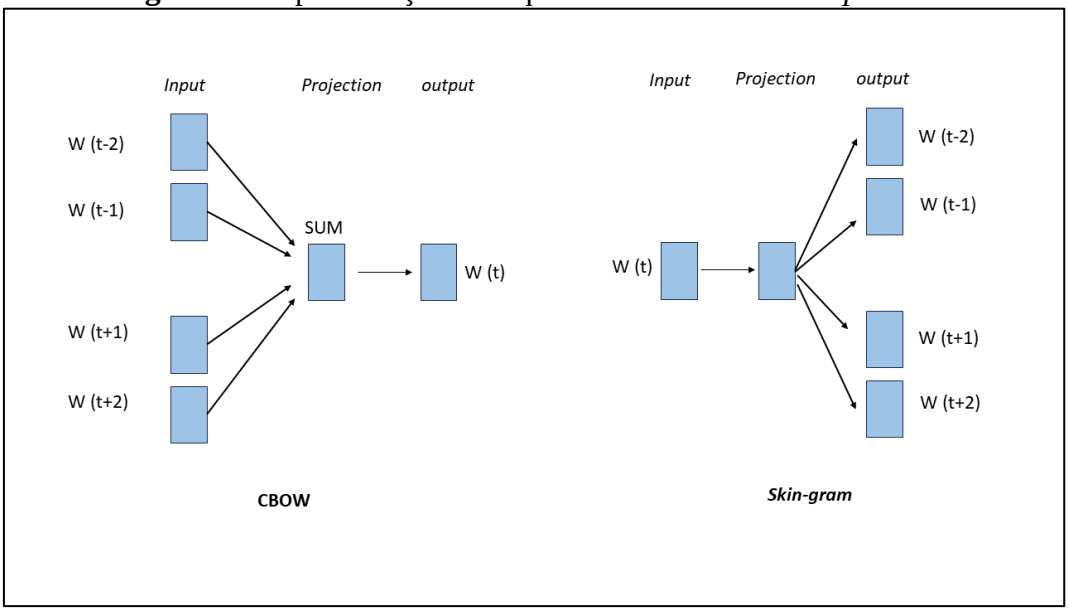
Fonte: Autoria própria (2023).

3.7.3 Word2vec

Outra ferramenta usada para calcular representações de palavras como vetores é o *Word2vec*. Estudada e introduzida por (Mikolov *et al*., 2013), tem a capacidade de capturar as relações semânticas entre palavras com base em seus contextos em um grande *corpus* de texto. Seu funcionamento ocorre tomando um *corpus* como *input* e devolvendo vetores de palavras como *output*. Segundo (Aguiar, 2016, p. 5) “no processo é construído um vocabulário, e ocorre o aprendizado das representações do vocabulário como vetores”. São feito uso de duas arquiteturas: *continuous bag-of-words* (CBOW) e *skip-gram*. Na prática, o modelo *skip-gram* fornece uma representação mais eficaz das palavras em conjuntos de dados menores, enquanto o CBOW se destaca por sua velocidade e aplicabilidade em bases de dados extensas (Aguiar, 2016).

No modelo CBOW, o *input* é um contexto, e o *output* é uma palavra que foi omitida. Para isso, são combinadas as representações das palavras circundantes a fim de prever a palavra central. O contexto é definido pelas palavras que estão em proximidade da palavra-alvo (Aguiar, 2016). No modelo *skip-gram*, o objetivo é prever o contexto de uma palavra dada. Nesse caso, o *input* consiste em uma única palavra que passa por uma camada oculta, e o resultado desejado é identificar o contexto mais provável para essa palavra (Aguiar, 2016). A Figura 7 ilustra um modelo simplificado de como funciona os dois tipos de arquiteturas.

Figura 7 - Representação da arquitetura do CBOW e *Skip-Gram*.



Fonte: Aguiar (2016), adaptado.

No CBOW, a tarefa consiste em prever uma palavra com base no contexto, enquanto no *skip-gram*, a tarefa é prever as palavras que cercam uma palavra específica (mais bem visualizado na Figura 7).

3.8 CLASSIFICADORES

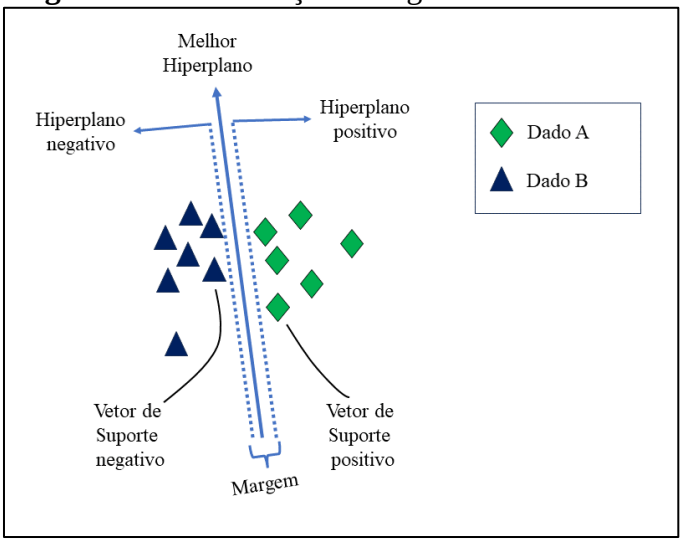
Nessa seção, vamos explorar os classificadores utilizados no trabalho, fornecendo uma visão aprofundada sobre seus conceitos, como ocorre o funcionamento de cada um e suas aplicações.

3.8.1 Linear Support Vector Classification (LinearSVC)

O *Linear Support Vector Classification* é uma das versões do *Support Vector Machines* (SVM). Trata-se de um algoritmo de aprendizado de máquina usado para realizar tarefas de classificação em conjuntos de dados. Pode ser muito utilizado para a classificação binária, onde os dados são divididos em duas classes distintas. O algoritmo *LinearSVC* é um algoritmo de SVM linear. Assim, a proposta desse algoritmo consiste em representar os dados em um plano de n dimensões, em que n corresponde ao número de características do problema, e o valor de cada característica é utilizado como a coordenada espacial de um ponto. Seu objetivo é encontrar um hiperplano que melhor se adequa ao problema, ou seja, que melhor diferencia as

classes do problema (Souza, 2021). Pode ser utilizado em várias aplicações como, categorização de textos, análise de imagens entre outros. A Figura 8 ilustra um exemplo básico desse algoritmo, mostrando a melhor fronteira que separa diferentes categorias de dados de maneira mais precisa.

Figura 8 – Classificação do algoritmo *LinearSVC*.



Fonte: Oliveira (2023), adaptado.

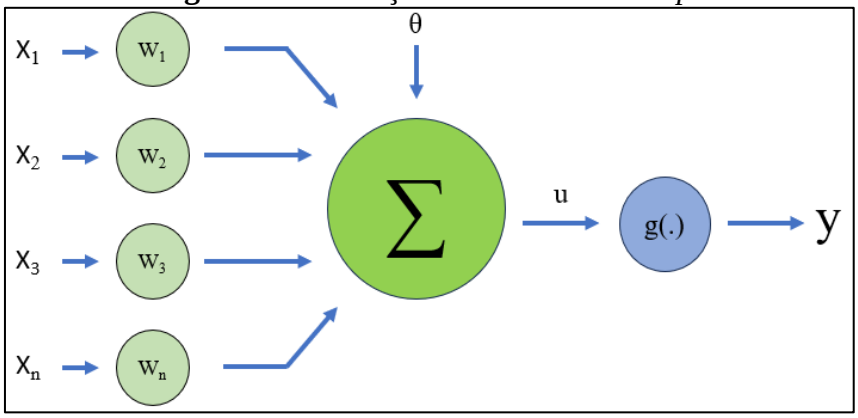
Esse exemplo ilustrado na Figura 8, retrata a classificação de dois grupos. Podemos observar que a reta (melhor hiperplano) está fazendo a separação do conjunto de dados utilizando duas retas auxiliarem pontilhadas (hiperplano positivo e negativo). A margem, uma dimensão fundamental nesse processo, é caracterizada pela distância existente entre esses dois hiperplanos pontilhados, cujo ajuste direciona a formação dos chamados vetores de suporte. Esses vetores de suporte, por sua vez, representam os primeiros pontos de dados do conjunto a serem interceptados pelas hiperplanos.

3.8.2 Perceptron

Segundo (Silva; Spatti; Flauzino, 2010), desenvolvido por Rosenblatt, o *Perceptron* representa uma das configurações mais básicas de uma rede neural artificial. Seu propósito era criar um modelo computacional inspirado na retina, com a finalidade de desenvolver um elemento eletrônico de percepção de sinais. Seu princípio de funcionamento é extremamente simples consistindo basicamente em entradas, pesos sinápticos, função de ativação e saída. Assim, a rede recebe os dados de entrada e os processa. Por meio dos pesos sinápticos a rede é capacitada a classificar ou separar os dados de acordo com o problema em questão. O

Perceptron emprega o método de treinamento supervisionado e faz uso da função de ativação em degrau, ou seja, opera de maneira binária. A Figura 9 exemplifica uma rede *Perceptron* simples, composta por apenas um único neurônio e constituídas de n entradas e somente uma saída.

Figura 9 – Ilustração de uma rede *Perceptron*.



Fonte: Silva; Spatti; Flauzino (2010), adaptado.

Seu princípio de funcionamento é bastante simples: inicialmente, os valores de entrada x_1, x_2, ..., x_n, que representam informações sobre o seu comportamento, serão multiplicados pelos pesos sinápticos correspondentes W_1, W_2, ..., W_n, com o propósito de qualificar a importância de cada um frente aos objetivos funcionais atribuídos ao neurônio. Em seguida, o valor resultante da composição de todas as entradas devidamente ponderadas pelos seus devidos pesos é diminuído de um determinado limiar de ativação (θ), em seguida, esse valor é direcionado para a função de ativação g(u), cuja função é restringir a saída do neurônio dentro de um intervalo específico de valores. O resultado será a saída (y) produzida pelo *Perceptron* (exemplificado na Figura 9) (Silva; Spatti; Flauzino, 2010). Em termos matemáticos, o processamento interno realizado pelo *Perceptron* pode ser descrito pelas seguintes Equações:

$$u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \qquad (4)$$

Em que:

w_i = é o peso sináptico *n* correspondente a entrada *i*;

x_i = é a entrada *n*-ésima do *Perceptron*;

θ = é o limiar de ativação;

$$y = g(u) \quad (5)$$

Em que:

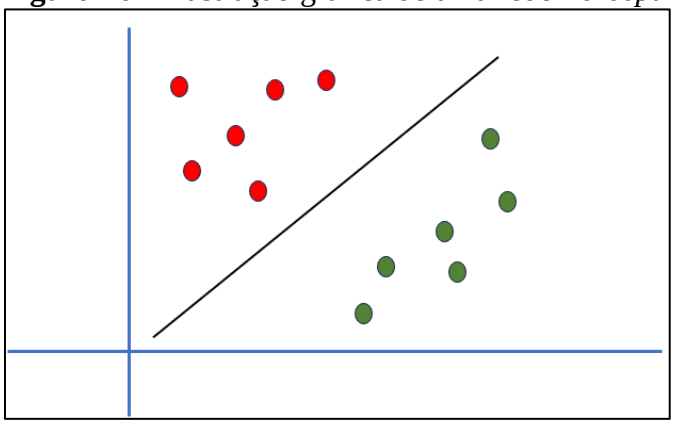
$u =$ é a saída do neurônio;

$g =$ é a função de ativação;

$y =$ é a saída geral da rede;

A Figura 10 exemplifica de maneira visual como um hiperplano pode ser utilizado por uma rede *Perceptron* para efetuar a separação de dois conjuntos de amostras linearmente separáveis.

Figura 10 – Ilustração gráfica de uma rede *Perceptron*.



Fonte: Autoria própria (2023).

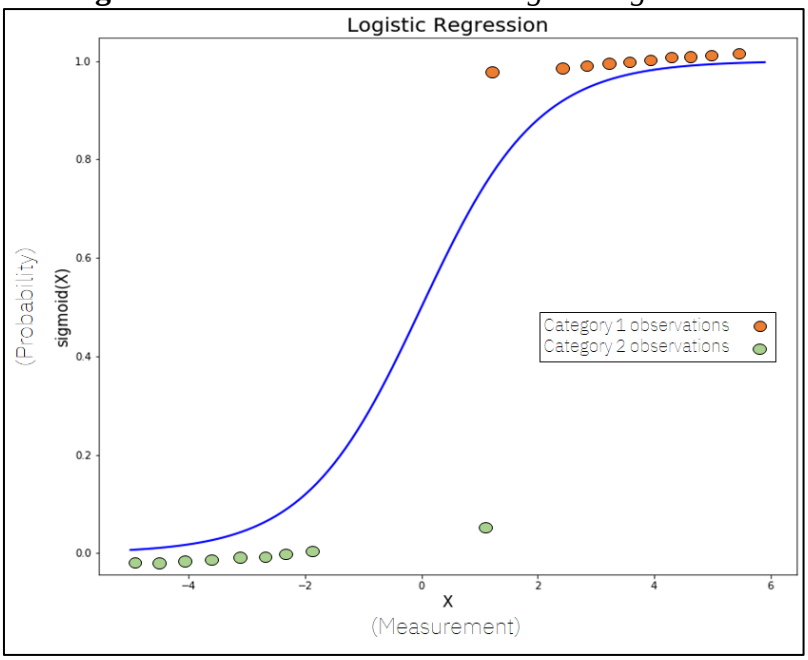
Como podemos observar pela Figura 10, nesse contexto, o hiperplano atua como uma fronteira decisória que divide os dois grupos de amostras de forma que todas as amostras de um grupo estejam de um lado do hiperplano, e todas as amostras do outro grupo estejam do lado oposto.

3.8.3 Logistic regression

Logistic regression conhecido em português como regressão logística é um método de classificação amplamente utilizado que é simples de implementar e mostra um bom desempenho em classes que podem ser separadas linearmente. Pode ser usada em situações em que a variável de resposta é qualitativa, como no caso de um dispositivo obter “sucesso” ou “falha”. Desse modo, torna-se muito eficaz para lidar com problemas de classificação. Assim, a regressão logística assemelha-se com o *Perceptron*, um modelo linear que pode ser utilizado

para a classificação binária (Raschaka; Mirjalili, 2019, tradução minha). De acordo com (Gonzalez, 2018) “a regressão logística trata-se de uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária”. Contudo, para o contexto de aprendizado de máquina, a regressão logística pode ser utilizada em problemas de classificação. Existem três tipos específicos de regressão Logística: regressão logística binária, regressão logística multinominal e regressão logística ordinal. Para este estudo, optaremos pela aplicação da regressão logística binária, uma vez que a variável de interesse é categorizada em duas classes distintas: positiva e negativa. A Figura 11 ilustra como ocorre o processo probabilístico utilizado pelo modelo de regressão logística.

Figura 11 – Curva característica *LogisticRegression*



Fonte: Espinho (2020)

Como podemos observar pela Figura 11, a regressão logística utiliza um tipo de função para determinar a probabilidade da classe de um dado. Essa função probabilística utilizada é a sigmoide. Desse modo, na hora de classificar as classes a função sigmoide determina a qual classe pertence os dados. Valores próximos de 0 indicam uma classificação na região negativa, enquanto valores próximos de 1 indicam uma classificação na região positiva, com 0,5 servindo como ponto de corte entre essas duas categorias. Logo, quando um dado se aproxima de 0, a função classificará como pertencente à categoria negativa, e quando se aproxima de 1, será classificado como pertencente à categoria positiva.

3.9 TECNOLOGIAS UTILIZADAS

A principal tecnologia utilizada para o desenvolvimento do trabalho foi a linguagem de programação Python, que apresenta uma fácil compreensão com relação às outras linguagens de programação, sobretudo pela sua sintaxe clara e intuitiva. Dessa forma, sendo possível implementar vários algoritmos e soluções de maneira eficiente e refinada. Além disso, o Python possui amplas bibliotecas padrões que oferecem uma imensa gama de funcionalidades prontas para uso, o que acelera o processo de desenvolvimento. Algumas das bibliotecas padrão utilizadas para o desenvolvimento do trabalho serão abordadas nas próximas seções.

3.9.1 Python

Desenvolvida em 1991, Python é uma linguagem de programação simples e intuitiva sendo caracterizada muitas vezes como genérica, porém com um grande poder computacional, pois pode ser utilizada em várias situações. Isso é possível devido a sua vasta rede de bibliotecas padrão que permitem o desenvolvimento de diversas aplicações, dentre elas o desenvolvimento *web*, programação matemática e computacional, e é amplamente adotada para automação de tarefas, análise de dados, inteligência artificial e muito mais (Kremer, 2023). Dessa forma, sendo utilizada por grandes empresas como Google e a NASA.

3.9.2 Pandas

O pandas é uma ferramenta muito importante para análise de dados. Se trata de uma biblioteca-padrão do Python que permite a realização de diversas tarefas relacionadas à manipulação, limpeza, transformação e análise de dados de forma eficiente e poderosa, oferecendo estruturas de dados de alto nível e funções que permitem agilizar, simplificar e tornar mais expressiva a tarefa de trabalhar com conjuntos de dados estruturados ou em formato de Tabela (Mckinney, 2018). No entanto, o que ela oferece de mais importante são os *Dataframes*, ou seja, estruturas bidimensionais com dados rotulados. Assim, pode-se dizer que se trata de uma planilha em que cada linha representa um dado e cada coluna um atributo (Kremer, 2023).

3.9.3 Scikit-learn

O *Scikit-learn* se trata de um Kit de ferramentas muito importante para o aprendizado de máquina na linguagem de programação Python (Mckinney, 2018). Disponibilizando diversos métodos para criar, treinar e avaliar modelos de aprendizado de máquina (Kremer, 2023). Sendo assim pode ser utilizado em vários contextos. Segundo (Santos, 2015) atualmente, é tida como uma das bibliotecas mais destacadas e eficazes para fins de mineração de dados e análise de dados, oferecendo um amplo conjunto de algoritmos de aprendizado de máquina supervisionados e não-supervisionado para as mais diversas finalidades, a partir de uma interface consistente e de fácil usabilidade. Suas bases são fundamentadas em um conjunto de outras bibliotecas, tais como:

- *Numpy* – Pacote para manipulação de arrays N-dimensional;
- *SciPy* – Pacote para computação científica;
- *Matplotlib* – Pacote para terminal interativo de comandos.

Além de fornecer suporte sólido para ambientes de produção, a biblioteca *Scikit-learn* também prioriza a usabilidade, clareza de código, colaboração, extensa documentação e um desempenho excepcional (Santos, 2015). Algumas funcionalidades concedida pelo *Scikit-learn* são:

- ***Clustering*** - Algoritmos não-supervisionados para agrupamento de dados não categorizados, como K-means;
- ***Cross Validation*** - Técnica para validação de modelos preditivos de algoritmos supervisionados contra dados não vistos;
- ***Feature extraction*** - Para definição de atributos em dados não estruturados como imagem e texto;
- ***Ensemble methods*** - Para combinação das predições de diversos modelos.

Logo, essa é uma das bibliotecas que se destaca no aprendizado de máquina, pois possui muitas ferramentas que são uteis na resolução de problemas.

3.9.4 NLTK

O *Natural Language Toolkit* conhecido também como NLTK representa uma ferramenta muito importante para o campo do processamento de linguagem natural (PLN). Ela

possibilita a criação de programas em Python que são capazes de executar uma variedade de tarefas, como tokenização de palavras, similaridade entre dois textos, entre outras finalidades relacionadas ao processamento de texto (Martins *et al.*, 2020). O NLTK oferece uma documentação abrangente, com detalhes completos sobre a API de cada módulo, classes e funções do conjunto de ferramentas, disponíveis no website http://www.nltk.org/. Essa documentação não apenas especifica os parâmetros essenciais, mas também apresenta exemplos práticos de como utilizar esses recursos (Santos, 2015).

3.10 MÉTRICAS DE AVALIAÇÃO E PERFORMACE

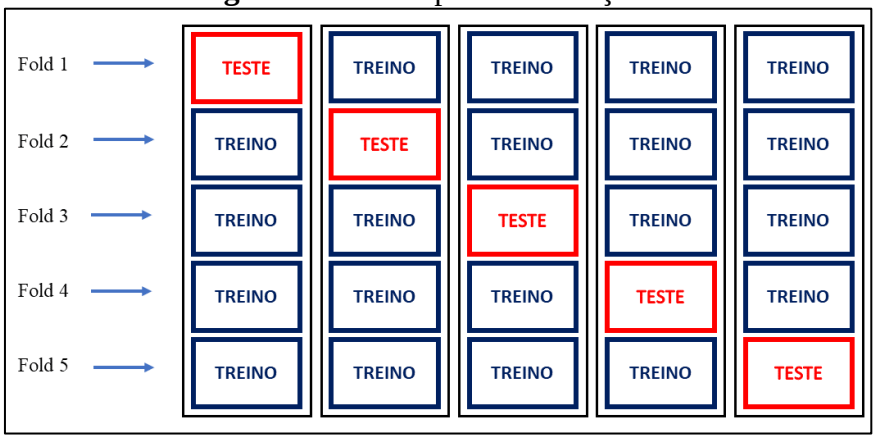
Avaliar métricas é essencial para a análise de qualquer método de classificação proposto aplicado a um conjunto de dados, desempenhando um papel crucial na determinação da qualidade do modelo selecionado. Habitualmente não existe técnica melhor que a outra, isto é, o que deve ser levado em consideração é a escolha correta de cada técnica para a resolução do problema trabalhado (Tiburcio, 2021 *apud* Carvalho *et al*., 2011).

Assim, para que seja possível compreender as métricas de avaliação é importante abordar alguns tópicos sobre técnicas de avaliação de performance dos algoritmos de classificação. Neste contexto, destaca-se a Validação Cruzada como a abordagem principal adotada neste estudo. E em seguida destaca-se a Acurácia como métrica de avaliação.

3.10.1 K-Fold cross-validation

A validação cruzada, também conhecida como *cross-validation* é uma técnica utilizada na área de aprendizado de máquina para avaliar a performance e a capacidade de generalização de um modelo em dados não vistos anteriormente. Dessa forma, ela evita o *Overfitting* e aumenta a confiabilidade das métricas do modelo gerado. Existem diferentes métodos de validação cruzada disponíveis, como o *K-Folds*, *hold-out*, *leave-one-out* entre outros (Kremer, 2023). Neste trabalho, o método utilizado será o *K-Folds*, onde os dados de treino são divididos em K partes. A Figura 12 ilustra como ocorre a validação cruzada com a divisão dos dados de treino em 5 *folds*.

Figura 12 – Exemplo de validação cruzada.



Fonte: Autoria própria (2023).

Como podemos observar na Figura 12, a validação cruzada ocorre com o método *K-Folds*, onde os dados de treino são divididos especificamente em 5 *folds* distintos. O procedimento ocorre com a divisão dos dados em 5 subconjuntos igualmente dimensionados, permitindo que cada *fold* seja usado uma vez como conjunto de validação (teste) e os demais como conjunto de treinamento durante o processo de avaliação. Assim, essa abordagem elimina a dependência de uma única divisão entre treinamento e validação. Com isso, o resultado do processo é representado por a média das *K* interações, ou seja, para esse exemplo o resultado será a soma de cada *K* interação dividido por 5, já que o conjunto foi divido em 5 *folds*. Deste modo, esse procedimento permite que todos os dados sejam usados tanto na parte de treino como na parte de validação. Logo, através da repetição do procedimento *K-Folds*, onde cada *fold* é utilizado como conjunto de validação, obtém-se uma visão abrangente do desempenho do modelo, considerando diferentes combinações de dados de treinamento e validação. E assim, dificultar o *Overfitting* garantindo uma boa análise nos modelos gerados.

3.10.2 Acurácia

A acurácia é uma métrica comum usada para avaliar o desempenho de modelos de classificação. Ela representa a proporção de predições corretas feitas pelo modelo em relação ao número total de predições (Oliveira, 2023). Em outras palavras, a acurácia mede a precisão geral do modelo em classificar corretamente os exemplos. A Equação 6 demostra como essa métrica é calculada.

$$Acurácia = \frac{VP}{VN} \tag{6}$$

Onde:

VP = número de previsões corretas e;

VN = número total de previsões.

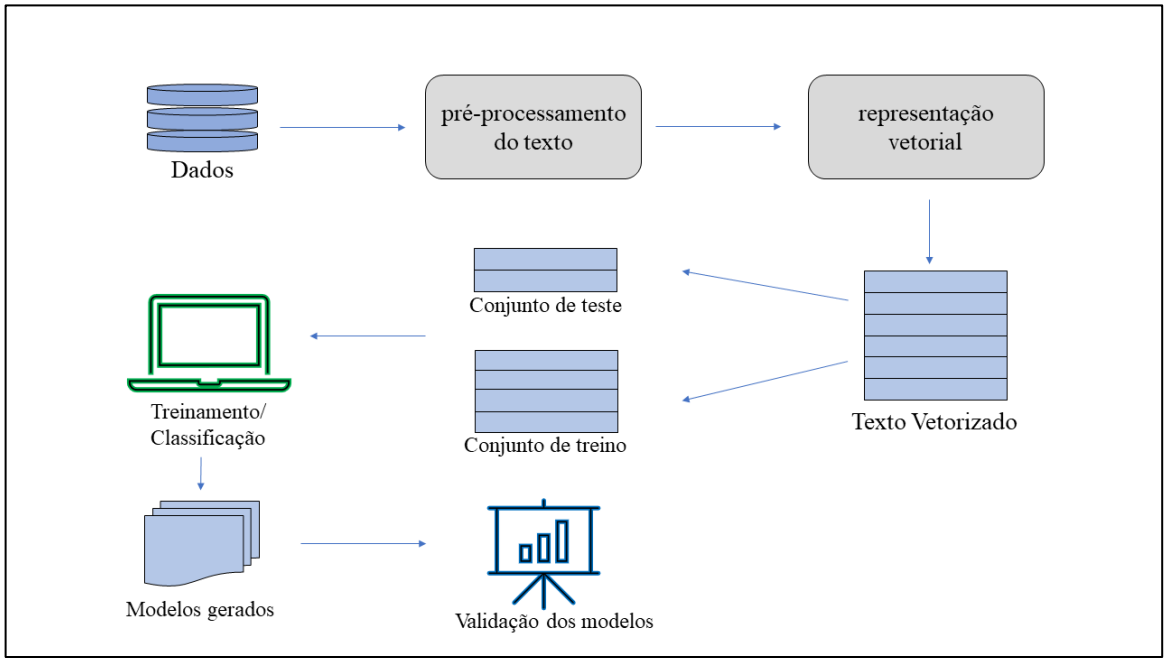
4 METODOLOGIA

A metodologia empregada neste trabalho, inicialmente, consiste em uma pesquisa bibliográfica para encontrar as diversas tecnologias e métodos utilizados para processamento de Linguagem Natural (PLN) e com isso, aplicar na análise de sentimentos com auxílio do aprendizado de máquina. Em seguida, apresenta-se a relação de ferramentas de desenvolvimento relacionada à área de aprendizagem de máquina (AM) e suas respectivas bibliotecas ou extensões, com o objetivo de classificação de textos. A seguir serão descritos todos os métodos, técnicas e ferramentas computacionais que foram utilizadas no trabalho seguindo um percurso metodológico.

4.1 PERCURSO METODOLÓGICO

Este trabalho segue algumas etapas metodológicas. A Figura 13 ilustra o percurso metodológico feito e as próximas seções detalham cada uma delas para uma melhor compreensão do processo de pesquisa.

Figura 13 – Percurso metodológico.



Fonte: Belline (2019), adaptado.

Como podemos observar pela Figura 13, o trabalho segue algumas etapas que são fundamentais para entender como ocorre todo o processo de aprendizado de máquina e que não

podem ser desconsideras, visto que isso resultaria em um resultado insatisfatório. O primeiro passo é entender o problema, e então obter a base de dados, que passa pelo pré-processamento, limpeza e é convertida para a forma vetorial, em seguida os dados representados vetorialmente são divididos em um conjunto de treino e um conjunto de testes, logo após isso, os classificadores são utilizados e modelos são gerados. Com isso, é possível verificar a eficácia dos classificadores e averiguar quais são suas vantagens e desvantagens. Às próximas seções detalham os processos descrito no percurso metodológico.

4.1.1 Base de dados e Divisão dos dados

A base de dados utilizada para este trabalho foi um *dataset* sobre avaliações de filmes com aproximadamente 50.000 dados, esse *dataset* é disponibilizado por uma plataforma online chamada *Kaggle*[1]. Essa base de dados consiste em colunas com as avaliações de filmes tanto em inglês quanto em português, além de uma coluna que expressa o sentimento associado a cada avaliação. A Figura 14 ilustra a base de dados em questão com apenas as cinco primeiras linhas e as cinco últimas linhas.

Figura 14 – Base de dados.

	id	text_en	text_pt	sentiment
0	1	Once again Mr. Costner has dragged out a movie...	Mais uma vez, o Sr. Costner arrumou um filme p...	neg
1	2	This is an example of why the majority of acti...	Este é um exemplo do motivo pelo qual a maiori...	neg
2	3	First of all I hate those moronic rappers, who...	Primeiro de tudo eu odeio esses raps imbecis, ...	neg
3	4	Not even the Beatles could write songs everyon...	Nem mesmo os Beatles puderam escrever músicas ...	neg
4	5	Brass pictures movies is not a fitting word fo...	Filmes de fotos de latão não é uma palavra apr...	neg
	id	text_en	text_pt	sentiment
49454	49456	Seeing as the vote average was pretty low, and...	Como a média de votos era muito baixa, e o fat...	pos
49455	49457	The plot had some wretched, unbelievable twist...	O enredo teve algumas reviravoltas infelizes e...	pos
49456	49458	I am amazed at how this movieand most others h...	Estou espantado com a forma como este filme e ...	pos
49457	49459	A Christmas Together actually came before my t...	A Christmas Together realmente veio antes do m...	pos
49458	49460	Working-class romantic drama from director Mar...	O drama romântico da classe trabalhadora do di...	pos

Fonte: Autoria própria (2023).

Com intuito de padronizar a base de dados apenas com dados em português a coluna “text_en” que contia avaliações na língua inglesa e a coluna “id” que contia apenas a numeração

[1] Disponível em: https://www.kaggle.com/datasets/luisfredgs/imdb-ptbr

das frases, foram removidas. Além disso, a coluna “sentiment” contida na base de dados foi transformada para um rótulo binário (label), ou seja, a palavra “pos” (positivo) foi substituída por o número 1, indicando o sentimento positivo, e a palavra “neg” (negativo) foi substituída por o número 0, indicando o sentimento negativo. Essa adaptação se mostrou essencial, uma vez que os sistemas computacionais não possuem a capacidade de compreender a linguagem humana de forma direta. A Figura 15 mostra como ficou a base de dados com as modificações feitas.

Figura 15 – Base de dados modificada.

	text_pt	sentiment	label
0	Mais uma vez, o Sr. Costner arrumou um filme p...	neg	0
1	Este é um exemplo do motivo pelo qual a maiori...	neg	0
2	Primeiro de tudo eu odeio esses raps imbecis, ...	neg	0
3	Nem mesmo os Beatles puderam escrever músicas ...	neg	0
4	Filmes de fotos de latão não é uma palavra apr...	neg	0

	text_pt	sentiment	label
49454	Como a média de votos era muito baixa, e o fat...	pos	1
49455	O enredo teve algumas reviravoltas infelizes e...	pos	1
49456	Estou espantado com a forma como este filme e ...	pos	1
49457	A Christmas Together realmente veio antes do m...	pos	1
49458	O drama romântico da classe trabalhadora do di...	pos	1

Fonte: Autoria própria (2023).

Em seguida, com a base de dados padronizada, os dados de treino e os dados de teste foram separados. Para esse trabalho foram usados 70% dos dados para treino e 30% dos dados para teste. Assim, 35.000 dados foram usados para treino e 15.000 dados para teste. A Figura 16, ilustra como foi realizado a divisão dos dados.

Figura 16 – Divisão dos dados.

```
x_treino = dados.loc[:35000, 'text_pt'].values
y_treino = dados.loc[:35000, 'label'].values
x_teste = dados.loc[35000:, 'text_pt'].values
y_teste = dados.loc[35000:, 'label'].values
```

Fonte: Autoria própria (2023).

Como podemos observar na Figura 16, a base de dados foi dividida em dois pares de grupos: x_treino, y_treino e x_teste, y_teste. Na primeira linha do código a variável x_treino armazena os dados de treino e a expressão “dados.loc[:35000, 'text_pt']” seleciona todas as linhas da coluna ‘text_pt’ da posição 0 até a posição 35.000. Já o método “.values” converte esses dados selecionados em um array Numpy. Na segunda linha é criada uma variável y_treino que armazena os rótulos correspondente aos textos de treinamento, a expressão “dados.loc[:35000, 'label']” seleciona todas as linhas da coluna ‘label’ da posição 0 até a posição 35.000 e o método “.values” converte esses dados em um array Numpy. De maneira análoga, na terceira linha é criada a variável x_teste que armazena os dados do texto de teste. A expressão dados.loc[35000:, 'text_pt'] seleciona as linhas da coluna 'text_pt' da base dados a partir da posição 35.000 até o final. Novamente, o método “.values” converte esses dados selecionados em um array Numpy. Na quarta linha a variável y_teste é criada para armazenar os rótulos correspondente aos textos de teste, selecionando a coluna 'label' e convertendo-a em um array NumPy. Com isso pronto, a etapa subsequente é a limpeza dos dados, isto é, deixar os dados no formato adequado para o aprendizado. Com esse intuito, um bom processamento de texto é fundamental.

4.1.2 Processamento de Texto

Essa etapa envolve o processamento de texto, ou seja, a limpeza do texto para remover quaisquer elementos indesejados, como pontuações, caracteres especiais, acentos e números. Assim, trata-se de uma etapa muito importante para a tarefa de classificação, melhor dizendo, para que a busca por padrões relevantes seja simplificada. Para a realização do processamento de texto foi utilizada algumas técnicas mencionadas na seção 3.6. Dentre as técnicas empregadas, temos;

- Remoção de *stop words*;
- Conversão para caixa baixa (*Lower case*);
- Remoção de acentos.

Para remover as *stop words*, isto é, palavras irrelevantes ou de pouco utilidade presentes nos textos, utilizou-se a biblioteca NLTK, aproveitando seu recurso específico para a língua portuguesa. Essa abordagem foi aplicada devido ao fato de que o *dataset* apresenta dados em

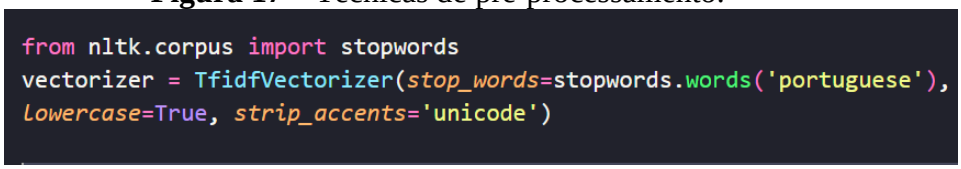
português, tornando a remoção das *stop words* especialmente relevante para o pré-processamento dos textos.

Para converter as palavras do *dataset* para letras minúsculas, utilizou-se o parâmetro *"lower case"*, mencionado na seção 3.6.1.1. Essa normalização é extremamente útil para evitar a duplicação de palavras com letras maiúsculas e minúsculas, o que, por sua vez, aprimora significativamente a qualidade dos resultados obtidos.

Com relação à remoção de acentos, foi utilizado o parâmetro "*strip_accents*". Esse parâmetro especifica que os acentos nas palavras serão removidos durante o processo de vetorização, o que contribui para uma maior coerência dos resultados. Ao remover os acentos, evitamos possíveis variações de palavras que, de outra forma, poderiam ser tratadas como distintas. Esse ajuste melhora a consistência da representação vetorial das palavras, tornando o processo mais eficiente e preciso. Com o processamento de texto finalizado, o próximo passo é vetorizar os dados. Nessa etapa, os elementos textuais são transformados em representações numéricas, permitindo que o modelo de aprendizado de máquina possa compreender e operar de maneira eficaz com as informações obtidas.

Todos esses processos foram implementados de uma única vez, isto é, de maneira simultânea no vetorizador. Para isso, criou-se a variável denominada "vectorizer", que é o vetorizador. Desta forma, os parâmetros para o processamento de texto foram incorporados juntamente com a instância da classe "TfidfVectorizer", que será abordada em detalhes na próxima seção. A Figura 17 ilustra como o código foi implementado utilizando as técnicas discutidas anteriormente.

Figura 17 – Técnicas de pré-processamento.



Fonte: autoria própria, (2023).

Como podemos notar, a Figura 17 ilustra os parâmetros usados na etapa de processamento de texto. O parâmetro *stop_words*=stopwords.words('portuguese') define que as palavras irrelevantes da língua portuguesa presentes no texto serão removidas durante a vetorização. Para isso, foi feita a importação do módulo stopwords da biblioteca NLTK usando o comando "from nltk.corpus import stopwords". O outro parâmetro, *lowercase*=True indica que todas as palavras do texto devem ser convertidas para letras minúsculas antes da

vetorização. Dessa forma, normalizando o texto. E por fim, o parâmetro *strips_accents*=‘unicode’ indica que todos os acentos do texto serão removidos antes da vetorização.

4.1.3 Representação Vetorial

Para a representação vetorial do texto, utilizamos o método TF-IDF, conforme abordado na seção 3.7.2. Esse método envolve o cálculo da frequência de uma palavra específica no texto e, posteriormente, sua ponderação pelo inverso da frequência dessa palavra em todo o *corpus* de documentos. Para implementar essa técnica, criamos um objeto chamado "vectorizer" utilizando a classe "TfidfVectorizer" do pacote *scikit-learn*, mencionado na seção 3.11.3. Essa escolha nos permite transformar os dados textuais em uma matriz numérica que reflete a importância das palavras em relação aos documentos do conjunto de dados. Dessa forma, o processo de vetorização com TF-IDF facilita a análise e o tratamento dessas informações em tarefas como classificação. A Figura 18 descreve o processo de vetorização do texto.

Figura 18 – Processo de vetorização do texto.

```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words=stopwords.words('portuguese'),
lowercase=True, strip_accents='unicode')
train_features = vectorizer.fit_transform(x_treino)
test_features = vectorizer.transform(x_teste)
```

Fonte: Autoria própria (2023).

Conforme mostrada na Figura 18, é ilustrado o procedimento de vetorização do texto. Esse processo foi executado após à etapa de pré-processamento do texto. Para isso duas variáveis foram criadas: train_features e test_features. Como podemos observar, na variável train_features foi aplicado o vetorizador ao conjunto de treino “x_treino” com o método “fit_transform” que serve para ajustar o vetorizador aos documentos de treinamento e, em seguida, transformar esses documentos em uma matriz de vetores ponderados TF-IDF. Já na variável test_features foi aplicado o vetorizador já ajustado aos dados de teste usando o método “transform”. Isso converte os documentos de teste em vetores TF-IDF usando as mesmas transformações (stop words, minúsculas e remoção de acentos) que foram aplicadas aos dados de treinamento. Com os dados vetorizados é possível realizar a tarefa de classificação.

4.1.4 Treinamento e Classificação

Com a representação vetorial feita e cada vetor de atributo representando cada comentário da base de dados, procedemos com a tarefa de classificação do sentimento de cada opinião, aplicando técnicas de aprendizado de máquina. Para esta tarefa foram usados três tipos de algoritmos de classificação, cada um com uma característica diferente, ou seja, com princípios de funcionamento diferentes, porém são similares quanto à sua finalidade de categorização e capacidade de análise dos dados. Assim, os classificadores usados foram;

- *Perceptron*;
- *Linear Support Vector Classification (LinearSVC)*;
- *Logistic Regression.*

Para realizar esse processo, criamos um objeto denominado "Classificador", por meio do qual cada um dos classificadores mencionados acima foi implementado utilizando o pacote *scikit-learn*. Essa abordagem nos permite utilizar e comparar os diferentes classificadores de forma organizada e eficiente. Dessa forma, explanando as características e aplicações de cada um, é possível entender como cada classificador aborda as peculiaridades dos dados e como eles se comportam com relação ao problema. Essa análise detalhada nos fornece uma visão mais clara das vantagens e limitações de cada modelo, possibilitando a seleção criteriosa do mais adequado para a tarefa em questão. Para cada classificador foram empregados parâmetros essenciais que eram necessários, a fim de contribuir para que entregassem o melhor desempenho possível, aprimorando os resultados. A Figura 19 ilustra os três tipos de classificadores utilizados e os parâmetros de cada um.

Figura 19 – Classificadores e parâmetros utilizados.

```
# Implementando o Perceptron
from sklearn.linear_model import Perceptron
classificador = Perceptron(max_iter=100, alpha=0.0001, early_stopping=True,
    random_state=1, shuffle=True)

# Implementando o LinearSVC
from sklearn.svm import LinearSVC
classificador = LinearSVC(max_iter=1000, tol=0.0001)

# Implementando o LogisticRegression
from sklearn.linear_model import LogisticRegression
classificador = LogisticRegression(random_state=1, max_iter=100, tol=0.0001,
 solver='newton-cg')
```

Fonte: Autoria própria (2023).

O primeiro classificador utilizado foi o *Perceptron*. Com podemos ver na Figura 19, para utilizar o *Perceptron* foi preciso importá-lo utilizando o pacote *scikit-learn*, assim, com o *Perceptron* importado, ajustamos parâmetros a fim de assegurar um desempenho eficaz do classificador. O primeiro parâmetro é o "*max_iter*=100*"* que define o número máximo de interações que o algoritmo de treinamento executará para atualizar os pesos do modelo. O segundo parâmetro é o "*alfha*=0.0001", esse parâmetro controla a taxa de aprendizado do *Perceptron*, isto é, determina a magnitude da atualização dos pesos durante cada iteração. O terceiro parâmetro é o "*early_stopin*=True" que garante que o treinamento seja interrompido se não houver melhora significativa no desempenho do modelo durante o treinamento. Isso ajuda a evitar o excesso de ajuste (*Overfitting*). O quarta parâmetro se trata do "*random_state*=1", é usado para embaralhar os dados de treinamento, quando o quinto parâmetro "*shuffle*=True" é ativado.

O segundo classificador utilizado foi o *LinearSVC*. De forma análoga ao *Perceptron*, esse algoritmo foi importando utilizando o pacote *scikit-learn*. Para esse classificador também foi necessário utilizar parâmetros a fim de garantir um melhor desempenho. O primeiro parâmetro utilizado foi o mesmo do *Perceptron*, o "*max_iter*=1000*"*, porém com o número de interações maior. O segundo parâmetro utilizado foi o "*tol*=0.0001", que é um parâmetro de tolerância, ou seja, é um critério para determinar quando o processo de otimização deve parar. Assim, se a melhoria nas iterações subsequentes for menor que essa tolerância, o treinamento é interrompido.

Por fim, o último classificador utilizado foi *LogisticRegression*. Para fazer uso dele também houve a necessidade de importá-lo utilizando o pacote *scikit-learn*. Assim como os demais, houve necessidade de implementar parâmetros para garantir um melhor desempenho. O primeiro parâmetro utilizado foi o "*random_state*=100", esse tem a mesma função do parâmetro do *Perceptron*, ou seja, é usado para embaralhar os dados de treinamento. O segundo parâmetro é o "*max_iter*=100*"*, que é similar ao parâmetro do *Perceptron* e define o número máximo de interações que o algoritmo de treinamento executará. O terceiro parâmetro é o "*tol=0.0001*", que possui a mesma função do parâmetro utilizado no *LinearSVC*, isto é, um critério para determinar quando o processo de otimização deve parar. O último parâmetro é o "*solver*=newton-cg", esse parâmetro define o método de solução a ser usado pelo algoritmo de otimização durante o treinamento do modelo, e "newton-cg" refere-se ao método de otimização de Newton-Conjugado, que é uma abordagem eficaz para resolver problemas de otimização em regressão logística. Dessa forma, com os modelos criados, a etapa seguinte é a validação dos modelos.

4.1.5 Validação dos Modelos

Após a criação dos modelos de cada classificador, procedemos com uma validação em comum para cada um deles, com o objetivo de avaliar o desempenho de cada algoritmo. Essa abordagem nos permitirá comparar e analisar objetivamente como os diferentes classificadores se comportam diante dos mesmos dados de validação, fornecendo resultados valiosos sobre a eficácia de cada modelo em relação à tarefa em questão. Para a validação dos modelos os seguintes métodos foram usados;

- Validação Cruzada;
- Acurácia.

A validação cruzada, foi realizada com 5 K-Folds, ou seja, o conjunto de dados foi dividido em 5 partes iguais. Em cada iteração, uma das partes foi usada como conjunto de teste, enquanto as outras quatro partes foram utilizadas como conjunto de treinamento. Esse processo foi repetido cinco vezes (5 *folds*), alternando a parte de teste em cada iteração. Essa abordagem garante que cada subconjunto de dados seja usado tanto para teste quanto para treinamento, resultando em uma avaliação mais precisa e robusta do desempenho do modelo, evitando o *Overfitting*. No fim o processo, uma média dos resultados obtidos foi realizada e utilizada como métrica final, isto é, cada *fold* gerou um resultado, com isso, foi feito uma média aritmética com esses valores encontrados. A Figura 20 ilustra a implementação da validação cruzada.

Figura 20 – Implementação da validação cruzada.

```
validação_cruzada = cross_val_score(classificador, train_features, y_treino, cv=5)
```

Fonte: Autoria própria (2023).

Como podemos observar pela Figura 20, foi declarado uma variável chamada validação_cruzada, com isso, utilizando o pacote *scikit-learn* foi usado a função "cross_val_score" que é utilizada para realizar a validação cruzada, em seguida foi fornecido o modelo (classificador) desejado para avaliar, na forma de parâmetro. Logo depois, o outro parâmetro usado foi o "train_features" que é o conjunto de características dos dados de treinamento já transformados no formato de matriz, após a etapa de vetorização. Posteriormente o parâmetro "y_treino" foi utilizado, ele representa a lista de rótulos correspondentes aos dados

de treinamento, contendo as saídas esperadas para cada exemplo do conjunto de treinamento. E por fim, o parâmetro "*cv*=5" que corresponde ao número de *folds* usado na validação cruzada foi empregado. Com isso, foi possível determinar o valor da acurácia média.

Logo, com o valor da acurácia média da validação cruzada, treinamos os três classificadores usando o conjunto de dados de treinamento (train_features) já vetorizado. Em seguida avaliamos o desempenho de cada um deles usando o conjunto de teste (test_features) vetorizado. A Figura 21 apresenta o código utilizado para realizar o treinamento dos classificadores.

Figura 21 – Código para o treinamento dos classificadores.

```
classificador.fit(train_features, y_treino)
predição = classificador.score(test_features, y_teste)*100
print('Acurácia nos dados de teste é {:.2f}%'.format(predição))
```

Fonte: Autoria própria (2023).

Como podemos ver pela Figura 21, para o treinamento do classificador foi utilizado a função "fit" que é um método da maioria dos modelos de aprendizado de máquina do pacote *Scikit-learn* que realiza o processo de treinamento. Após isso, foi usado os parâmetros "train_features" e "y_treino" que é os dados de treinamento que o modelo está sendo treinado e o rótulo dos dados, respectivamente. Com isso, cada um dos modelos de classificadores foi usado e os resultados foram analisados para determinar a eficácia relativa de cada abordagem. Na próxima seção, serão descritos os resultados da implementação dos modelos de classificadores propostos anteriormente.

5 RESULTADOS E DISCUSSÕES

Nesta seção, serão apresentados os resultados da análise de sentimentos realizada sobre os dois conjuntos de dados selecionados, contendo reviews de filmes. Dessa forma, verificando quais os melhores métodos empregados e quais as vantagens e desvantagens dos algoritmos de classificação para a tarefa em questão.

5.1 RESULTADOS DOS CLASSIFICADORES

Após a implementações dos três algoritmos de classificação no mesmo conjunto de dados, verificou-se a eficácia de cada um. Essa análise possibilitou a comparação entre eles, visando identificar qual demonstrou um desempenho melhor. Nas próximas seções, para cada algoritmo, é demonstrado o resultado da sua acurácia e a eficácia dos métodos utilizados.

5.1.1 Classificador Perceptron

Para tarefas de classificação de textos existem algoritmos baseados em redes neurais que são eficazes quando implementados de forma correta e em casos não muito complexos. O *Perceptron* é um desses, trata-se de uma das formas mais simples de rede neural que pode trazer resultados satisfatório para a análise de sentimento. Para este trabalho o *Perceptron* foi usado em reviews (comentários) de filmes para avaliar e designar o sentimento presente em cada comentário. Para a implementar o algoritmo de classificação e seus devidos parâmetros foi feito o uso da biblioteca *Scikit-learn*. Assim, após sua implementação e seus devidos parâmetros, foi realizada uma avaliação abrangente para medir a eficácia do *Perceptron* na análise de sentimento em reviews de filmes. A métrica de desempenho escolhida foi a acurácia, que mede a proporção de classificações corretas em relação ao total de classificações realizadas. A Tabela 3 ilustra a acurácia média e a acurácia real alcançada, utilizando o *Perceptron* como classificador, nos dados de treino e nos dados de teste, respectivamente.

Tabela 3 – Acurácia nos dados de treino e teste *Perceptron*.

Acurácia nos dados de treino	Acurácia nos dados de teste
84,25%	73,02%

Fonte: Autoria própria (2023).

Como podemos ver pela Tabela 3, a acurácia nos dados de treino demonstra a capacidade do *Perceptron* de aprender e se ajustar aos padrões presentes no conjunto de treinamento, enquanto a acurácia nos dados de teste revela a habilidade do modelo em generalizar seus aprendizados para dados não vistos anteriormente, já que houve uma pequena diferença de aproximadamente 11%. Isso sugere que o modelo está conseguindo evitar o sobreajuste e mantendo um bom desempenho em novos cenários, mesmo com essa ligeira disparidade entre as acurácias nos conjuntos de treino e teste. Essa análise comparativa evidencia a eficácia do *Perceptron* como classificador e sua capacidade de lidar com novos dados.

5.1.2 Classificador LinearSVC

Com relação ao *LinearSVC*, ele mostrou resultado significativa com relação a tarefa em questão. A métrica escolhida para verificar sua eficiência foi a mesma do *Perceptron*. Para implementar esse algoritmo também foi utilizado a biblioteca *Scikit-learn*. A Tabela 4 mostra o resultado da acurácia média e a acurácia real obtida pelo *LinearSVC*, com relação aos dados de treino e de teste, respectivamente.

Tabela 4 – Acurácia nos dados de treino e teste *LinearSVC*.

Acurácia nos dados de treino	Acurácia nos dados de teste
86,27%	76,96%

Fonte: Autoria própria (2023).

Assim, como podemos observar pela Tabela 4, o *LinearSVC* demostrou um resultado satisfatório para a tarefa em questão. Nos dados de treino o classificador conseguiu uma performance ótima, destacando sua capacidade de aprender e se ajustar aos padrões presentes no conjunto de treinamento. Quanto aos dados de teste, o modelo manteve um nível considerável na acurácia, destacando sua habilidade em generalizar para exemplos não vistos anteriormente.

5.1.3 Classificador LogisticRegression

Para o classificador *LogisticRegression* o cenário foi semelhante aos outros algoritmos demonstrados até o momento. Também foi implementado usado a biblioteca *Scikit-learn*. Nos

dados de treino o modelo conseguiu uma boa taxa de aprendizado, assim como nos dados de treino. A Tabela 5 ilustra os valores obtidos na acuraria dos dados de treino e de teste.

Tabela 5 – Acurácia nos dados de treino e teste *LogisticRegression*.

Acurácia nos dados de treino	Acurácia nos dados de teste
86,97%	77,45%

Fonte: Autoria própria (2023)

Como podemos perceber pela Tabela 5, a acurácia nos dados de treino é significativa, já nos dados de teste a acurácia se manteve aceitável, visto que a disparidade entre os dois valores chega a aproximadamente 9,5%, dessa maneira demostrando a capacidade do classificador em realizar previsões consistentes e generalizadas para novos dados.

5.2 DISCUSSÃO DOS RESULTADOS OBTIDOS

Com os resultados obtidos para os três classificadores podemos notar uma eficácia dos três algoritmos de classificação. Como verificado nas seções anteriores podemos verificar que o *Perceptron* teve uma acurácia nos dados de treino de 84,25% e nos dados de teste teve um valor de 73, 02%, assim, a diferença entre os dois valores de acurácia corresponde a aproximadamente 11%. O *LinearSVC* obteve uma acurácia relativamente melhor nos dados de treinamento, alcançando um valor de 86,27%, nos dados de teste também foi verificado melhor eficácia com relação ao *Perceptron*, alcançando um valor de 76,96%. Para esse caso, a diferença entre os valores de acurácia nos dados de treino e teste chegaram a 9,31%. No caso do RegressionLogistic, os valores de acurácia nos conjuntos de treinamento e teste apresentaram uma notável semelhança com o *LinearSVC*, registrando valores de 86,97% e 77,45%, respectivamente. No entanto, houve uma diferença de 9,45% entre os valores. Sendo assim, como podemos observar pelos valores de acurácia alcançados, os três classificadores obtiveram um bom desempenho, com valores da acurácia similares, indicando uma consistência em suas capacidades de classificação e generalização para novos modelos. Esses valores são muito semelhantes pois os três modelos de classificadores seguem uma metodologia parecida, no entanto cada um desses algoritmos possui suas próprias vantagens e características que são intrínsecas e podem ser mais apropriadas para diferentes contextos ou conjuntos de dados, oferecendo aos desenvolvedores e analistas a flexibilidade necessária para escolher a abordagem mais adequada às demandas específicas de seus projetos. Além disso, ao ser testados

em novas frases ambos os algoritmos conseguiram acertar o sentimento presente nas frases. Para comprovar isso, os três algoritmos de classificação foram testados com novos comentários de filmes de diversas categorias e com algumas adversidades como má pontuação, abreviações de palavras, e outros desafios linguísticos. A Tabela 6 ilustra os comentários, juntamente com a análise de sentimento realizada pelos três algoritmos de classificação.

Tabela 6 – Comentários de filmes com seus respectivos sentimentos.

Comentários	Sentimento do *Perceptron*	Sentimento do *LinearSVC*	Sentimento do *LogisticRegression*
O filme é extremamente decepcionante, pela quantidade de atores famosos que traz. Busca fazer uma crítica a sociedade atual, mas apenas por um ponto de vista. Divide a sociedade em dois e coloca os bilionários como o centro da destruição do mundo.	Negativo	Negativo	Negativo
Até que gostei, é um filme simples, sem muitas pretensões e com personagens carismáticos. As referências mexicanas que são conhecidas aqui no Brasil são ótimas.	Positivo	Positivo	Positivo
5?! Mds. A militância iria achar o que se o filme recebesse uma nota entre 2 e 3? Filmes de conteúdo (ou falta de) igual não passam de 3 para os 'críticos'! Olha... esse filme pra ser ruim tem que melhorar muito, o filme é feito de militante para militante, se você não for do grupo deles (mídia, partidos, etc) verá um filme nota 2,5!	Negativo	Negativo	Negativo
Eu assisti e recomendo, divertido, prende a atencao o tempo todo! Compre a pipoca	Positivo	Positivo	Positivo

e o guarana e assista no cinema que vale a pena totalmente!			
O filme é o mais 'fraco' da franquia. Ele engata um pouco na última meia hora, a primeira hora é pura enrolação. Confesso q fiquei decepcionada, geralmente as produções da BlumHouse são ótimas, mas essa deixou a desejar. Passa a impressão q é uma prévia para uma futura produção, porém poderia ter sido melhor explorado...	Negativo	Negativo	Negativo
É sem dúvidas um show visual, cgi impecável, e efeitos especiais dignos do orçamento de 250 milhões. Mas acaba por aí (aviso de spoilers) ...	Positivo	Positivo	Positivo
É um filme bobo que entrega aquilo que promete, um roteiro raso com ameaças que não geram a sensação de perigo que deveriam, mas possui diversos personagens carismáticos.	Negativo	Negativo	Negativo
O filme é maravilhoso. A experiência imersiva nas belas paisagens de Pandora são o ponto alto do filme. O mundo criado por James Cameron é realmente fantástico. Há, no entanto, alguns problemas no roteiro, mas nada que tenha atrapalhado a experiência.	Positivo	Positivo	Positivo
O final ficou muito vago, aliás o filme tem muitos deslizes.	Negativo	Negativo	Negativo

Fonte: Autoria própria (2023).

Como podemos observar pela Tabela 6, há diversos comentários relacionados a filmes de todas as categorias, expressando uma variedade de opiniões e emoções por parte dos espectadores. Seus respectivos sentimentos são expressos de acordo com a análise dos

algoritmos. Com isso, ao analisar os comentários, observamos que os classificadores desenvolveram com eficácia a tarefa de analisar o sentimento presente em cada nova frase, já que ao ler as frases podemos perceber o sentimento presentes nelas, dessa maneira, validando assim a precisão e a capacidade dos métodos de análise de sentimento empregados. Deste modo, todos os algoritmos de classificação desempenharam uma boa análise já que conseguiram acertar o sentimento predominante de cada frase.

6 CONCLUSÃO

Em conclusão, o aprendizado de máquina aplicado à análise de sentimentos representa um campo de pesquisa e aplicação em constante evolução, com implicações significativas em diversos setores. A capacidade de extrair informações emocionais a partir de textos e comentários tornou-se uma ferramenta valiosa em áreas como marketing, pesquisa de mercado, atendimento ao cliente e muito mais.

Ao decorrer da pesquisa, desenvolveu-se um estudo detalhado sobre as melhores técnicas de processamento de texto e aprendizado de máquina, voltadas para a tarefa de análise de sentimentos no contexto de comentários de filmes. Após a utilização de três diferentes algoritmos de aprendizado de máquina estudados na literatura, procedeu-se à avaliação de sua eficácia na tarefa de classificação. Foi observado que todos os três algoritmos estudados demonstraram resultados satisfatórios e muito próximos entre si, o que pode ser atribuído à semelhança em suas abordagens adotadas para a análise de sentimentos. O resultado alcançados por eles ficaram entre 73% e 77%, comprovando sua capacidade de generalização. Outrossim, após serem testados em comentários nunca visto antes, esses algoritmos demonstraram a notável capacidade de identificar com precisão todos os sentimentos presentes.

Além disso, comprovou-se que as técnicas de processamento de texto realmente são de grande importância, visto que a linguagem humana é rica em símbolos e estruturas sintáticas que a linguagem de máquina não é intrinsecamente capaz de compreender sem a aplicação de abordagens específicas. No entanto, à medida que avançamos, é necessário continuar aprimorando e refinando nossas abordagens, explorando novos algoritmos e técnicas que nos permitam compreender contextos mais complexos de forma efetiva.

Portanto, com esse estudo espera-se incentivar ainda mais a pesquisa e desenvolvimento de ferramentas capazes de nos auxiliar em diversas tarefas, visto que essas ferramentas proporcionam com êxito a realização delas. Ademais, espera-se que outros modelos de algoritmos de classificação sejam estudados e que outras técnicas também sejam estudadas para comprovar sua efetividade.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, Eliane Martins de. **Aplicação do Word2vec e do Gradiente Descendente Estocástico em Tradução Automática**. 2016. 78 f. Dissertação (Mestrado) - Curso de Programa de Pós-Graduação, Escola de Matemática Aplicada, Fundação Getulio Vargas, Rio de Janeiro, 2016.

BELLINI, Ana Júlia de Oliveira. **Análise de Sentimentos em Críticas de Filmes por meio de Algoritmos Clássicos de Aprendizado de Máquina**. 2019. 146 p. Trabalho de conclusão de curso (Bacharela em Ciência da Computação) - Universidade Federal de São Paulo, São José dos Campos, 2019.

CLAVERA, WALTER V. **Aprendizado de Máquina (Machine Learning)**. [S. l.], 22 maio 2019. Disponível em: https://www.redesdesaude.com.br/aprendizado-de-maquina-machine-learning/. Acesso em: 23 set. 2023.

CRESCENCIO, M.; GONÇALVES, A. L..; TODESCO, J. L. **Um processo de classificação de texto: análise de sentimento das opiniões no tripadvisor sobre a atração oktoberfest blumenau**. Anais do Congresso Internacional de Conhecimento e Inovação – ciki, [S. l.], v. 1, n. 1, 2020. DOI: 10.48090/ciki.v1i1.867.

DIAS, Ariel da Silva. **Processamento de Linguagem Natural**. São Paulo: Platos Soluções Educacionais S.A, 2021.

ESPINHO, James. **Regressão Logística Explicada**. [S. l.], 8 fev. 2020. Disponível em: https://towardsdatascience.com/logistic-regression-explained-9ee73cede081. Acesso em: 23 set. 2023.

FACELLI, Katti *et al*. **Inteligência Artificial**: uma abordagem de aprendizado de máquina. Rio de Janeiro: Ltc, 2011

FELTRIN, Fernando Belomé. **Inteligência Artificial com Python**. 1. ed. [S. l.: s. n.], 2020. 182 p

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**: concepts, tools, and techniques to build intelligent systems. 2. ed. Canada: O'reilly Media, 2019. 1150 p.

GONZALEZ, Leandro de Azevedo. **Regressão Logística e suas Aplicações**. 2018. 4 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Tecnológicas, Universidade Federal do Maranhão, São Luiz, 2018.

HAYKIN, Simon. **Redes Neurais**: princípios e práticas. 2. ed. Canadá: Artmed, 1999. 908 p.

KREMER, Gustavo Ribeiro. **Algoritmos de Aprendizado de Máquina Aplicados a Dados Públicos para Obtenção de Insights em Segurança Pública**. 2023. 76 f. Monografia (Bacharelado em Ciência da Computação) - Curso de Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2023.

MARTINS, Júlio Serafim; MARIANO, Diego César Batista; LENZ, Maikon Lucian; MARTINS, Juliano Vieira; SILVA, Michel Bernardo Fernandes da; RODRIGUES, Sofia Maria Amorim Falco; BEZERRA, Wheslley Rimar; OLIVEIRA, Raiza Artemam de; PICHETTI, Roni Francisco. **PROCESSAMENTOS DE LINGUAGEM NATURAL**. Porto Alegre: Sagah, 2020. 285 p. Revisão técnica: Adriana Neves dos Reis, Júlia Mara Colleoni Couto, Olimar Teixeira Borges.

MCKINNEY, William Wesley. **Python para análise de dados**: tratamento de dados com pandas, numpy e ipython. 2. ed. São Paulo: Novatec, 2018. 685 p. Tradução de: Lúcia A. Kinoshita.

MIKOLOV, Tomas *et al*. Distributed Representations of Words and Phrases and their Compositionality. **Computer Science > Computation And Language**, [S.L.], v. 1, n. 6, p. 1-8, 16 out. 2013. ArXiv. http://dx.doi.org/10.48550/ARXIV.1310.4546.

MOTTA, Larissa Santos da. **Análise de Sentimentos em tweets sobre a pandemia covid-19 usando redes neurais long short-term memory**. 2021. 68 p. Trabalho de conclusão de curso (Bacharelado em sistemas de informação) - Instituto Federal do Espírito Santo, Serra, 2021.

OLIVEIRA, Thiago do Nascimento. **Construção e classificação de uma base textual em português**. 2023. 83 f. TCC (Graduação) - Curso de Ciência da Computação, Instituto de Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2023.

RASCHAKA, Sebastian; MIRJALILI, Vadih. **Python Marchine Learning**. 3. ed. Birmingham: Radhika Atitkar, 2019.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 3. ed. Rio de janeiro: Elsevier, 2013.

SANTOS, Cedric Michael dos. **Classificação de Documentos com Processamento de Linguagem Natural**. 2015. 2017 f. Dissertação (Mestrado) - Curso de Informática e Sistemas, E Engenharia Informática e de Sistemas, Instituto Superior de Engenharia de Coimbra, Coimbra, 2015.

SILVA, Ivan Nunes da; SPATTI, Danilo Ernane; FLAUZINO, Rogério Andrade. **Redes Neurais Artificiais**: para engenharia e ciências aplicadas. 2. ed. São Paulo: Artliber, 2010. 135 p.

SOUZA, Gabriel Araújo de. **Inteligência Artificial para a detecção e classificação de mensagens de texto relevantes em evidências criminais**. 2021. 83 f. Monografia (Graduação) - Curso de Ciência da Computação, Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, 2021.

SVG, Silh. **CÉLULA neurologia inteligência neurônio**. [S. l.]. Disponível em: https://svgsilh.com/pt/image/2022398.html. Acesso em: 18 set. 2023.

TIBURCIO, Gabriel Valentin. **Avaliação Experimental de Classificadores para Análise de Sentimentos em Dados de Redes Sociais**. 2021. 61 f. TCC (Graduação) - Curso de Sistemas de Informação, Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2021.

APÊNDICES

APÊNDICE A – EXPERIMENTO COM O *PERCEPTRON*

```
# Importando as bibliotecas
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Perceptron
from nltk.corpus import stopwords

# Importando a base de dados e removendo colunas
dados = pd.read_csv('C:/Users/ggust/Dropbox/PC/Documents/DATASET/imdbreviewsptbr.csv')
dados.drop(columns=['text_en', 'id'], inplace=True)

# Transformei a coluna sentiment para um rótulo binário
# (0 para sentimentos negativos, 1 para sentimentos positivos)
dados['label'] = dados['sentiment'].apply(lambda x: 0 if x == 'neg' else 1)

# Dividindo os dados de treino e teste
x_treino = dados.loc[:35000, 'text_pt'].values
y_treino = dados.loc[:35000, 'label'].values
x_teste = dados.loc[35000:, 'text_pt'].values
y_teste = dados.loc[35000:, 'label'].values

# Extraindo as características das avaliações usando o TF-IDF
vectorizer = TfidfVectorizer(stop_words=stopwords.words('portuguese'),
                             lowercase=True, strip_accents='unicode')
train_features = vectorizer.fit_transform(x_treino)
test_features = vectorizer.transform(x_teste)

# Usando o algoritmo de classificação
classificador = Perceptron(max_iter=100, alpha=0.0001, early_stopping=True,
     random_state=1, shuffle=True)

# Realizando a validação cruzada
validação_cruzada = cross_val_score(classificador, train_features, y_treino, cv=5)

# Acurácia média
print('Acurácia média: {:.2f}%'.format(sum(validação_cruzada)/len(validação_cruzada)*100))

# Treinando o classificador e mostrando sua acurácia
classificador.fit(train_features, y_treino)
predição = classificador.score(test_features, y_teste)*100
print('Acurácia nos dados de teste é {:.2f}%'.format(predição))

# Cria uma nova frase para teste
nova_frase1 = "O final ficou muito vago, aliás o filme tem muitos deslizes."

# Extrai características da nova frase usando o vetorizador ajustado
nova_frase_features = vectorizer.transform([nova_frase1])

# Usa o modelo treinado para prever a polaridade da nova frase
nova_frase_sentimento = classificador.predict(nova_frase_features)[0]

if nova_frase_sentimento == 1:
    print('O sentimento da nova frase é \033[32mpositivo\033[m')
else:
    print('O sentimento da nova frase é \033[31mnegativo\033[m')
```

Fonte: Autoria própria (2023).

APÊNDICE B – EXPERIMENTO COM O *LINEARSVC*

```python
# Importando as bibliotecas
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score
from sklearn.svm import LinearSVC
from nltk.corpus import stopwords

# Importando a base de dados e removendo colunas
dados = pd.read_csv('C:/Users/ggust/Dropbox/PC/Documents/DATASET/imdbreviewsptbr.csv')
dados.drop(columns=['text_en', 'id'], inplace=True)

# Transformei a coluna sentiment para um rótulo binário
# (0 para sentimentos negativos, 1 para sentimentos positivos)
dados['label'] = dados['sentiment'].apply(lambda x: 0 if x == 'neg' else 1)
```

```python
# Dividindo os dados de treino e teste
x_treino = dados.loc[:35000, 'text_pt'].values
y_treino = dados.loc[:35000, 'label'].values
x_teste = dados.loc[35000:, 'text_pt'].values
y_teste = dados.loc[35000:, 'label'].values

# Extraindo as características das avaliações usando o TF-IDF
vectorizer = TfidfVectorizer(stop_words=stopwords.words('portuguese'),
                             lowercase=True, strip_accents='unicode')
train_features = vectorizer.fit_transform(x_treino)
test_features = vectorizer.transform(x_teste)

# Usando o algoritmo de classificação
classificador = LinearSVC(max_iter=1000, tol=0.0001)
```

```python
# Realizando a validação cruzada
validação_cruzada = cross_val_score(classificador, train_features, y_treino, cv=5)

# Acurácia média
print('Acurácia média: {:.2f}%'.format(sum(validação_cruzada)/len(validação_cruzada)*100))

# Treinando o classificador e mostrando sua acurácia
classificador.fit(train_features, y_treino)
predição = classificador.score(test_features, y_teste)*100
print('Acurácia nos dados de teste é {:.2f}%'.format(predição))
```

```python
# Cria uma nova frase para teste
nova_frase1 = "O final ficou muito vago, aliás o filme tem muitos deslizes."

# Extrai características da nova frase usando o vetorizador ajustado
nova_frase_features = vectorizer.transform([nova_frase1])

# Usa o modelo treinado para prever a polaridade da nova frase
nova_frase_sentimento = classificador.predict(nova_frase_features)[0]

if nova_frase_sentimento == 1:
    print('O sentimento da nova frase é \033[32mpositivo\033[m')
else:
    print('O sentimento da nova frase é \033[31mnegativo\033[m')
```

Fonte: Autoria própria (2023).

APÊNDICE C – EXPERIMENTO COM O *LOGISTICREGRESSION*

```
# Importando as bibliotecas
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score
from nltk.corpus import stopwords
from sklearn.linear_model import LogisticRegression

# Importando a base de dados e removendo colunas
dados = pd.read_csv('C:/Users/ggust/Dropbox/PC/Documents/DATASET/imdbreviewsptbr.csv')
dados.drop(columns=['text_en', 'id'], inplace=True)

# Transformei a coluna sentiment para um rótulo binário
# (0 para sentimentos negativos, 1 para sentimentos positivos)
dados['label'] = dados['sentiment'].apply(lambda x: 0 if x == 'neg' else 1)
```

```
# Dividindo os dados de treino e teste
x_treino = dados.loc[:35000, 'text_pt'].values
y_treino = dados.loc[:35000, 'label'].values
x_teste = dados.loc[35000:, 'text_pt'].values
y_teste = dados.loc[35000:, 'label'].values

# Extraindo as características das avaliações usando o TF-IDF
vectorizer = TfidfVectorizer(stop_words=stopwords.words('portuguese'),
                             lowercase=True, strip_accents='unicode')
train_features = vectorizer.fit_transform(x_treino)
test_features = vectorizer.transform(x_teste)

# Usando o algoritmo de classificação
classificador = LogisticRegression(random_state=1, max_iter=100, tol=0.0001,
 solver='newton-cg')
```

```
# Realizando a validação cruzada
validação_cruzada = cross_val_score(classificador, train_features, y_treino, cv=5)

# Acurácia média
print('Acurácia média: {:.2f}%'.format(sum(validação_cruzada)/len(validação_cruzada)*100))

# Treinando o classificador e mostrando sua acurácia
classificador.fit(train_features, y_treino)
predição = classificador.score(test_features, y_teste)*100
print('Acurácia nos dados de teste é {:.2f}%'.format(predição))
```

```
# Cria uma nova frase para teste
nova_frase1 = "O final ficou muito vago, aliás o filme tem muitos deslizes."

# Extrai características da nova frase usando o vetorizador ajustado
nova_frase_features = vectorizer.transform([nova_frase1])

# Usa o modelo treinado para prever a polaridade da nova frase
nova_frase_sentimento = classificador.predict(nova_frase_features)[0]

if nova_frase_sentimento == 1:
    print('O sentimento da nova frase é \033[32mpositivo\033[m')
else:
    print('O sentimento da nova frase é \033[31mnegativo\033[m')
```

Fonte: Autoria própria (2023).