CEUB | PIC

IDENTIFICAÇÃO DE DEPRESSÃO E/OU ANSIEDADE EM USUÁRIOS DE REDES SOCIAIS

Professor orientador: Salvador Alves de Melo Junior

Alunos: Cecília Neves Vieira e
Guilherme Lopes Costa Pereira

PROGRAMA DE
INICIAÇÃO CIENTÍFICA
PIC/CEUB

RELATÓRIOS DE PESQUISA

VOLUME 9 Nº 1- JAN/DEZ
·2023·

ISSN: 2595-4563

fapdf CNPq CEUB

CENTRO UNIVERSITÁRIO DE BRASÍLIA - CEUB
PROGRAMA DE INICIAÇÃO CIENTÍFICA

CECÍLIA NEVES VIEIRA
E GUILHERME LOPES COSTA PEREIRA

IDENTIFICAÇÃO DE DEPRESSÃO E/OU ANSIEDADE EM USUÁRIOS DE REDES SOCIAIS

Relatório final de pesquisa de Iniciação Científica apresentado à Assessoria de Pós-Graduação e Pesquisa.

Orientação: Salvador Alves de Melo Junior

BRASÍLIA

2024

RESUMO

Considerando a crescente problemática de doenças como depressão, este trabalho busca desenvolver um modelo preditivo capaz de classificar postagens realizadas em redes sociais em relação à depressão, utilizando técnicas de processamento de linguagem natural (PLN). A depressão é uma condição de saúde mental que afeta milhões de pessoas em todo o mundo e sua detecção precoce é crucial para oferecer suporte e intervenções apropriadas. Com o aumento da utilização das redes sociais, essas plataformas tornaram-se uma fonte rica de dados onde as pessoas expressam pensamentos e emoções, tornando este ambiente virtual uma potência para aplicação de técnicas de PLN, desde a coleta e pré-processamento dos dados, à aplicação de métodos de extração de características de textos e elaboração de modelos preditivos, para auxílio na questão da saúde mental. Este estudo realiza uma revisão bibliográfica de artigos recentes, buscando entender e aplicar as melhores técnicas e modelos, incluindo o uso de algoritmos de machine learning, como *Support Vector Machine* (SVM) e *Random Forest*, e técnicas de PLN, como TF-IDF e *Bag of Words*, que são essenciais para a coleta e limpeza de dados, entendimento das nuances linguísticas e implementação de algoritmos preditivos. Neste contexto, o objetivo do presente trabalho é desenvolver modelos de análise preditiva que aprendam a identificar discursos com tendências a distúrbios nas redes sociais, contribuindo para a literatura existente e abrindo novas possibilidades para a aplicação de tecnologias emergentes na área da saúde mental. Espera-se que os modelos desenvolvidos possam alcançar alta precisão na detecção de sinais de depressão e ansiedade, proporcionando uma ferramenta valiosa para profissionais de saúde, plataformas de redes sociais e para a elaboração de políticas públicas. Os resultados obtidos têm o potencial de melhorar significativamente a identificação precoce desses distúrbios, facilitando intervenções mais eficazes e oportunas. O impacto deste trabalho pode contribuir para auxiliar na promoção de tratamentos preventivos, bem como na diminuição do estigma que há sobre as questões de saúde mental.

Palavras-chave: depressão; machine learning; processamento de linguagem natural.

SUMÁRIO

1. INTRODUÇÃO

Doenças como ansiedade e depressão estão se tornando cada vez mais tópicos de discussão e interesse para a saúde pública, principalmente após os últimos anos, em que a incidência mundial destas doenças aumentou em 25% devido à pandemia (Organização Pan-Americana da Saúde – OPAS, 2022) [1]. A estigmatização das doenças mentais, uma vida cada vez mais acelerada e que resta pouco tempo para o autocuidado e lazer, desigualdades sociais e, como citado anteriormente, a pandemia da COVID-19, aumentam os fatores de risco e necessidade de atenção a esses tópicos (Organização Mundial da Saúde – OMS, 2022) [2].

Além desses distúrbios, o estresse excessivo a que os indivíduos constantemente são submetidos também resulta em desequilíbrios químicos no organismo que pode levar a diversas doenças físicas e mentais. De acordo com uma pesquisa online realizada em 2018, no Reino Unido, com 4.169 respostas, 74% das pessoas haviam se sentido tão estressadas no ano anterior, que ficaram sobrecarregadas ou incapazes de lidar com essa condição; dessas, 51% relataram se sentir depressivas e 61% ansiosas (*Mental Health Foundation*, 2018) [3].

Como opção para analisar o comportamento das pessoas, atualmente, tem-se as redes sociais que contam com milhões de usuários ao redor do mundo. Com a evolução da tecnologia, possibilitando que grandes volumes de dados fossem coletados e analisados, é possível identificar padrões e tendências relacionadas a distúrbios mentais em atividades diárias dos usuários nas plataformas sociais.

Em estudo publicado por Tsugawa e colaboradores [4], os resultados obtidos demonstravam que a depressão pode ser reconhecida nos usuários com precisão aproximada de 69% e também foi estipulado que dois meses de observação são suficientes para a previsão, com longos períodos de coleta de dados diminuindo a precisão. Outro pesquisador com diversos artigos na área, De Choudhury e demais colaboradores em estudo publicado em 2021 [5], utilizaram a rede social *Twitter* (atualmente, X) para prever depressão em indivíduos, com resultados de precisão de classificação em 70% e demonstrando, também, que usuários com depressão exibem

menor atividade social, mais emoções negativas e um aumento de pensamentos religiosos e sobre problemas medicinais.

Dado esse problema crescente, este estudo se propõe a desenvolver ferramentas utilizando técnicas avançadas de Processamento de Linguagem Natural (PLN) e aprendizado de máquina para identificar, em postagens de usuários em redes sociais, tendências de depressão e ansiedade. Ao explorar o vasto conjunto de dados disponíveis nas redes sociais, busca-se contribuir para a minimizar o problema, detectando, de forma precoce, sinais desses distúrbios mentais, contribuindo assim para intervenções mais eficazes e oportunas no campo da saúde pública.

Este trabalho utilizará revisão bibliográfica realizada pelos autores como base para escolha da melhor abordagem a ser empregada no desenvolvimento dos modelos, incluindo qual algoritmo de *machine learning* será aplicado nos *datasets* obtidos e processados. Nesta análise de literatura, foi realizado um estudo exploratório sobre as diferentes técnicas de obtenção de dados, pré-processamento e elaboração de modelos utilizadas em artigos elaborados entre os anos de 2018 e 2023.

As publicações analisadas traziam assuntos diversos sobre análises de sentimentos desenvolvidas utilizando variadas fontes de aquisição dos dados. Em relação à obtenção dos dados, alguns estudos utilizaram material coletados de redes e fóruns sociais enquanto outros desenvolveram questionários próprios. Já sobre a temática, a grande maioria trazia análises sobre predição de depressão e ansiedade, uma menor quantidade continha estudos sobre estresse e um deles apresentava uma extração de sentimento de postagens de um fórum médico, focando em condições médicas e medicações e as reações do usuário em relação a essas temáticas. Mais detalhes sobre as publicações podem ser encontrados na Fundamentação Teórica do presente artigo.

Assim, este trabalho busca contribuir para os estudos de processamento de linguagem natural, principalmente na identificação de sinais de depressão em usuários de redes sociais. As técnicas empregadas e os empecilhos encontrados podem auxiliar estudos futuros na área; e os resultados obtidos têm potencial para auxiliar na

elaboração de políticas públicas e no aprimoramento das redes sociais para apoio de usuários em situações preocupantes relacionadas à saúde mental.

OBJETIVOS

Dado este contexto da grande utilização de redes sociais e aumento do adoecimento mental pela sociedade atual, o objetivo deste trabalho é desenvolver um modelo preditivo, utilizando técnicas avançadas de Processamento de Linguagem Natural (PLN), capaz de classificar postagens em relação à depressão, utilizando técnicas de processamento de linguagem natural.

Para garantir a eficácia e clareza deste objetivo, além de um desenvolvimento bem definido, foi utilizada a metodologia SMART, que permite definir metas que são Específicas, Mensuráveis, Atingíveis, Relevantes e Temporais:

- **Específica:** elaboração de uma abordagem para o processamento dos dados com o intuito de identificar postagens com cunho depressivo;
- **Mensurável:** o objetivo será mensurado utilizando matriz de confusão e métricas de desempenho, como acurácia, precisão, especificidade, *recall* e *f1-score*; que permitem analisar o desempenho dos modelos de predição;
- **Atingíveis:** existem diversos métodos, ferramentas e tecnologias para o desenvolvimento do objetivo, tornando-o atingível;
- **Relevantes:** contribuição para aprimoramento das técnicas de processamento de linguagem natural, podendo auxiliar em políticas públicas; além disso, a literatura nesta área em português é escassa;
- **Temporais:** o desenvolvimento será realizado ao longo de um ano, incluindo coleta e processamento dos dados, e testes com os algoritmos.

Além desse objetivo geral, existem os específicos, que consistem em identificar os principais indicadores, validação do modelo preditivo desenvolvido por meio de testes com usuários e análise dos resultados obtidos para identificação de eventuais

limitações e oportunidades de melhoria na detecção dos sinais de depressão em usuários de redes sociais.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. ANÁLISE DE DADOS E SENTIMENTOS

A identificação de depressão e ansiedade em usuários de redes sociais é um campo de estudo que combina técnicas de análise de sentimentos e aprendizado de máquina para detectar sinais desses transtornos mentais em postagens nas redes sociais. A análise de sentimentos utiliza processamento de linguagem natural (PLN), estatísticas e aprendizado de máquina para identificar e extrair informações subjetivas de dados textuais.

Existem diferentes tipos de análise de dados, processo que busca transformar dados coletados por diversas fontes em informações que possam auxiliar na tomada de decisões [6], cada uma objetivando algum propósito.

A análise descritiva tem foco no passado, oferecendo uma descrição de dados para retratar o que aconteceu, utilizando métricas da estatística como média, mediana, moda, desvio padrão, variância, frequência; é a análise mais aplicada pelos analistas de dados.

Outro tipo é a análise exploratória, que objetiva descobrir relações ou novos fatos desconhecidos, utilizando correlação entre variáveis e técnicas de regressão e análise de variância; deriva da análise descritiva. A análise diagnóstica também está ligada à análise descritiva, tendo como objetivo entender o porquê dos fatos, realizando uma verificação mais ampla ao destrinchar os dados, utilizando mapeamento de anomalias, regressão estatística, teoria da probabilidade e outras técnicas.

Em relação a previsões futuras, a análise preditiva utiliza dados do passado, também chamados de dados históricos, para prever eventos por vir ou como algo se comportará; sendo uma das técnicas mais utilizadas atualmente. Esse tipo analisa padrões, utilizando estatística e *machine learning*, para propor ações.

Por fim, a análise prescritiva fornece *insights* de possíveis cenários, mais voltada para projeções e consequências de decisões adotadas. É o tipo mais complexo de análise, envolvendo conhecimento da área, mineração de dados, *machine learning*, e ferramentas avançadas de *business intelligence*. Com essa forma de análise é possível, também, gerar tomadas de decisões de forma automática ou semiautomática, como sistemas de concessão de crédito.

A análise preditiva e a análise prescritiva são as principais áreas de atuação do cientista de dados.

2.2. MACHINE LEARNING E PLN

Aprendizado de máquina, ou *machine learning*, é um ramo da inteligência artificial que visa fazer com que os computadores, por meio do uso de algoritmos, sejam capazes de aprender ao reconhecerem padrões de dados e de realizarem análises preditivas, sendo diretamente ligada à Ciência de Dados. Como citado no livro "*Fundamentals of Machine Learning for Predictive Data Analytics*" [7], *machine learning* é um "processo automatizado que extrai padrões de dados" (tradução livre) e para construir modelos preditivos, é mais comumente utilizado o aprendizado de máquina supervisionado, em que o modelo aprende a relação entre um conjunto de características descritivas e uma característica alvo, baseado em conjunto de exemplos históricos ou instâncias.

A evolução desse ramo está diretamente ligada à melhoria da capacidade de processamento dos sistemas, pois com isso foi possível o rápido tratamento e análise de uma maior variedade e volume de dados, estando, também, relacionada ao surgimento do *Big Data*, que, de acordo com (Oracle, s.d.) [8], "é um conjunto de dados maior e mais complexo, especialmente de novas fontes de dados."

Já em relação à análise de sentimento, também chamada de mineração de opiniões, pode ser definida como uma técnica de classificação de textos em que se busca determinar em qual contexto ele se encontra, geralmente categorizados como positivos, negativos ou neutros.

A análise de sentimentos tem como desafio lidar com dados ruidosos e detectar sarcasmo e ironia. A revisão manual e a interpretação humana são frequentemente necessárias para validar os resultados. Para identificar depressão e ansiedade, combina-se a análise de sentimentos com indicadores psicológicos e linguísticos específicos.

Pessoas com depressão ou ansiedade tendem a usar mais palavras relacionadas a emoções negativas, solidão e desesperança, além de apresentarem padrões de sono [9] e atividade irregulares em suas postagens.

2.3. TÉCNICAS E ESTUDOS RELEVANTES

Nesse contexto também, como citado anteriormente, foram exploradas diversas publicações sobre análise de sentimentos e *machine learning*, além de processos de obtenção e limpeza de dados e técnicas de seleção e extração de características. Abaixo os artigos utilizados:

- "*An in-depth analysis of machine learning approaches to predict depression*"; autores: Md. Sabab Zulfiker, Nasrin Kabir, Al Amin Biswas, Tahmina Nazneen, Mohammad Shorif Uddin [10];
- "Análise de Técnicas de PLN e *Machine Learning* na Detecção de Indícios de Depressão"; autores: Dyonathan Jordan, Victor Cesar, Lucas do Carmo [11];
- "Análise de Sentimento em Redes Sociais para a Língua Portuguesa Utilizando Algoritmos de Classificação"; autores: Erikson Júlio de Aguiar, Bruno S. Faiçal, Jó Ueyama, Glauco Carlos Silva, André Menolli [12];
- "Aplicação de Técnicas de Aprendizagem de Máquina para Diagnóstico de Depressão, Ansiedade e Estresse"; autores: Ana Laura Barros de Melo, André Luiz Firmino Alves [13];
- "*Predicting Anxiety, Depression and Stress in Modern Life using Machine Learning Algorithms*"; autores: Anu Priya, Shruti Garg, Neha Prerna Tigga [14];

- "*Medical Sentiment Analysis using Social Media: Towards building a Patient Assisted System*"; autores: Shweta Yadav, Asif Ekbal, Sriparna Saha, Pushpak Bhattacharyya [15];
- "Utilizando Análise de Sentimentos e SVM na Classificação de *Tweets* Depressivos"; autores: Omar Andres Carmona Cortes, Wesley Eduardo de Oliveira Melo [16];
- "*DP-Symptom-Identifier:* uma estratégia para classificar sintomas de depressão utilizando um conjunto de dados textuais na língua portuguesa"; autores: Vinicius Casani, Alinne C. Correa Souza, Rafael G. Mantovani, Francisco Carlos M. Souza [17];
- "*Machine Learning Algorithms for Depression Detection and Their Comparison*"; autores: Danish Muzafar, Furqan Yaqub Khan, Mubashir Qayoom [18];
- "*Depression Detection on Social Media Network (Twitter) using Sentiment Analysis*"; autores: Prof. S. J. Pachouly, Gargee Raut, Kshama Bute, Rushikesh Tambe, Shruti Bhavsar [19];
- "*Depression Detection from Social Media Data Using CNN and Linguistic Metadata Features*"; autores: J. Rachana, Pullaboina Sneha, Sathyam Laxmi Venkata Prasad, Resu Sai Chand Goud [20].

Os estudos abordavam a forma de elaboração da base de dados empregada nos artigos; alguns trabalhos, como de Jordan [11] e Aguiar [12], utilizaram bases de outros estudos ou grupos de pesquisa que haviam coletado material do *Twitter*, com bases já rotuladas; já o de Melo [13], utilizou uma base disponibilizada no *Kaggle*, aplicando testes psicológicos nas bases, como o DASS-42 (*Depression, Anxiety and Stress Scale*) e o TIPI (*Ten Iten Personality Inventory*) e o de Rachana [20] utilizou um dataset existente de posts do *Reddit*.

Outras pesquisas realizaram a criação do próprio dataset; nos artigos de Priya [14] e Zulfiker [10], foram desenvolvidos questionários próprios para a coleta de dados que posteriormente utilizaram testes psicológicos, como o DASS-21, ou escalas de medição de depressão, como o BDC (*Burns Depression Checklist*).

Em relação a artigos que desenvolveram a base de dados utilizando coleta de dados de redes sociais, os estudos de Cortes [16], Casani [17], Muzafar [18] e Pachouly [19] utilizaram o *Twitter* e o de Yadav [15] coletou postagens de usuários de um fórum médico (paciente.info).

Destaca-se neste aspecto o artigo de Casani [17], que possuía como objetivo "apresentar uma estratégia denominada *DP-Symptom-Identifier*", com a criação de um *dataset* possuindo sentenças que indiquem sintomas de depressão para auxiliar estudos em português brasileiro, uma vez que há uma lacuna na literatura. Para isso, criou-se uma base de dados com o auxílio de uma psicóloga, contendo 200 sentenças que remetem a sinais sintomáticos da depressão, rotuladas em comportamental, psíquico e fisiológico. Após isso ocorreu a coleta de mensagens por meio da API do *Twitter* e uma ferramenta desenvolvida em JAVA, de acordo com palavras-chave selecionadas das três categorias anteriormente citadas; e posteriormente esse material foi avaliado com o desenvolvimento de outra aplicação para que a psicóloga definisse os rótulos, separando os tweets entre os tipos de sintomas ou em nenhum sintoma.

A base de dados elaborada no estudo de Casani [17] será utilizada no presente trabalho para o desenvolvimento do modelo preditivo.

Após essa etapa inicial, os estudos realizaram a fase de pré-processamento dos elementos, aplicando normalização, com retirada de menções, *links*, *hashtags*, *emojis* e *emoticons*, símbolos e demais sentenças indesejadas, além de tratamento de abreviações. Nessa etapa também foi comum a remoção das *stop words*, palavras que são consideradas irrelevantes para o sentido da frase, como artigos e pronomes; porém nota-se que alguns estudos escolheram manter algumas dessas palavras, como é o caso de Pachouly [19], em que os pronomes na primeira pessoa foram preservados. Ainda no pré-processamento, processos de *stemming* [16] [17] [18] [19] ou *lemmatization* [11] [18] foram aplicados.

O artigo desenvolvido por Zulfiker utilizou outros métodos de seleção de características para eliminar informações redundantes ou desnecessárias, como *Select K-best features* (SelectKBest), *Minimum redundancy and maximum relevance* (mRMR) e o algoritmo Boruta; necessitando também de realizar um balanceamento das classes

utilizando *Synthetic Minority Oversampling Technique* (SMOTE), que consiste em criar novas instâncias sintéticas, exemplos gerados artificialmente pelo algoritmo, da classe minoritária para aumentar sua representação no dataset.

Em relação a técnicas de processamento de linguagem natural, foram utilizadas técnicas de extração de características de textos e de incorporação de palavras (*word embedding*), alguns artigos utilizaram uma ou mais de uma técnica, sendo citadas as seguintes: *Bag of Words* (BoW) nos artigos [11], [12], [17], [19]; *Term Frequency-Inverse Documento Frequency* (TF-IDF) em [11], [17], [19], *Word2Vec* em [20] e [18] para aplicação em SVM, *Naive Bayes* e *Random Forest*; *Glove* no artigo [20]; e, por fim, *Doc2Vec* em [18] para a abordagem em LSTM.

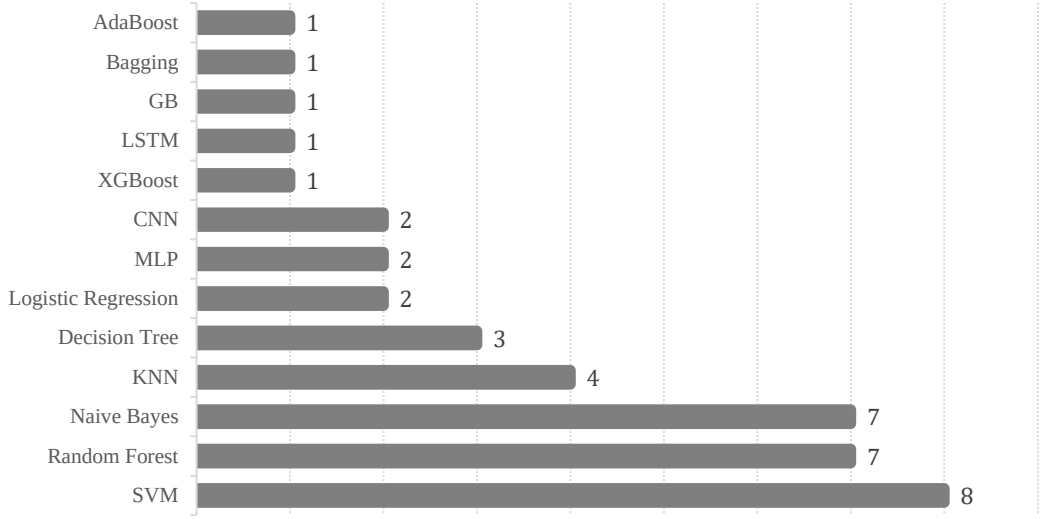
Após o pré-processamento e aplicação das técnicas de processamento de linguagem natural, as bases de dados foram separadas em amostras de treinamento e de testes. Nos artigos que informaram sobre essa etapa, a divisão se deu de variadas formas. Alguns utilizaram uma combinação de 70% de dados para treinamento e 30% para testes [14] [19], ou 80% para treinamento e 20% para testes [10] [18]; o estudo [11] apenas cita que empregou a função *train_test_split* da biblioteca *Sklearn* e o artigo [13] separou em 25.267 registros para treinamento e 13.606 para testes, aproximadamente 65% para treinamento e 35% para testes.

Houve, também, artigos [12] [16] [19] que utilizaram a validação cruzada, que objetiva evitar que a predição seja centrada em valores prejudiciais para a análise quando se trabalha com um conjunto de amostras reduzida, decompondo a base em treino e teste de forma aleatória K vezes, com a possibilidade de sobreposição.

A principal exceção nessa etapa de separação para treinamento e testes se deu em Casani [17] que utilizou duas bases separadas para treinamento e teste; porém para avaliar a qualidade do conjunto de dados, apenas o conjunto de treinamento descrito passou por uma validação cruzada, com 10 partições.

A etapa seguinte a ser realizada é a de elaboração dos modelos. Nos estudos, diversos algoritmos foram utilizados; abaixo, o gráfico demonstra a quantidade de vezes que cada um foi empregado, nota-se que alguns estudos utilizaram mais de um algoritmo:

Gráfico 01 - Quantidade de vezes que os algoritmos foram utilizados nos artigos selecionados



Fonte: Elaborado pelos autores

2.4. AVALIAÇÃO DE DESEMPENHO

Junto aos algoritmos, também foram apresentadas as métricas de desempenho obtidas nos estudos, comparando diversas combinações de algoritmo e técnicas de processamento nos aspectos de acurácia, *F1-score*, precisão, *recall*, AUC (*Area Under the Curve*), entre outros.

Essas métricas são medidas quantificáveis e mensuráveis utilizadas para analisar a performance de processos, ações ou estratégias. Elas são essenciais para avaliar modelos de predição, medindo o quão bom é o desempenho de um modelo em relação aos dados reais utilizados. Além disso, as métricas podem ser aproveitadas para comparar diferentes modelos e/ou algoritmos.

A Matriz de Confusão em si não é uma métrica, mas sua representação é utilizada para definir diversas medidas de desempenho. Esse tipo de matriz pode ser utilizado para classificações binárias ou para modelo de classificação multi-classes. A matriz de confusão é criada comparando a classe prevista e a classe real de um conjunto de dados [21]. A seguir, há um modelo de matriz de confusão:

		Classificação Real	
		Positivo	Negativo
Classificação Prevista	Positivo	VP	FP
	Negativo	FN	VN

Tabela 1 - Exemplo de Matriz de Confusão

Onde:

- VP = Verdadeiro Positivo, valor real e previsto positivos
- FP = Falso Positivo, o valor real é negativo, mas o previsto foi tido como positivo
- VN = Verdadeiro Negativo, valor real e previsto negativos
- FN = Falso Negativo, o valor real é positivo, mas o previsto foi tido como negativo

As principais métricas de desempenho podem ser geradas com os valores obtidos na matriz de confusão, como citado anteriormente.

A acurácia, uma das medidas mais populares, é responsável por calcular a porcentagem de acertos em relação ao total de entradas. Essa medida geralmente é utilizada quando as classes do conjunto de dados estão equilibradas e as previsões corretas, previsões verdadeiras positivas e verdadeiras negativas, possuem grande importância na análise [21]. A fórmula para cálculo da acurácia é a seguinte:

$$acurácia = \frac{VP + VN}{VP + FN + VN + FP}$$

Outra medida utilizada é a precisão, responsável por avaliar a taxa entre os verdadeiros positivos e todos os valores positivos; ou seja, é o número de previsões realizadas que são realmente corretas em relação ao total de previsões baseadas na classe positiva. De acordo com o livro de Sarkar e colaboradores [21] , “Um modelo com alta precisão identificará uma fração maior da classe positiva em comparação a um modelo com menor precisão. A precisão torna-se importante em casos onde estamos mais preocupados em encontrar o máximo possível da classe positiva, mesmo que a acurácia total seja reduzida” (tradução livre). Sua fórmula é dada por:

$$precisão = \frac{VP}{VP + FP}$$

O *recall*, ou sensibilidade, é a medida para avaliar a capacidade do modelo de prever resultados classificados como positivos, identificando a porcentagem de dados relevantes. É importante em casos onde é necessário detectar o "maior número possível de instâncias de uma classe específica, mesmo que isso aumente os falsos positivos" (tradução livre) [21]. A equação dessa medida é dada por:

$$recall = \frac{VP}{VP + FN}$$

Tem-se também o *F1-score*, métrica que calcula a média harmônica entre a precisão e o *recall* [21], sendo utilizado quando as duas medidas possuem muita importância para a análise. A fórmula do *F1-score* é:

$$f1 = 2 * \frac{precisão * recall}{precisão + recall}$$

Por sua vez, a especificidade é responsável por avaliar a capacidade do modelo de prever os resultados classificados como positivos, sendo utilizada em situações onde se faz necessário minimizar os falsos positivos. Ela indica quantas das classificações previstas estão certas por meio da equação:

$$especificidade = \frac{VN}{VN + FP}$$

Por fim, a medida AUC (*Area Under the Curve*) é a medida numérica da área resultante sob a curva do gráfico ROC (*Receiver Operating Characteristic Curve*). Esse gráfico permite a visualização e análise de um classificador, comparando diferentes classificadores e determinando qual é o mais adequado, ao observar a curva formada utilizando os eixos de taxa de verdadeiro positivo e de taxa de falso positivo. O desempenho de um classificador será melhor à medida que a curva ROC dele estiver mais próxima do canto superior esquerdo do gráfico. Abaixo, uma imagem representativa de um gráfico ROC:

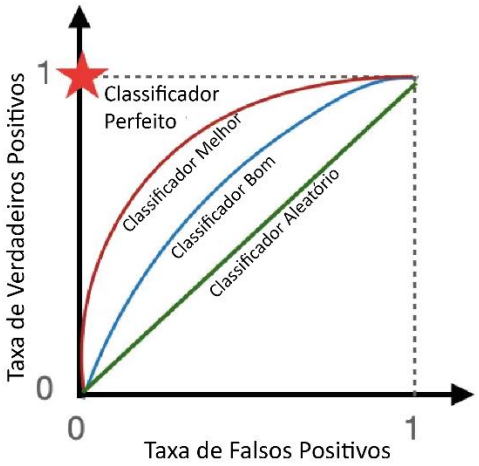
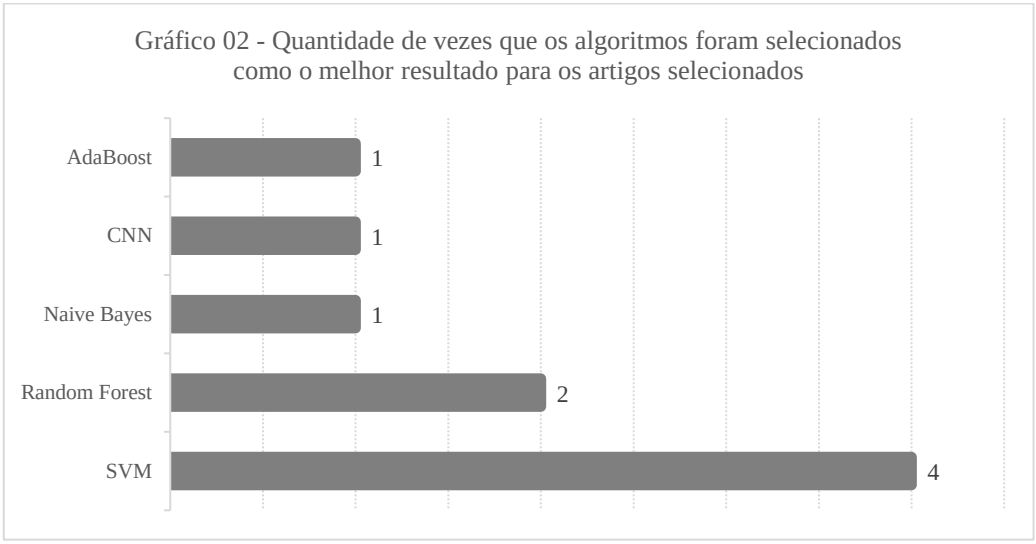


Figura 1 – Gráfico ROC adaptado de (Sípio, R., 2021) [22]

A medida AUC é a avaliação numérica da área resultante sob a curva do gráfico ROC. Expressa a capacidade de um classificador binário de distinguir entre classes, podendo variar de 0 a 1. Quanto mais próximo de 1 o valor, melhor é o desempenho do classificador, uma vez que isso indica que ele conseguiu distinguir corretamente entre as classes positivas e negativas [22].

Considerando essas métricas e os resultados dos estudos analisados, a utilização do algoritmo SVM se mostrou a mais indicada. Abaixo está apresentada a quantidade de vezes que cada algoritmo foi citado como o de melhor desempenho nos artigos:



Fonte: Elaborado pelos autores

Após essa revisão bibliográfica realizada, foi identificado que para uma boa elaboração de uma base de dados é de grande importância a presença de profissionais da área da saúde mental, além dos seguintes pontos nas demais etapas: o pré-

processamento é realizado utilizando diversas técnicas e de maneira cuidadosa e que os resultados de análise obtidos com melhor desempenho foram utilizando o algoritmo SVM (*Support Vector Machine*).

3. MÉTODO

3.1. SELEÇÃO DE DADOS

Inicialmente, uma das intenções do presente projeto era a elaboração de uma base de dados atualizados para treinamento e teste do modelo a ser desenvolvido; com postagens de usuários das redes sociais *Twitter* (atualmente, X) e *Reddit*, em que o teor do texto seria classificado com auxílio da área de psicologia; aos moldes do que foi realizado no estudo de Casani [17]. Porém, com recentes atualizações da API (Interface de Programação de Aplicação) dessas redes, em que a extração dos dados passou a ter um alto valor cobrado aos pesquisadores e também dificultou o desenvolvimento de robôs para raspagem de dados, não foi possível o desenvolvimento de uma base de dados própria neste estudo.

Com isso, *datasets* já existentes com postagens das duas redes sociais, e também já rotulados, foram utilizados no desenvolvimento da aplicação. Foram utilizados três *datasets* que coletaram os dados do *Twitter*, dois em inglês e um em português, e um *dataset* que possui dados do *Reddit*, em inglês.

As bases em inglês foram obtidas no site *Kaggle*, plataforma muito utilizada na comunidade de ciência de dados, contando com competições e disponibilização de *datasets*. Foram utilizados os seguintes:

- "*Depression: Twitter Dataset + Feature Extraction*" [23]. Este *dataset* conta com 20.000 *tweets* classificados em "depressivos" ou "não depressivos". Possui também dois *notebooks* disponibilizados pelo autor, em um deles há explicação e código sobre a extração de recursos de modelagem de tópicos, utilizando LDA (*Latent Dirichlet Allocation*), modelo estatístico generativo não supervisionado que explica um conjunto de observações por meio de grupos não

observados; e no outro *notebook* há o código relativo à extração de sentimento de emojis, classificando expressões em positivas, negativas ou neutras;

- "*Twitter Suicidal Data*" [24]. Este *dataset* é mais simples que o anterior, possuindo 9119 *tweets*, classificados de acordo com a intenção suicida identificada. Porém, não há maiores explicações. Foi encontrado o mesmo *dataset* disponibilizado desde 2020 no *Github*, com outro autor [25]. Optou-se pela utilização mesmo assim do conjunto de dados para maior variedade de fontes e aumento do número de *tweets* disponíveis para estudo.
- "*Dataset for Stress Analysis in Social Media*" [26]. Diferente dos demais citados acima, este *dataset* foi elaborado com posts do Reddit, de cinco *subreddits* (similar a comunidades) distintos, sendo eles: *anxiety*, *relationships*, *ptsd*, *domesticviolence*, *survivorsofabuse*. Diferencia-se também em relação ao tema de análise, enquanto os outros focavam em identificar depressão, este conjunto de dados focou em estresse.

Por sua vez, o dataset em português utilizado foi o do estudo realizado por Casani [17]. Para obtenção da base de dados, foi necessário entrar em contato com o autor, que encaminhou também explicação sobre colunas e valores da coluna label. Este conjunto de dados possui oito colunas ('id_tweet', 'ds_tweet', 'fisiologico', 'psiquico', 'comportamental', '*label*', 'validade_espec', 'dh_tweet') e oito valores possíveis para a coluna '*label*' (0 - Nenhum sintoma (sem indícios de depressão); 1 - Somente fisiológico; 2 - Somente comportamental; 3 - Somente psíquico; 4 - Psíquico e fisiológico; 5 - Comportamental e fisiológico; 6 - Comportamental e Psíquico; 7 - Todos os sintomas presentes). Possui 4051 registros.

O '*label*' foi criado pelos autores para facilitar o treinamento dos modelos, uma vez que trabalhar com as três colunas 'fisiologico', 'psiquico' e 'comportamental' estava oferecendo dificuldades no estudo realizado.

Essa classificação não binária se dá devido ao modo que a pesquisa foi conduzida, em que houve a classificação de sintomas em aspectos fisiológicos, comportamentais e/ou psíquicos, de acordo com validação e estudo de psicóloga que participou da

elaboração do *dataset*. A coluna 'validado_espec' indica se o tweet foi validado, rotulado e classificado pela especialista em psicologia.

Com as bases de dados estabelecidas, iniciou-se a etapa de pré-processamento. A base "*Twitter Suicidal Data*" [24] já contava com uma limpeza inicial realizada, não necessitando de intervenções neste primeiro momento. Ressalta-se que a linguagem utilizada para todos os scripts do presente trabalho será a linguagem *Python*, por sua grande difusão na comunidade de ciência de dados, o que proporciona diversas bibliotecas e módulos já existentes.

Já em relação à base "*Depression: Twitter Dataset + Feature Extraction*" [23], esta possuía *emojis*, caracteres especiais, menções a outros usuários e URLs, além de diversas colunas que não seriam utilizadas na análise, com dados sobre usuário e detalhes sobre o tweet, como data de criação e quantidade de vezes em que foi replicado ('*retweets*'). Abaixo um fragmento dos *tweets* existentes nesse *dataset*:

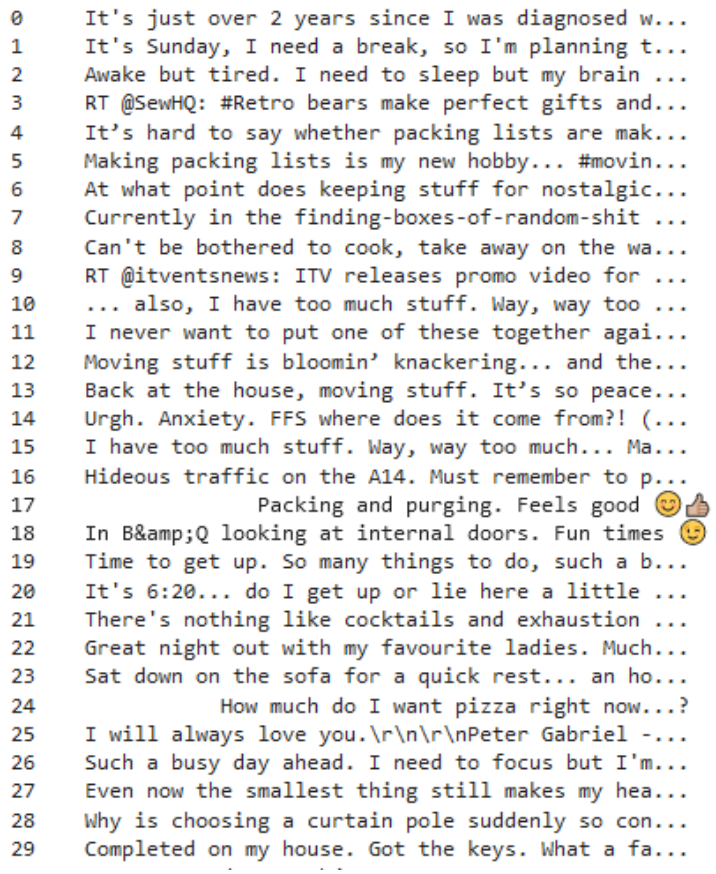


Figura 2 - Fragmento exibindo os primeiros 30 tweets do dataset. Foi destacado apenas o conteúdo textual dos tweets.

Primeiramente foi realizada uma limpeza das colunas. Para essa etapa, a biblioteca ‘pandas’ da linguagem *Python* foi utilizada. Este conjunto de dados contava com as colunas ‘0’, ‘post_id’, ‘post_created’, ‘post_text’, ‘user_id’, ‘followers’, ‘friends’, ‘favourites’, ‘statuses’, ‘retweets’ e ‘label’; como demonstrado abaixo:

	Unnamed: 0	post_id	post_created	post_text	user_id	followers	friends	favourites	statuses	retweets	label
0	0	637894677824413696	Sun Aug 30 07:48:37 +0000 2015	It's just over 2 years since I was diagnosed w...	1013187241	84	211	251	837	0	1
1	1	637890384576778240	Sun Aug 30 07:31:33 +0000 2015	It's Sunday, I need a break, so I'm planning t...	1013187241	84	211	251	837	1	1
2	2	637749345908051968	Sat Aug 29 22:11:07 +0000 2015	Awake but tired. I need to sleep but my brain ...	1013187241	84	211	251	837	0	1
3	3	637696421077123073	Sat Aug 29 18:40:49 +0000 2015	RT @SewHQ: #Retro bears make perfect gifts and...	1013187241	84	211	251	837	2	1
4	4	637696327485366272	Sat Aug 29 18:40:26 +0000 2015	It's hard to say whether packing lists are mak...	1013187241	84	211	251	837	1	1
5	5	637692793083817985	Sat Aug 29 18:26:24 +0000 2015	Making packing lists is my new hobby... #movin...	1013187241	84	211	251	837	1	1

Figura 3 - Primeiros cinco registros do dataset [24], demonstrando as colunas existentes e dados.

Para descartar as colunas que não seriam utilizadas, foi elaborado um novo *dataset*, de acordo com o código a seguir:

```
import pandas as pd

tweets_label = pd.DataFrame(df_tweets, columns = ['post_text', 'label'])
```

O comando acima resultou no seguinte novo *dataset*:

	post_text	label
0	It's just over 2 years since I was diagnosed w...	1
1	It's Sunday, I need a break, so I'm planning t...	1
2	Awake but tired. I need to sleep but my brain ...	1
3	RT @SewHQ: #Retro bears make perfect gifts and...	1
4	It's hard to say whether packing lists are mak...	1
...	...	...
19995	A day without sunshine is like night.	0
19996	Boren's Laws: (1) When in charge, ponder. (2) ...	0
19997	The flow chart is a most thoroughly oversold p...	0
19998	Ships are safe in harbor, but they were never ...	0
19999	Black holes are where God is dividing by zero.	0

20000 rows × 2 columns

Figura 4 - Novo dataset, retiradas as colunas que não seriam utilizadas na análise.

3.2. PRÉ-PROCESSAMENTO

Concluída a etapa de seleção de dados, realizou-se a remoção dos *emojis* observados no *dataset*. A principal biblioteca utilizada nesta etapa foi a 'emoji'. Com essa biblioteca, foi criada a função 'remove_emojis' que recebe um texto 'text' como argumento. Essa função percorre cada caractere na *string* 'text' de entrada e verifica se é um *emoji* utilizando a função 'emoji.emoji_count' da biblioteca, retornando 0 ou 1. Se o caractere não for um *emoji*, o que é verificado no filtro 'if not', o caractere é incluído na nova lista de caracteres, que é, ao fim, concatenada em uma única *string* sem os *emojis*.

Essa função foi aplicada na coluna 'post_text' do dataset 'tweets_sem_emoji' criado a partir do 'tweets_label':

```
import emoji


# Função para remover emojis de um texto
def remove_emojis(text):
    return ''.join([char for char in text if not emoji.emoji_count(char)])


# Aplicar a função a uma coluna do DataFrame
tweets_sem_emoji = pd.DataFrame(tweets_label)
tweets_sem_emoji['post_text'] = tweets_label['post_text'].apply(remove_emojis)
```

Para a remoção das URLs, foi utilizado o módulo 're', que fornece funções para trabalhar com expressões regulares, sequência de caracteres que fornece uma forma concisa para identificar segmentos de texto. Com isso, foi criada uma função 'remove_urls', que assim como a função 'remove_emojis', recebe como argumento um texto 'text':

```
import re


# Função para remover URLs de um texto usando regex
def remove_urls(text):
    return re.sub(r'http\S+|t.co\S+', '', text)
```

Esta função ‘remove_urls’ possui uma expressão regular para identificar as URLs, que procura por uma sequência de caracteres que inicie com “http” ou com “t.co”. O segmento “\S” corresponde a qualquer caractere que não seja um espaço em branco, e o “+” após indica que deve haver um ou mais caracteres. Por fim, o operador “|” representa um “ou” lógico, demonstrando que a sequência a ser identificada é a “http\S+” ou a “t.co\S+”. Por fim, a função ‘re.sub’ substitui a expressão por uma *string* vazia, como indicado no segundo argumento.

Novamente, a função foi aplicada na coluna ‘post_text’ do *dataset*, dessa vez nomeado de ‘tweets_sem_url’.

A terceira etapa da limpeza foi a remoção das menções. Como utilizado anteriormente na etapa de remover as URLs, também foi utilizado o módulo ‘re’, uma vez que menções são identificadas por um caractere “@” seguido por um conjunto de caracteres de letras e/ou números.

```
tweets_sem_mention = pd.DataFrame(tweets_sem_url)


# Remover mentions
tweets_sem_mention['post_text'] = tweets_sem_url['post_text'].str.replace('@[A-Za-
z0-9]+\s?', '', regex=True)
```

Nesta limpeza não foi criada uma função para a remoção, optando-se pelo método ‘str.replace()’, que substitui partes de *strings* em uma coluna. Nota-se que ele recebe três argumentos principais, sendo: a expressão regular; a *string* que irá substituir o texto, no caso uma *string* vazia; e por fim o argumento ‘regex’, que indica a utilização de uma expressão regular.

Para a remoção de caracteres especiais, algumas particularidades foram identificadas no conjunto de dados. Por exemplo, existiam *tweets* contendo “&amp” que não estão em URLs, como é o caso do tweet número 18 da Figura 1. Optou-se por substituir essa expressão por “&”, sendo posteriormente removido quando ocorresse a limpeza dos caracteres especiais.

Ao observar o *dataset*, um *tweet* foi identificado para verificar a eficiência da remoção de caracteres especiais:

	post_text	label
5248	"_Bot_: I have noticed you Cali (\•U•/)-kōhai.\r\n\r\n #senpaiBot" OH MY GOD IM CRYING LMFAO	1

Figura 5 - Tweet do dataset para verificação da remoção de caracteres especiais.

É possível observar que além dos caracteres a serem removidos nesta etapa, também existem expressões regulares de formatação presentes, com "\r" e "\n", que representam quebras de linhas. A limpeza dessas expressões foi realizada, então:

```
# Removendo \r
tweets_sem_especiais['post_text'] =
tweets_sem_especiais['post_text'].str.replace('\r', '', regex=True)

# Removendo \n
tweets_sem_especiais['post_text'] =
tweets_sem_especiais['post_text'].str.replace('\n', '', regex=True)
```

Outro aspecto encontrado foi o de pontos finais, vírgulas e hifens sem espaço após estes; o que ocasionaria, ao removê-los, a junção de palavras. Por exemplo, no *tweet* número 25 da Figura 1, ao remover as quebras de linha o texto resulta em "*I will always love you.Peter Gabriel - In Your Eyes*"; ao remover o ponto final, as palavras "*you*" e "*Peter*" se uniriam, o que não é o desejado. Com isso, novamente o módulo 're' foi utilizado, para trocar esses caracteres por espaço, não por *string* vazia:

```
# Substituir pontos finais que estão sem espaço
tweets_sem_especiais['post_text'] =
tweets_sem_especiais['post_text'].str.replace('(?<=[a-zA-Z0-9])\.(?=[a-zA-Z0-9])',
' ', regex=True)


# Substituir vírgulas que estão sem espaço
tweets_sem_especiais['post_text'] =
tweets_sem_especiais['post_text'].str.replace('(?<=[a-zA-Z0-9])\,(?=[a-zA-Z0-9])',
' ', regex=True)


# Substituir hifens que estão sem espaço
tweets_sem_especiais['post_text'] =
tweets_sem_especiais['post_text'].str.replace('(?<=[a-zA-Z0-9])\-(?=[a-zA-Z0-9])',
' ', regex=True)
```

Após esses ajustes, foi possível realizar a remoção dos caracteres especiais. Essa etapa também utilizou o módulo ‘re’ aplicado à coluna ‘post_text’, empregando uma expressão regular que corresponde a qualquer caractere que não seja uma letra, um dígito ou um espaço em branco:

```
# Remover caracteres especiais
tweets_sem_especiais['post_text'] =
tweets_sem_especiais['post_text'].str.replace('[^a-zA-Z0-9\s]', '', regex=True)
```

O símbolo “^” na expressão regular indica negação, ou seja, corresponde a caracteres que não estão dentro do conjunto apresentado posteriormente de letras, números e espaço em branco. Mais uma vez, os caracteres não desejados serão substituídos por uma *string* vazia.

Pode-se notar, ao retomar o *tweet* da Figura 4 que ocorreu a devida limpeza dos caracteres especiais:

	post_text	label
5248	Bot I have noticed you Cali Ukhai senpaiBot OH MY GOD IM CRYING LMFAO	1

Figura 6 - Tweet da Figura 4 após limpeza apresentada.

Nesta etapa do pré-processamento, foi discutido se os números seriam mantidos ou transformados em palavras para futura identificação na fase de extração de características dos textos. Neste *dataset* essa alteração foi feita de forma manual em etapa posterior. No entanto, em outros *datasets* utilizou-se uma expressão com o módulo regex, que será apresentada posteriormente

Por fim, todos os caracteres foram convertidos para minúsculas, utilizando a função '.str.lower()':

```
tweets_sem_especiais['post_text'] = tweets_sem_especiais['post_text'].str.lower()
```

O *dataset* em português, como citado anteriormente, possui diversas colunas. Abaixo um fragmento do conjunto:

	id_tweet	ds_tweet	fisiologico	psiquico	comportamental	label	validado_espec	dh_tweet
0	3696	dormi o dia inteiro tinha horas w meu desperta...	1	1	0	1	1	2019-10-09 02:32:41.000000
1	3885	já entendi que eu não sou importante tá bom	0	1	0	3	1	2019-10-12 12:15:04.000000
2	3888	chorando pq não sou importante pra ngm 👍 http...	0	1	0	3	1	2019-10-11 23:56:04.000000
3	3889	Sinto sempre que nao sou importante para certa...	0	1	0	3	1	2019-10-12 14:43:38.000000
4	8369	Tô indo pra escola com a força do ódio	0	0	0	0	1	2019-10-11 12:10:22.000000

Figura 7 - Fragmento contendo os 5 primeiros registros do dataset.

É possível observar que os textos contêm URLs, *emojis*, acentos e em outras ocorrências observou-se caracteres especiais e menções a outros usuários. Com isso, as etapas realizadas foram as mesmas aplicadas anteriormente, com adição de remoção de acentos.

Desta forma, criou-se um novo *dataset* contendo apenas as colunas 'ds_tweet', 'label' e 'validado_espec'. Após isso, foi aplicada a função de remoção de *emojis* 'remove_emojis' e depois a 'remove_url'. A remoção de menções e de caracteres especiais também foram realizadas do mesmo modo, incluindo as etapas de

transformação de “&amp” em “&”, remoção de ‘\r’ e ‘\n’ e transformação de pontos finais, vírgulas e hifens sem espaço em *string* com espaço vazio.

Para a remoção da acentuação, foi importado o módulo ‘unicode’ e utilizado do seguinte modo, convertendo caracteres Unicode com acento em seus equivalentes ASCII sem acentos:

```
coluna_alvo = 'ds_tweet'

# Utilizar a função unidecode para remover acentos da coluna
tweets_sem_acento[coluna_alvo] = tweets_sem_acento[coluna_alvo].apply(lambda x:
unidecode(x))
```

Por fim, os caracteres também foram convertidos em minúsculas com a função ‘.str.lower()’.

O último *dataset* a passar por esse primeiro processamento foi o do *Reddit* [26], e diferentemente dos demais, a limpeza foi realizada utilizando o programa Excel, com a ferramenta Power Query. Como primeira etapa, observou-se as colunas do arquivo. O *dataset* possuía 116 colunas, porém apenas duas foram aproveitadas: ‘*label*’ e ‘*text*’.

Já a tarefa de remoção de elementos desnecessários (caracteres especiais, *emojis* e pontuação) foi realizada por meio da criação de uma nova coluna para receber os dados limpos. A fórmula ‘Text.Select([Nome da coluna], {“A”..“z”, “0”..“9”})’ foi utilizada para extrair apenas os caracteres relevantes para este estudo, fazendo com que os elementos indesejados não fossem selecionados. ‘Text.Select’ é uma função capaz de extrair o tipo de caracteres especificado dentro da função:

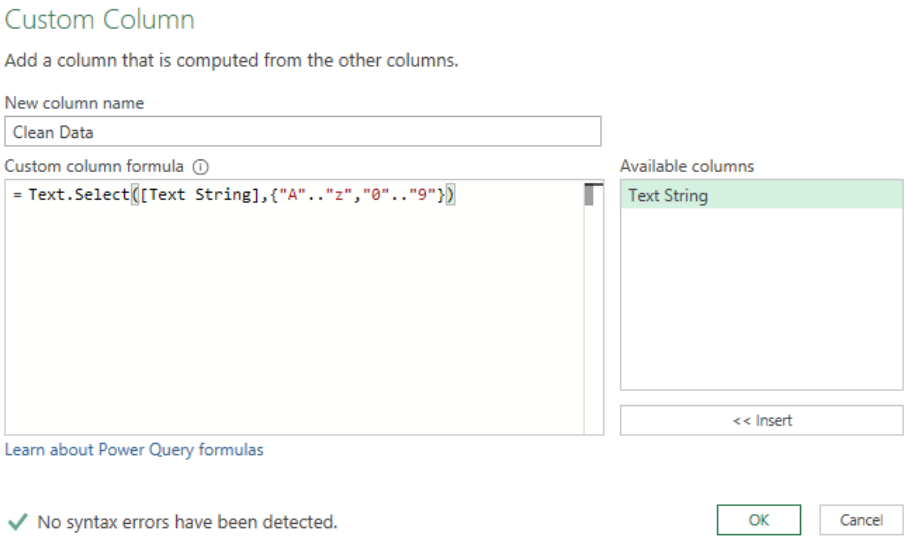


Figura 8 - Demonstração da aplicação da fórmula para limpeza dos dados no dataset contendo posts do Reddit

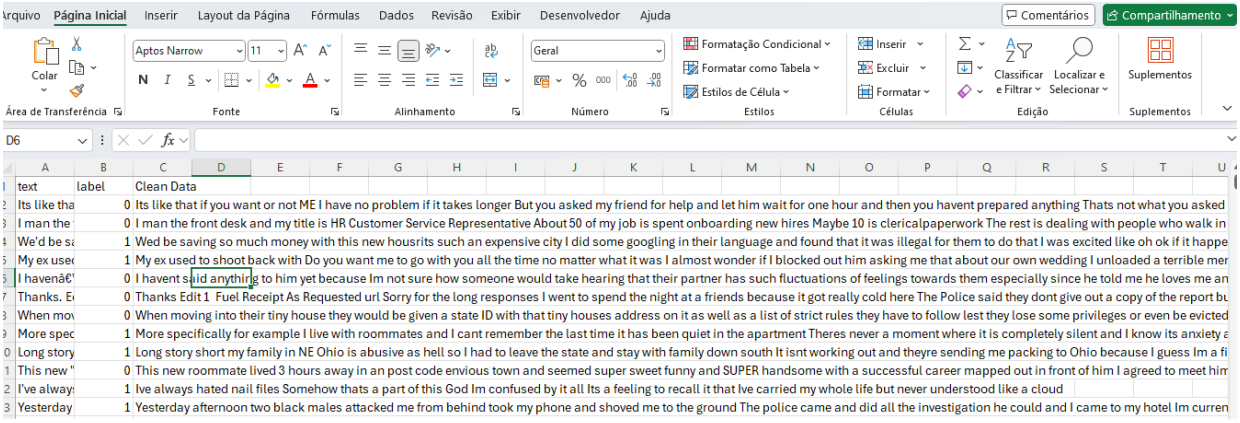


Figura 9 - Fragmento do dataset após limpeza

Como próxima etapa da normalização, foi realizado o tratamento de abreviações, gírias, *stopwords* e contrações.

Para os *datasets* em inglês, tanto de postagens do *Twitter* quanto do *Reddit*, foi construído um arquivo 'slang_dic.py' (APÊNDICE A) que contém o dicionário '*abbreviations*', elaborado com a utilização de um notebook presente no *Kaggle* [27] e também outras abreviações encontradas nos *datasets*. O código elaborado possui a seguinte forma:

```
from slang_dic import abbreviations as abb

def substituir_abreviacoes(texto):
    palavras = texto.split()
    palavras_substituidas = [abb[p] if p in abb else p for p in palavras]
    return ' '.join(palavras_substituidas)
```

A função ‘substituir_abreviacoes’ foi então aplicada à coluna do *dataset* que contém o texto, substituindo as abreviações e gírias encontradas pela sua forma completa, de acordo com o dicionário criado.

Já para o tratamento das *stopwords*, palavras que são consideradas irrelevantes para o sentido da frase, como artigos e pronomes, foi utilizada a classe ‘*stopwords’* da biblioteca ‘nltk’. Uma lista foi criada com as *stopwords* em inglês disponíveis nesta classe, da maneira abaixo:

```
lista_stopwords = set(stopwords.words('english'))
```

Algumas contrações foram adicionadas a esta lista, uma vez que ao retirar os caracteres especiais os apóstrofos foram eliminados e contrações como verbo *to be* + *not* (negação) foram perdidas.

```
# Variações adicionadas
variacoes = {'arent', 'didnt', 'doesnt', 'dont', 'hadnt', 'hasnt', 'havent',
'isnt', 'wasnt', 'werent', 'wouldnt'}

# Adiciona as variações ao conjunto original
lista_stopwords.update(variacoes)
```

Nos estudos analisados, o de Pachouly [19] escolheu preservar os pronomes em primeira pessoa. Esta opção também foi empregada no presente trabalho, e além dos pronomes de primeira pessoa, todos os outros também foram mantidos. Esta opção se deu, pois, ao analisar emoções, diversas vezes as pessoas envolvidas são importantes. A seguir está exibido o comando para as palavras que serão removidas da lista de *stopwords*, a fim de serem mantidas nos textos dos *datasets*:

```
# Palavras a serem removidas da lista de stopwords
remover = {"he", "hes", "her”, "hers", "herself", "him", "himself", "his", "i",
"it", "its", "itself", "me", "my", "myself", "our", "ours", "ourselves", "she",
"shes", "their", "theirs", "them", "themselves", "they", "we", "you", "your",
"yours", "yourself", "yourselves"}

# Remove as palavras do conjunto original
for palavra in remover:
    lista_stopwords.discard(palavra)
```

Nesta etapa todos os *datasets* com fonte do *twitter* tiveram o nome de suas colunas alterados. A coluna em que possuía o texto do tweet passou a ser denominada '*tweet'* e a coluna que indica o teor do tweet passou a se chamar '*label*'.

Com essa alteração, o código para a limpeza das *stopwords* inicialmente utiliza um método 'apply()' para aplicar a função anônima 'lambda', que possui um argumento 'x' representando cada postagem da coluna de texto, no caso '*tweet*'. O 'split()' dentro desta função 'lambda' divide o texto do campo em uma lista de palavras onde o delimitador é o espaço. Após isso, há uma verificação se a palavra analisada está ou não na lista de *stopwords*, representada por 'lista_stopwords'. Caso a palavra não esteja presente nesta lista, é incluída em uma nova lista que será concatenada novamente em uma única *string* pelo método 'join()'. O código que executa essa ação é o seguinte:

```
df_tweets_sem_stopwords['tweet'] = df_tweets_sem_stopwords['tweet'].apply(lambda
x:' '.join([word for word in x.split() if word not in (lista_stopwords)]))
```

De modo semelhante ao que foi elaborado para as abreviações, também foi criado um dicionário 'contraction_dic' (APÊNDICE B) para as contrações. Esse dicionário possui o dicionário '*contractions'*, elaborado pelos autores deste trabalho. Destaca-se neste momento uma questão encontrada, em que contrações como "*hes*" e "*shes*", respectivamente "*he's*" e "*she's*", podem ser resultantes tanto de "*he is/she is*" (verbo *to be* – ser/estar) quanto de "*he has/she has*" (verbo *have* – ter).

Não foi possível tratar esta questão manualmente, por exigir um alto nível de entendimento contextual e pelo tamanho dos *datasets*, e também não foi encontrado um modo de automatizar esta tarefa. Houve uma tentativa de utilização de inteligência artificial a fim de identificar de acordo com o contexto qual o verbo estaria sendo utilizado, mas também não apresentou resultados. Questões de processamento de linguagem natural por vezes exigem uma desambiguação contextual que requer redes neurais sofisticadas e treinamento extenso, o que não seria viável dentro do escopo deste trabalho.

Também foi considerado o treinamento futuro de dois modelos, um que considerasse todas as incidências como verbo *to be*, e outro como verbo *have*; porém essa opção foi descartada pois a ambiguidade ainda estaria presente, o que poderia gerar resultados conflitantes, induzindo a erros de análise; além de que isso adicionaria complexidade e gasto de recursos que não se justificam uma vez que o efeito dessa ambiguidade pode ser relativamente pequeno.

Com essas ponderações, optou-se por aceitar essas contrações como uma limitação do modelo atual, mantendo a consistência nos modelos.

O código utilizado para utilização do dicionário criado é muito similar ao de abreviações, também sendo aplicado posteriormente à coluna que contém o texto das postagens:

```
from contraction_dic import contractions as con

def substituir_contracoes (texto):
    palavras = texto.split()
    palavras_substituidas = [con[p] if p in con else p for p in palavras]
    return ' '.join(palavras_substituidas)
```

Com esses passos executados, o processo de normalização foi finalizado nos conjuntos de dados de língua inglesa; restando aplicação de técnicas de stemização ou de lematização para finalizar o pré-processamento desses *datasets*.

Em relação à base de dados em português, a limpeza de abreviações e remoção de *stopwords* foi similar à realizada anteriormente nos demais conjuntos; porém com adaptações lógicas à linguagem utilizada.

Um dicionário também foi elaborado para o tratamento das abreviações, com o nome de ‘slang_dic_pt’ (APÊNDICE C), utilizando listas elaboradas por Luana Simões [28] e pelo Programa Mais Você [29], além de ocorrências formuladas pelos autores do presente trabalho. Não houve a necessidade de um dicionário de contrações, uma vez que não existe no português contrações nos mesmos moldes das observadas na língua inglesa, que justificassem um tratamento diferenciado das palavras.

A função aplicada é semelhante à função ‘substituir_abreviacoes’; com a mudança para utilizar o dicionário em português; e executada na coluna de texto do *dataset*.

Em relação às *stopwords*, a classe ‘stopwords’ da biblioteca ‘nltk’ possui uma lista de palavras e português, sendo utilizado do seguinte modo:

```
lista_stopwords = set(stopwords.words('portuguese'))
```

Assim como para a língua inglesa, decidiu-se manter os pronomes no *dataset*, com isso uma lista de palavras a serem removidas de ‘lista_stopwords’ foi elaborada:

```
# Palavras a serem removidas da lista de stopwords
remover = {""dela", "delas", "dele", "deles", "ela", "elas", "ele", "eles", "eu",
"meu", "meus", "minha", "minhas", "nossa", "nosso", "nossas", "nossos", "nós",
"seu", "seus", "sua", "suas", "tu", "tua", "tuas", "você", "vocês", "não", "sem"}

# Remove as palavras do conjunto original
for palavra in remover:
    lista_stopwords.discard(palavra)
```

Também nesta lista de *stopwords* foi necessário remover as acentuações que já haviam sido retiradas na etapa anterior de limpeza. Para isso, o módulo Unicode foi utilizado novamente, aos moldes do executado para o *dataset* completo.

Após esses tratamentos na ‘lista_stopwords’, aplicou-se os mesmos comandos utilizados para os *datasets* em inglês no conjunto de dados em português; finalizando o processo de normalização desta base de dados também.

Ainda na etapa de pré-processamento, técnicas de *stemming* e *lemmatization* foram aplicadas.

Stemming, ou stemização, é utilizado para reduzir as palavras de um dado textual à sua forma base ao remover prefixos (casos mais incomuns) e sufixos, mantendo o radical das palavras. Um exemplo são as palavras “brincar” e “brincadeira”, que seriam reduzidas para “brinca” ou “brinc”.

Já *lemmatization*, ou lematização, é uma técnica que consiste em reduzir as palavras de um dado textual à sua forma de dicionário ou lema. O lema pode ser um substantivo, um verbo no infinitivo, um adjetivo ou um advérbio existente na gramática. Um exemplo é a palavra “maçãs”, que seria reduzida para “maçã” que é a forma base dessa palavra.

Os *datasets* foram separados em dois *dataframes* cada, um ‘df_stemming’ e um ‘df_lemmatization’. Os dados em inglês foram processados utilizando a biblioteca ‘nltk’; o ‘wordnet’ do ‘nltk.corpus’, que é um banco de dados lexical de inglês; e o ‘WordNetLemmatizer’ do ‘nltk.stem’, lematizador baseado em WordNet.

Para iniciar o processo de lematização, é necessário baixar o modelo de “*tagging*” de partes do discurso, que marca palavras de acordo com os respectivos tipos gramaticais (substantivos, verbos, adjetivos, etc).

```
nltk.download('averaged_perceptron_tagger')
```

Após isso, utilizou-se uma função para converter as tags do Penn Treebank, conjunto de *tags* utilizado pela biblioteca ‘nltk’, para as tags equivalentes do WordNet. As *tags* são códigos que representam partes do discurso; no caso do Penn Treebank, “J” é utilizado para adjetivos, “V” para verbos, “N” para substantivos e “R” para advérbios. Com isso, tem-se o seguinte código:

```
# Função para converter tags do Penn Treebank para WordNet
def penn_to_wordnet(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

O passo seguinte foi o de criar uma função de lematização com *POS Tagging*, ou *Part-of-Speech tagging*, em que as *tags* anteriormente citadas auxiliam no processo de lematização ao ajudar na desambiguação do significado de palavras iguais em diferentes contextos, melhorando a precisão do modelo.

A primeira etapa da função ‘lemmatize_with_pos_tag’ é a criação de uma instância do lemmatizador, utilizando o método ‘WordNetLemmatizer()’. Em seguida, os métodos ‘pos_tag()’ e ‘word_tokenize()’ são utilizados para tokenizar e taggear a sentença recebida de argumento, criando uma lista de tuplas onde cada tupla contém uma palavra e sua respectiva *tag*.

Na sequência, há um laço que realiza a conversão da *tag* do Penn Treebank para a *tag* correspondente do WordNet, utilizando a função ‘penn_to_wordnet’ criada anteriormente. Ainda dentro do laço, há uma verificação se a *tag* é válida no WordNet, o que realizará a lematização de acordo com a *tag* recebida. A palavra lematizada é, então, adicionada à lista ‘lemmatized_sentence’, que será transformada em uma *string* única, separada por espaços, com o método ‘join()’.

```
# Função de Lematização com POS tagging
def lemmatize_with_pos_tag(sentence):
    lemmatizer = WordNetLemmatizer()
    tagged_words = nltk.pos_tag(nltk.word_tokenize(sentence))
    lemmatized_sentence = []
    for word, tag in tagged_words:
        wn_tag = penn_to_wordnet(tag)
        if wn_tag is not None:
            lemma = lemmatizer.lemmatize(word, wn_tag)
        else:
            lemma = lemmatizer.lemmatize(word)
        lemmatized_sentence.append(lemma)
    return ' '.join(lemmatized_sentence)
```

Por fim, houve a execução dessa função em uma coluna que recebe identificador que foi lematizada. Na situação apresentada, foi denominada ‘tweet_lemmatized’ que é criada a partir da coluna ‘*tweet*’, do seguinte modo:

```
df_lemmatization['tweet_lemmatized'] =
df_lemmatization['tweet'].apply(lemmatize_with_pos_tag)
```

Para o processo de stemização, dois algoritmos distintos foram utilizados, de Porter e de Lancaster. O primeiro é o algoritmo mais popular atualmente para o

stemming, enquanto o segundo emprega um método mais agressivo, possuindo duas vezes mais regras de *stemming* que o método de Porter, apresentando tendências de reduzir excessivamente as palavras [30].

Iniciou-se este processo com a importação das classes 'PorterStemmer' e 'LancasterStemmer' de 'nltk.stem'. Após isso, duas funções foram criadas, muito semelhantes à função de lematização e também utilizando *POS tagging*:

```
# Função para stemizar um texto usando o PorterStemmer com POS tagging
def stem_with_porter_stemmer(sentence):
    stemmer = PorterStemmer()
    tagged_words = nltk.pos_tag(nltk.word_tokenize(sentence))
    stemmed_tokens = []
    for word, tag in tagged_words:
        wn_tag = penn_to_wordnet(tag)
        if wn_tag is not None:
            stemmed_token = stemmer.stem(word)
            stemmed_tokens.append(stemmed_token)
    return ' '.join(stemmed_tokens)


# Função para stemizar um texto usando o LancasterStemmer com POS tagging
def stem_with_lancaster_stemmer(sentence):
    stemmer = LancasterStemmer()
    tagged_words = nltk.pos_tag(nltk.word_tokenize(sentence))
    stemmed_tokens = []
    for word, tag in tagged_words:
        wn_tag = penn_to_wordnet(tag)
        if wn_tag is not None:
            stemmed_token = stemmer.stem(word)
            stemmed_tokens.append(stemmed_token)
    return ' '.join(stemmed_tokens)
```

Com as funções criadas, duas novas colunas foram geradas recebendo os dados da coluna '*tweet*' com as funções aplicadas. Do mesmo modo da lematização, dependerá do *dataset* o nome da nova coluna. No caso em apresentação, ocorreu da seguinte forma:

```
df_stemming['tweet_stemmed_porter'] =
df_stemming['tweet'].apply(stem_with_porter_stemmer)

df_stemming['tweet_stemmed_lancaster'] =
df_stemming['tweet'].apply(stem_with_lancaster_stemmer)
```

A seguir, um fragmento do *dataset* [23] após o pré-processamento, com etapas de normalização, lematização e stemização:

	tweet	label	tweet_lemmatized	tweet_stemmed_porter	tweet_stemmed_lancaster
0	it is 2 years since i diagnosed anxiety depression today i am taking moment reflect far i have come since	1	it be 2 year since i diagnose anxiety depression today i be take moment reflect far i have come since	is year i diagnos anxieti depress today i am take moment reflect far i have come	is year i diagnos anxy depress today i am tak mom reflect far i hav com
1	it is sunday i need break i am planning spend little time possible a14	1	it be sunday i need break i be plan spend little time possible a14	is sunday i need break i am plan spend littl time possibl a14	is sunday i nee break i am plan spend littl tim poss a14
2	awake tired i need sleep my brain ideas	1	awake tire i need sleep my brain idea	awak tire i need sleep brain idea	awak tir i nee sleep brain idea
3	retweet retro bears make perfect gifts great beginners get stitching octobers sew sale yay	1	retweet retro bear make perfect gift great beginner get stitch october sew sale yay	retweet retro bear make perfect gift great beginn get stitch octob sew sale yay	retweet retro bear mak perfect gift gre begin get stitch octob sew sal yay
4	it is hard say whether packing lists making life easier reinforcing much still needs movinghouse anxiety	1	it be hard say whether pack list make life easy reinforce much still need movinghouse anxiety	is hard say pack list make life easier reinforc much still need movinghous anxieti	is hard say pack list mak lif easy reinforc much stil nee movingh anxy
5	making packing lists my new hobby movinghouse	1	make pack list my new hobby movinghouse	make pack list new hobbi movinghous	mak pack list new hobby movingh

Figura 10 - Fragmento de dataset após aplicação de lematização e stemização

O tratamento dos dados do dataset em língua portuguesa necessitou de algumas alterações em relação aos de língua inglesa. No processo de lematização, a biblioteca utilizada não foi a 'nltk', mas sim a 'spaCy', que possui um suporte mais robusto para o português.

Essa etapa do processamento foi iniciada com a carga do modelo de língua portuguesa da '*spaCy*':

```
# Carregar o modelo da língua portuguesa
nlp = spacy.load("pt_core_news_sm")
```

Após isso, foi desenvolvida uma função de lematização, 'lemmatize_with_spacy', que recebe um texto de argumento. No início da função, cria-se um objeto 'doc', que representa o texto processado, contendo tokens analisados pelo modelo de *spaCy* carregado em 'nlp'. O passo seguinte é a utilização de uma *list comprehension* iterando sobre cada *token* em 'doc'; para cada *token*, o atributo 'lemma_' é acessado para obter a forma lematizada da palavra, que é então adicionada a uma lista chamada

‘lemmatized_tokens’. Por fim, o método ‘join()’ combina os elementos dessa lista em uma *string* separada por espaços. Observa-se a função criada a seguir:

```
# Função para lematizar um texto usando o spaCy
def lemmatize_with_spacy(text):
    doc = nlp(text)
    lemmatized_tokens = [token.lemma_ for token in doc]
    return ' '.join(lemmatized_tokens)
```

Com essa função criada, a lematização é aplicada da mesma forma que nos *datasets* anteriores, com a criação de uma coluna ‘*tweet_lemmatized*’, que recebe o texto lematizado da coluna ‘*tweet*’.

Para a stemização optou-se pela utilização da biblioteca ‘nltk’ novamente, pois oferece diversos algoritmos e simplicidade para o processo de *stemming*. Dessa vez, a classe importada foi a ‘RSLPStemmer’. A função elaborada para esse processamento possui uma lógica muito similar à função ‘lemmatize_with_spacy’:

```
# Inicializar o stemmer para o português
stemmer = RSLPStemmer()


# Função para stemizar
def stem_port(text):
    tokens = nltk.word_tokenize(text)
    stems = [stemmer.stem(token) for token in tokens]
    return ' '.join(stems)
```

Por fim, a função foi aplicada na coluna '*tweet*' e salva em uma nova coluna '*tweet_stemmed*'. Abaixo um fragmento de como ficou o *dataset* em português após o pré-processamento:

	tweet	label	validado_espec	tweet_lemmatized	tweet_stemmed
0	dormi dia inteiro horas w meu despertafor tocava aj eu pensavabnossa to tristebsou pessima stanbdo chittaphon ai lembrava sabadonnossa depressao total	1	1	dormi dia inteiro hora w meu despertafor tocar aj eu pensavabnossar to tristebsar pessimo stanbdo chittaphon ai lembrar sabadonnossa depressao total	dorm dia int hor w meu despertaf toc aj eu pensavabnoss to tristebs pess stanbd chittaphon ai lembr sabadonnoss depressa total
1	entendi eu nao importante ta bom	3	1	entendi eu nao importante ta bom	entend eu nao import ta bom
2	chorando porque nao importante pra ninguem	3	1	chorar porque nao importante pra ninguem	chor porqu nao import pra ning
3	sinto sempre nao importante certas pessoas maneira eu queria	3	1	sinto sempre nao importante certo pessoa maneira eu querer	sint sempr nao import cert pesso man eu quer
4	to indo pra escola forca odio	0	1	to ir pra escola forca odio	to ind pra escol forc odi
5	minha mae disse eu namoro escondido tao escondido eu sei	0	1	meu mae dizer eu namoro esconder tao esconder eu saber	minh mae diss eu namor escond tao escond eu sei

Figura 11 - Fragmento de dataset após aplicação de lematização e stemização

3.3. EXTRAÇÃO DE CARACTERÍSTICAS

A próxima etapa envolve a aplicação de técnicas de extração de características de texto e incorporação de palavras. Observou-se nos estudos analisados que as formas mais utilizadas para isso são a *Bag-of-words* (BoW), técnica capaz de identificar as palavras mais significativas de um texto ao verificar quantas vezes tais palavras são utilizadas, atribuindo um valor numérico para cada uma usando a frequência como parâmetro; e a técnica *Term Frequency-Inverse Document Frequency* (TF-IDF), capaz de identificar as palavras mais importantes e de atribuir um valor numérico para cada uma. Diferentemente do BoW, a frequência de uma palavra no TF-IDF não é o principal parâmetro utilizado, mas sim o peso de cada termo.

Nesta fase do projeto, ocorreu a limpeza dos números, após decisão de eliminá-los e não converter em palavras.

Ao buscar dígitos numéricos no dataset "*Twitter Suicidal Data*" [25], foram encontradas diversas colunas com o mesmo conteúdo, como demonstrado abaixo:

	tweet	label	tweet_lemmatized	tweet_stemmed_porter	tweet_stemmed_lancaster
1794	i want die	1	i want die	i want die	i want die
7351	i want die	1	i want die	i want die	i want die
7370	i want die	0	i want die	i want die	i want die
7389	i want die	1	i want die	i want die	i want die
7395	i want die	1	i want die	i want die	i want die
...	...	...	...	...	...
8899	i want die	0	i want die	i want die	i want die
8900	i want die	0	i want die	i want die	i want die
8901	i want die	0	i want die	i want die	i want die
8902	i want die	0	i want die	i want die	i want die
8903	i want die	0	i want die	i want die	i want die

199 rows × 5 columns

Figura 12 - Fragmento de dataset demonstrando haver 199 linhas com o mesmo conteúdo

Essa repetição poderia causar redundância e introduzir um viés nos dados do estudo, dificultando o aprendizado do modelo e resultando em uma menor capacidade de generalização. Com isso, as linhas duplicadas foram removidas e apenas uma das ocorrências foi mantida. Optou-se por manter uma linha que possuía *label* = 1; pois a frase após os processos de limpeza claramente indica uma tendência suicida.

Optou-se pela remoção dos números após unir os dois *datasets* do *twitter* que possuíam a língua inglesa, o "*Twitter Suicidal Data*" [25] e o "*Depression: Twitter Dataset + Feature Extraction*" [23]. Como citado anteriormente, a remoção dos números se deu de maneira manual, por meio do seguinte processo: primeiro, as bases foram concatenadas, uma vez que suas colunas já estavam nomeadas da mesma forma; após isso, uma lista com todas as palavras presentes no *dataset* concatenado foi gerada e ordenada.

Ao realizar essa ordenação, notou-se que haviam palavras em outras línguas no conjunto de dados, como árabe, tailandês, coreano, russo, dentre outras que não utilizam o alfabeto latino.

Com isso, listas foram criadas contendo palavras em outros alfabetos, números e palavras identificadas que não faziam sentido, como 'aa', 'aaa', 'aaaaa', 'aaaaaaaaadddd', 'aaaaaaaah' e outras. Abaixo, um exemplo dessas listas:

```
words_to_remove_02 = ['00', '090', '09251982', 'より', 'урьте', '0' ,'والبنت_o', '0mins', '0ne', '0say
 '10year', '11', '110', '111', '1111', '1112', '1115', '1117', '112', '11228', '1127', '11293', '11
 '115lbs', '116', '1160', '118', '11am', '11ml', '11mobile', '11pm', '11th', '11yrs', '12', '120',
 '1213', '1215', '122', '1221am', '1230', '1234', '1235', '125', '1250hr', '125130',
 '1253', '1255', '125k', '127', '129', '12am', '12i', '12pack', '12pagi', '12th',
 '12v', '12yr' ,'12yrs', '13', '130', '130927', '130ish' ,'130quot', '1314' '133',
 '135', '1354', '138', '139', '13am', '13ish', '13k', '13studied', '13th',
 '13yearold', '13yrs', '14', '140' ,'1409', '141' ,'1415throughout', '1416',
 '1417', '1419', '144', '1440' ,'1456' ,'148' ,'1482922616', '14c', '14f', '14i',
 '14th', '15', '150', '153', '155', '157', '15847usd' ,'15apollonah', '15g' ,'15i',
 '15k', '15m', '15mg','15min' ,'15th', '15yearold', '15yo', '16', '160', '1617',
 '165', '1659' ,'165lbs', '16f', '16i', '16last', '16m', '16reb', '16th',
 '16yearold' ,'16yo' ,'17', '170' ,'1718' ,'1727272' ,'173', '1799', '17ast', '17f',
 '17hrs', '17k', '17month', '17th' ,'17wasvsdal', '17yearold', '17yo', '17yrs',
 '18', '180', '182' ,'1829', '1838255', '1850' ,'1850s', '1855', '1872', '1875',
 '18c' ,'18i', '18k', '18m', '18meme' ,'18months' ,'18on' ,'18th', '18yearold',
 '18yoi' ,'19' ,'1912', '1914', '1928', '1932' ,'1950', '1950s', '1953', '1955',
 '1956', '1960crooks', '1960s', '1962', '1968' ,'1970s' ,'1972', '1973' ,'1973the']
```

Figura 13 - Exemplo de lista criada com palavras a serem removidas do conjunto de dados

A remoção destas palavras foi feita utilizando um laço que itera sobre cada palavra da lista, que são removidas quando encontradas nas colunas especificadas do *dataset*:

```
for word in words_to_remove_02:
    df_limpar[columns_to_clean] = df_limpar[columns_to_clean].replace(word, '',
regex=True)
```

Após essa limpeza feita de forma manual, que contou com revisão humana para a substituição das palavras, as técnicas de extração de características de texto começaram a ser aplicadas. Porém, devido ao tamanho do *dataset*, que no momento contava com 28.650 instâncias, os algoritmos utilizados para o BoW e para o TF-IDF se mostraram incapazes de ser processados pelos computadores da equipe.

Diante deste desafio, soluções alternativas foram buscadas para evitar a redução do tamanho do *dataset* e consequente perda de dados, o que poderia prejudicar a análise. Foram exploradas opções como o contato com instituições para utilizar computadores com maior capacidade de processamento e até mesmo a tentativa de implementação de programação distribuída. No entanto, a única solução que demonstrou resultados efetivos foi a redução do tamanho do conjunto de dados.

Com isso, foram selecionadas 10.000 amostras do conjunto, de forma aleatória, de acordo com o código abaixo:

```
tweets_sample = tweets_en.sample(n = 10000, random_state = 42)
```

Para execução do BoW, foi utilizada a classe '*CountVectorizer*' do módulo '*feature_extraction.text*' da biblioteca '*scikit-learn*'.

Essa classe foi então atribuída à variável 'vectorizer', criando um objeto que será utilizado para transformar o texto de cada coluna e instância em uma matriz de contagem. A aplicação deste objeto será nas três colunas geradas: de lematização, *stemming* com Porter e *stemming* com Lancaster:

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()

tweets_sample["tweet_lemmatized_bow"] =
vectorizer.transform(tweets_sample["tweet_lemmatized"]).toarray().tolist()

tweets_sample["tweet_st_porter_bow"] =
vectorizer.transform(tweets_sample["tweet_stemmed_porter"]).toarray().tolist()

tweets_sample["tweet_st_lancaster_bow"] =
vectorizer.transform(tweets_sample["tweet_stemmed_lancaster"]).toarray().tolist()
```

Após a criação destas colunas com o BoW aplicado, foi realizada uma remoção nas colunas contendo textos, uma vez que no treinamento do modelo não seriam mais utilizadas:

```
colunas_drop = ["tweet", "tweet_lemmatized", "tweet_stemmed_porter",
"tweet_stemmed_lancaster"]

tweets_sample.drop(colunas_drop, axis = 1, inplace = True)
```

Com isso, o *dataset* de *tweets* em inglês passou a ter a seguinte forma:

	label	tweet_lemmatized_bow	tweet_st_porter_bow	tweet_st_lancaster_bow
17884	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
10186	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
17756	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
15936	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
12340	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

Figura 14 - Estrutura e dados após aplicação do Bag-of-words

Enquanto o método do BoW consiste na criação de um *array* contendo a contagem da incidência das palavras do texto no *dataset*; o TF-IDF é um cálculo estatístico que une duas partes: TF, que se refere à frequência do termo, e IDF que significa frequência inversa do documento.

Em outras palavras, o TF é calculado como a razão entre o número de vezes que a palavra aparece em um texto e o número total de palavras no documento. Já o IDF, calcula o peso de palavras raras em todo o texto, com isso, palavras que possuem pouca incidência possuem um valor maior de IDF. Esses cálculos são realizados da seguinte forma:

$$TF = \frac{Nº\ de\ ocorrências\ da\ palavra\ no\ documento}{Nº\ total\ de\ palavras\ no\ documento}$$

$$IDF(p) = \log\left(\frac{N}{1 + DF(p)}\right)$$

Onde:

- p = termo ou palavra para a qual o IDF está sendo calculado;
- N = total de instâncias no *dataset;*
- DF(p) = número de instâncias em que o termo *p* aparece.

A multiplicação dessas duas equações, ou seja, TF x IDF, resulta no TF-IDF.

Nota-se que o cálculo realizado apresenta um maior custo computacional que o BoW. Com isso, mesmo na amostra de 10.000 instâncias não foi possível executar esta técnica, resultando em erro de memória, em que não foi possível alocar este recurso de maneira suficiente para a execução do *array* resultante desta técnica. Foi realizado um teste para separar as duas etapas, mas também não obteve sucesso.

Desse modo, os conjuntos de dados em inglês para o *Twitter* apenas apresentarão modelos de predição com a técnica *Bag-of-words*.

Para o *dataset* com dados do *Reddit*, também ocorreu uma limpeza anterior à aplicação das técnicas de extração de características; para limpeza de números, palavras com números e *underlines* que foram identificados no conjunto de dados.

Essa última limpeza foi realizada utilizando o módulo 're' novamente e com a criação de uma função 'clean_text', que recebe como argumento um texto. Abaixo, o código e as expressões regulares utilizadas:

```
# Função para limpeza do texto

def clean_text(text):
    cleaned_text = re.sub(r'\b\w*\d\w*\b', '', text)  # Remover palavras com
números
    cleaned_text = re.sub(r'\d+', '', cleaned_text)    # Remover todos os números
    cleaned_text = re.sub(r'_', '', cleaned_text)      # Remover underlines
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)   # Remover espaços extras
    cleaned_text = cleaned_text.strip()                # Remover espaços no início
e no fim
    return cleaned_text
```

Essa função foi aplicada às colunas 'post_lemmatized', 'posts_stemmed_porter' e 'posts_stemmed_lancaster' por meio do método 'apply()', criando novas colunas limpas como demonstrado a seguir:

```
df_reddit_posts_lemmatized['cleaned_posts'] =
df_reddit_posts_lemmatized['posts_lemmatized'].apply(clean_text)

df_reddit_posts_stemmed['cleaned_posts_porter'] =
df_reddit_posts_stemmed['posts_stemmed_porter'].apply(clean_text)

df_reddit_posts_stemmed['cleaned_posts_lancaster'] =
df_reddit_posts_stemmed['posts_stemmed_lancaster'].apply(clean_text)
```

Por fim, as colunas anteriores foram removidas e o processo para a técnica BoW pode ser aplicado. As mesmas etapas e funções utilizadas anteriormente foram empregadas para este *dataset*.

Do mesmo modo ocorrido com a amostra de *tweets* do conjunto de dados em inglês, a técnica de TF-IDF também não foi possível de executar para o *dataset* do *Reddit*;

ocorrendo o mesmo erro de memória em que os computadores não conseguiram alocar recurso suficiente para elaboração do *array*.

Para finalização desta etapa, o *dataset* de *tweets* em português brasileiro passou pela mesma limpeza realizada para remoção de números, palavras com números e outras ocorrências que o ocorrido no conjunto do *Reddit*; utilizando a mesma função 'clean_text'. A aplicação da técnica BoW também se deu da mesma forma, porém este *dataset* possui apenas duas colunas, resultando no seguinte:

	label	validado_espec	tweet_lemmatized_bow	tweet_stemmed_bow
0	1	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1	3	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2	3	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3	3	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
4	0	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

Figura 15 - Fragmento do dataset em português após aplicação do Bag-of-words

Ao contrário dos demais conjuntos de dados, a aplicação da técnica TF-IDF foi bem sucedida. Para isso, foi importada a classe 'TfidfVectorizer', da biblioteca 'sklearn', módulo 'sklearn.feature_extraction.text'. Após isso, cria-se uma variável 'vectorizer', transformada em objeto ao receber a classe 'TfidfVectorizer()'. Este objeto é então aplicado a cada coluna do *dataset*, como demonstrado abaixo:

```
vectorizer = TfidfVectorizer()

tweets_pt['tweet_lemmatized_tfidf'] =
vectorizer.fit_transform(tweets_pt['tweet_lemmatized']).toarray().tolist()

tweets_pt['tweet_stemmed_tfidf'] =
vectorizer.fit_transform(tweets_pt['tweet_stemmed']).toarray().tolist()
```

O *dataset* ficou com o seguinte aspecto após a aplicação do TF-IDF e a retirada das colunas de texto:

	label	validado_espec	tweet_lemmatized_tfidf	tweet_stemmed_tfidf
0	1	1	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1	3	1	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
2	3	1	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3	3	1	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
4	0	1	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...

Figura 16 - Fragmento do dataset em português após aplicação do TF-IDF

3.4. MODELAGEM

Após todos os *datasets* estarem com as etapas de pré-processamento finalizadas e com as técnicas de extração de características aplicadas, foi realizado o treinamento do modelo. Como citado anteriormente, os estudos analisados indicam o algoritmo SVM como o que apresenta melhor desempenho nesta tarefa.

Esse algoritmo é empregado em tarefas de classificação e análise de regressão. Seu objetivo é reconhecer os padrões de dados e prever em qual das duas classes eles melhor se encaixam. As classes são divididas por um hiperplano, uma linha divisória que busca manter as classes o mais separadas possíveis.

Em relação ao desenvolvimento dos modelos, inicialmente diversas bibliotecas e classes foram importadas; tanto para separação de amostras em treino e teste, quanto para o próprio algoritmo SVM e também para as métricas de análise de desempenho:

```
from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix

import numpy as np
```

Após isso, foi realizada a conversão dos elementos das colunas dos *datasets* para tipos numéricos, em dados gerados pela técnica BoW para ‘inteiro’ e nos gerados por TF-IDF para ‘*float*’, uma vez que estavam com tipo de *string* o que impede o processamento pelo algoritmo.

Para cada modelo, optou-se por remover as outras colunas do *dataset*, ou seja, por exemplo, no modelo combinado de lematização e *Bag-of-words*, apenas foram mantidas as colunas '*label*' e a coluna relativa a essa combinação, como 'tweet_lemmatized_bow':

	label	tweet_lemmatized_bow
0	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
3	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
4	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

Figura 17 - Fragmento do dataset de tweets em inglês, neste caso para processamento do modelo que combina lematização e Bag-of-words

Em seguida, o campo contendo a lista foi transformado em várias colunas. Cada nova coluna representa uma palavra, uma vez que a técnica de extração de características de texto cria a lista com base no número de palavras distintas no dataset. Quando uma determinada palavra aparece, o algoritmo adiciona um na posição correspondente a essa palavra na lista.

```
# Dividindo a coluna 'tweet_lemmatized_bow' em várias colunas
tweets_bow_lem = pd.concat([tweets_bow_lem.drop('tweet_lemmatized_bow', axis=1),
pd.DataFrame(tweets_bow_lem['tweet_lemmatized_bow'].tolist())], axis=1)

# Renomeando as novas colunas conforme necessário
new_col_names = [f'elemento_{i}' for i in range(tweets_bow_lem.shape[1])]
tweets_bow_lem.columns = new_col_names

# Renomear elemento_0 para label
tweets_bow_lem = tweets_bow_lem.rename(columns={'elemento_0': 'label'})
```

Cabe ressaltar que uma vez que cada coluna representa uma palavra distinta é possível saber com facilidade quantas palavras foram identificadas em cada *dataset*.

Dataset		Número de Palavras
Tweets em Inglês (amostra)	Lematização	16.723
	Stemização	16.723
Tweets em português	Lematização	6.093
	Stemização	4.547
Reddit - inglês	Lematização	10.329
	Stemização	Porter: 8.480 Lancaster: 7.292

Tabela 2 - Quantidade de Palavras em cada dataset a ser utilizado para os modelos de previsão

A coluna '*label*', contendo a classificação que se deseja prever, é a variável dependente, ou variável alvo, e as demais colunas são as variáveis independentes, que influenciam ou explicam o resultado da variável alvo. Deste modo, criou-se as variáveis 'X' e 'y' utilizadas na separação de amostra de treino e teste, sendo nomeadas de acosto com o modelo a ser criado:

```
# Listar todas as colunas exceto 'label'
colunas_bow_lem = [col for col in tweets_bow_lem.columns if col != 'label']


# Criar X (excluindo a coluna 'label') e y – bl: bow e lematização
X_bl = tweets_bow_lem[colunas_bow_lem]
y_bl = tweets_bow_lem['label']


# Dividindo os dados em conjunto de treinamento e teste (80% para treinamento e
20% para teste)
X_bl_train, X_bl_test, y_bl_train, y_bl_test = train_test_split(X_bl, y_bl,
test_size=0.2, random_state=42)
```

Após a divisão do *dataset* em conjuntos de treinamento e teste, realizou-se a inicialização do classificador SVM e treinamento do modelo. Depois, previsões foram feitas no conjunto de testes, para identificar como o modelo havia aprendido.

3.5. AVALIAÇÃO

Para avaliar o desempenho dos modelos preditivos, utilizamos uma série de métricas de avaliação padrão na área de aprendizado de máquina, que nos permitiram verificar a eficácia dos modelos de forma abrangente:

- Acurácia: calcula a proporção de previsões corretas em relação ao total de previsões realizadas. É útil especialmente quando as classes no conjunto de dados estão balanceadas, proporcionando uma visão geral do desempenho do modelo.
- Precisão: avalia a proporção de verdadeiros positivos em relação ao total de instâncias classificadas como positivas. Esta métrica é crucial para minimizar falsos positivos, sendo importante em contextos onde a classificação errônea de positivos deve ser evitada.

- *Recall* (Sensibilidade): mede a capacidade do modelo de identificar corretamente as instâncias da classe positiva. É fundamental quando a maximização da detecção de verdadeiros positivos é prioritária, mesmo que isso aumente os falsos positivos.
- *F1-score*: Calcula a média harmônica entre precisão e *recall*, oferecendo um equilíbrio entre ambas as métricas. É particularmente útil quando se busca um compromisso entre precisão e *recall*, fornecendo uma única métrica que considera ambos.

Além dessas métricas, a Matriz de Confusão foi utilizada para fornecer uma visualização detalhada do desempenho do modelo, mostrando a contagem de verdadeiros positivos (VP), falsos positivos (FP), verdadeiros negativos (VN), e falsos negativos (FN). Esta matriz ajuda a entender melhor como o modelo lida com cada classe e onde ocorrem os erros.

Para uma avaliação mais visual e detalhada do desempenho, a Área sob a Curva ROC (AUC-ROC) também foi calculada. Essa métrica mede a capacidade do modelo de distinguir entre classes, com valores mais próximos de 1 indicando um desempenho superior. A curva ROC é plotada utilizando as taxas de verdadeiros positivos e falsos positivos, ajudando a visualizar a eficácia do classificador.

As métricas foram calculadas utilizando a função classification_report da biblioteca *sklearn* no *Python*. Abaixo estão os últimos trechos de código que ilustram a implementação dessas etapas de avaliação:

```
# Inicializando o classificador SVM
svm_classifier_bl = SVC(kernel='linear')

# Treinando o modelo SVM
svm_classifier_bl.fit(X_bl_train, y_bl_train)

# Fazendo previsões no conjunto de teste
y_bl_pred = svm_classifier_bl.predict(X_bl_test)

# Calculando as métricas do modelo
accuracy_bl = accuracy_score(y_bl_test, y_bl_pred)
precision_bl = precision_score(y_bl_test, y_bl_pred, average='weighted')
recall_bl = recall_score(y_bl_test, y_bl_pred, average='weighted')
f1_bl = f1_score(y_bl_test, y_bl_pred, average='weighted')

# Calcular a matriz de confusão
conf_matrix_bl = confusion_matrix(y_bl_test, y_bl_pred)

# Exibir as métricas e a matriz de confusão
print(f"Acurácia do modelo SVM: {accuracy_bl}")
print(f"Precisão: {precision_bl:.2f}")
print(f"Recall: {recall_bl:.2f}")
print(f"F1-Score: {f1_bl:.2f}\n")
print("Matriz de Confusão:")
print(conf_matrix_bl)
```

Na tabela a seguir, os resultados das métricas de desempenho de acordo com as características do modelo:

Dataset	Técnica de Extração de Características	Stemização/Lematização	Métricas			
			Acurácia	Precisão	Recall	F1-Score
Tweets em Inglês (amostra)	BoW	Lematização	0,724	0,73	0,72	0,72
		Stemização Porter	0,718	0,72	0,72	0,71
		Stemização Lancaster	0,705	0,71	0,70	0,70
Tweets em português	BoW	Lematização	0,800	0,80	0,80	0,80
		Stemização	0,792	0,79	0,79	0,79
	TF-IDF	Lematização	0,808	0,81	0,81	0,80
		Stemização	0,800	0,80	0,80	0,79
Reddit - inglês	BoW	Lematização	0,690	0,69	0,69	0,69
		Stemização Porter	0,690	0,69	0,69	0,69
		Stemização Lancaster	0,680	0,68	0,68	0,68

Tabela 3 - Resultados das Métricas de Desempenho

4. RESULTADOS E DISCUSSÃO

Com os resultados obtidos em relação às métricas de desempenho, pode-se dizer que em geral a lematização tende a apresentar melhores resultados quando em comparação com os métodos de stemização. Nota-se que principalmente a stemização com o algoritmo de Lancaster apresenta desempenho ligeiramente inferior, uma vez que, por ser mais agressiva pode reduzir excessivamente as palavras, prejudicando a análise.

Em relação às técnicas de extração de característica dos textos, o TF-IDF apresentou os melhores resultados, especialmente quando combinado com a lematização. Porém, é uma técnica que envolve grande custo computacional e não foi possível ter mais dados processados dessa forma para confirmar seu melhor desempenho para as outras situações além da língua portuguesa.

Como os resultados em português foram melhores que em inglês na técnica BoW, é difícil afirmar que o modelo TF-IDF realmente é o mais adequado. Esse melhor desempenho nos resultados pode ser causado por um processamento de linguagem natural mais adequado para a estrutura linguística do português ou que o *dataset* é mais consistente.

Além dos resultados apresentados das métricas de desempenho, também foram geradas as matrizes de confusão:

Tweets em Inglês – BoW/Lematização		Valores Previstos	
		0	1
Valores Reais	0	VN = 818	FP = 224
	1	FN = 328	VP = 630

Tweets em Inglês – BoW/Lematização		Valores Previstos	
		0	1
Valores Reais	0	VN = 835	FP = 207
	1	FN = 358	VP = 600

Tweets em Inglês – BoW/Lematização		Valores Previstos	
		0	1
Valores Reais	0	VN = 823	FP = 219
	1	FN = 371	VP = 587

Tabelas 4, 5 e 6 - Matrizes de Confusão para o dataset de tweets em inglês (amostras)

Reddit – BoW/Lematização		Valores Previstos	
		0	1
Valores Reais	0	VN = 233	FP = 127
	1	FN = 98	VP = 267

Reddit – BoW/Lematização		Valores Previstos	
		0	1
Valores Reais	0	VN = 234	FP = 126
	1	FN = 97	VP = 268

Reddit – BoW/Lematização		Valores Previstos	
		0	1
Valores Reais	0	VN = 235	FP = 125
	1	FN = 105	VP = 260

Tabelas 7, 8 e 9 - Matrizes de Confusão para o dataset com postagens do Reddit

Tweets em Português BoW/Lematização		Valores Previstos			
		0	1	2	3
Valores Reais	0	365	13	12	31
	1	13	62	0	10
	2	10	3	55	8
	3	48	5	9	165

Tweets em Português TF-IDF/Lematização		Valores Previstos			
		0	1	2	3
Valores Reais	0	393	7	4	17
	1	19	55	0	11
	2	13	2	50	11
	3	64	2	5	156

Tweets em Português BoW/Stemização		Valores Previstos			
		0	1	2	3
Valores Reais	0	359	13	11	38
	1	15	61	1	8
	2	12	3	54	7
	3	46	3	11	167

Tweets em Português TF-IDF/Stemização		Valores Previstos			
		0	1	2	3
Valores Reais	0	388	6	5	22
	1	19	56	0	10
	2	13	2	52	9
	3	69	2	5	151

Tabelas 10, 11, 12 e 13 - Matrizes de Confusão para o dataset de tweets em português

É possível observar que no *dataset* de *tweets* em inglês há uma quantidade significativa de Falsos Positivos e Falsos Negativos (entre 27,6% e 29,5% do total), indicando que há uma dificuldade em identificar corretamente as classes.

Do mesmo modo, no *dataset* do *Reddit*, também há um número significativo de Falsos Positivos e Falsos Negativos se for analisada a proporção de incidências em relação ao total de testes. De modo absoluto aparenta ter um desempenho melhor nessa questão que o *dataset* anterior, mas de modo relativo apresenta de 30,5% a 31,7% de FP e FN do total.

Em ambos os conjuntos de dados, existe uma variação pequena entre as matrizes, indicando uma boa consistência nos resultados.

Reforçando o obtido nas métricas, as matrizes de confusão do *dataset* em português indicam que o desempenho do TF-IDF combinado com a lematização apresenta os melhores resultados; com alta taxa de acerto das classes, e uma redução de erros ao comparar com ad demais matrizes.

Nota-se nesta base de dados que há uma certa dificuldade do algoritmo em indicar corretamente a classe 0 (nenhum sintoma). Os maiores valores de erros em todas as matrizes apontam que o algoritmo classificou instâncias da classe 0 como sendo da classe 3 (somente psíquico).

Com base nas métricas de desempenho, observou-se que a lematização tende a apresentar melhores resultados em comparação com a stemização, particularmente quando se utiliza o TF-IDF. As matrizes de confusão geradas para os diferentes *datasets* revelam:

- *Tweets* em Inglês: Alta taxa de falsos positivos e negativos, variando entre 27,6% e 29,5%, sugerindo dificuldades na correta identificação das classes.
- Postagens do *Reddit*: Desempenho ligeiramente melhor que os tweets em inglês, mas ainda com 30,5% a 31,7% de falsos positivos e negativos.

Tweets em Português: Melhor desempenho geral com TF-IDF e lematização, evidenciando alta taxa de acerto e consistência nos resultados. Outra forma de entender

os *dataset* utilizada que auxilia na identificação de elementos que podem ter uma grande importância no conjunto é a nuvem de palavras, ou *word cloud*. Essa ferramenta auxilia na representação da frequência das palavras em uma forma gráfica de fácil apreensão visual, permitindo identificar rapidamente os termos mais comuns utilizados nos textos. Essa técnica poderia ter sido utilizada para maior refinamento da limpeza dos dados, uma vez que foi executada após a normalização.

A seguir, as nuvens de palavras de cada *dataset* utilizado:

Figura 18 - Nuvem de palavras do dataset de tweets em inglês

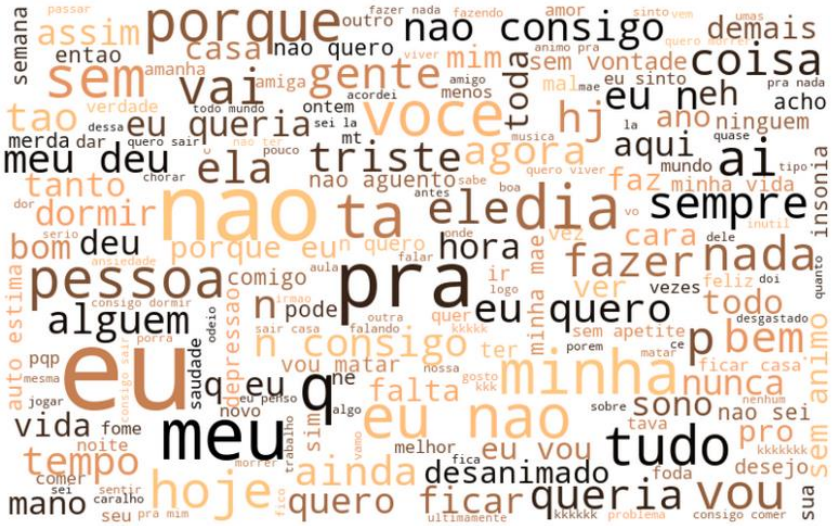


Figura 19 - Nuvem de palavras do dataset de tweets em português

Figura 20 - - Nuvem de palavras do dataset de postagens no reddit

Observa-se nessas nuvens que palavras como '*feel*', '*know*' e '*think*' estão fortemente presentes nos dois *datasets* em inglês, principalmente a palavra '*feel*', que significa sentir. As palavras '*life*' e '*people*', 'vida' e 'pessoas' respectivamente, também possuem uma razoável frequência em ambos os conjuntos em inglês. Essas palavras em alta frequência podem indicar que os textos possuem um conteúdo emocional e subjetivo, sobre percepções e pensamentos sobre a vida e as pessoas, sendo relevantes para a identificação de distúrbios como depressão.

No *dataset* em português algumas palavras podem ser relacionadas a essas impressões obtidas nos de língua inglesa, por exemplo, 'gente', 'pessoa', 'ele', 'alguem' são palavras com certo destaque na nuvem. Esta nuvem também exibe alta frequência de termos como 'triste, 'desanimado', 'nao consigo', que podem acender o alerta para depressão.

Outro ponto a considerar é o da palavra 'eu' estar muito presente no *dataset* em português, assim como no do *Reddit* há uma alta frequência do termo '*im*', que é contração de '*I am*', traduzindo, 'eu sou/eu estou'. A grande frequência desses termos indica alta frequência de autorreferência, sugerindo que os usuários frequentemente falem sobre si mesmos, o que pode ser sinal de introspecção ou de compartilhamento de experiências e sentimentos pessoais.

Para além dos resultados obtidos, existiram questões a serem observadas ao longo do desenvolvimento do trabalho. Primeiro, podemos citar a atual dificuldade de raspagem e obtenção de dados em redes sociais. Mudanças recentes em políticas internas das plataformas tornaram essa uma tarefa que gastaria muitos recursos, ou temporais e computacionais ao tentar desenvolver perfis robôs que coletam os dados, com prejuízo em questões de coletar anonimamente; ou financeiras, ao pagar para acessar as APIs. Existe opção de tentar acesso gratuito justificando motivos acadêmicos, porém para o presente trabalho não obtivemos resposta quanto ao pedido ao *Twitter* (atualmente X), além de ser um acesso extremamente limitado.

Outro desafio enfrentado foi em relação à limpeza, muitos aspectos exigem um conhecimento em relação às nuances das línguas, com entendimento de regras gramaticais, variações linguísticas, gírias e contextos culturais. Além disso, o tratamento

de abreviações, erros de digitação ou ortográficos e diferentes formas de expressões idiomáticas demanda conhecimento de técnicas sofisticadas e uma grande especialização para garantir que os dados sejam preparados da melhor forma para as etapas de modelagem e análise.

Entretanto, o maior empecilho encontrado foi em relação à capacidade computacional exigida para alguns modelos. Para o processamento de linguagem natural é importante uma grande quantidade de dados, pois quanto mais dados, mais abrangente e precisa é a representação da língua, auxiliando no entendimento de padrões e regularidades da língua e na generalização dos modelos, além de oferecer mais robustez a variações e ruídos presentes na linguagem natural.

A linguagem humana é muito diversa e complexa, grandes quantidades de dados são essenciais para um bom modelo. Com isso, não era uma boa opção reduzir demais os *datasets*, optou-se por realizar isso o mínimo possível, mesmo que prejudicasse a análise em uma das técnicas.

Em resumo, apesar das dificuldades encontradas, os resultados alcançados indicam um desempenho satisfatório na aplicação de técnicas de processamento de linguagem natural. Existem melhorias que poderiam ser realizadas, com mais conhecimentos linguísticos e recursos computacionais; porém as técnicas utilizadas estão bem descritas e detalhadas, demonstrando quais as melhores opções a serem seguidas em trabalhos futuros.

5. CONSIDERAÇÕES FINAIS

Diante do crescente desafio que as doenças mentais, especialmente a depressão, representam, principalmente após os impactos da pandemia de COVID-19, a análise de dados obtidos em redes sociais se mostra uma ferramenta promissora para a identificação precoce e monitoramento de distúrbios. Este estudo teve como objetivo explorar o potencial do processamento de linguagem natural e técnicas de machine learning na detecção de sinais de depressão a partir de postagens em plataformas sociais.

Com a revisão bibliográfica realizada permitiu identificar as técnicas mais adequadas para este tipo de análise, mas desafios significativos foram enfrentados, como a nova política de APIs pagas, dificuldades no pré-processamento adequado dos dados e a falta de recursos computacionais. Essas dificuldades, no entanto, não impediram o progresso do estudo.

Deste estudo inicial foram aproveitados o passo a passo do que devia ser realizado e as técnicas, iniciando pelo pré-processamento com normalização (tratamento de abreviações, remoção de emojis, caracteres especiais e outros), remoção das *stopwords* e stemização ou lematização. Após isso, nos diversos artigos analisados existiam algumas técnicas de extração de características, contudo, as abordagens que se destacaram em termos de uso foram a *Bag-of-words* e o TF-IDF, sendo também escolhidas para o presente trabalho. Por fim, ao invés de realizarmos diversos testes com diversos algoritmos, o que seria muito dispendioso, foi escolhido, com base nos estudos analisados, o algoritmo SVM por apresentar o melhor desempenho para o processo de modelagem.

Por último, a análise de desempenho das diversas combinações de técnicas e *datasets* foi realizada, e chegou-se à conclusão que apesar de ser possível realizar melhorias no tratamento dos dados, a abordagem utilizada obteve um bom resultado, indicando métodos adequados para serem utilizados no processamento de linguagem natural, com destaque para a utilização da lematização em união com o TF-IDF.

Em resumo, o presente trabalho demonstrou que, apesar dos desafios enfrentados, é possível utilizar técnicas de PLN e *machine learning* para detectar sinais de depressão em redes sociais com bom nível de acurácia. Futuras pesquisas podem focar na superação das limitações encontradas, explorando novos recursos computacionais e refinando as técnicas de pré-processamento para aprimorar ainda mais os resultados.

REFERÊNCIAS

1. PAN AMERICAN HEALTH ORGANIZATION. Pandemia de COVID-19 desencadeia aumento de 25% na prevalência de ansiedade e depressão em todo o mundo. **Pan American Health Organization – PAHO**, 2022. Disponivel em: <https://www.paho.org/pt/noticias/2-3-2022-pandemia-covid-19-desencadeia-aumento-25-na-prevalencia-ansiedade-e-depressao-em>. Acesso em: 17 Junho 2023.

2. WORLD HEALTH ORGANIZATION. **World Mental Health Report - Transforming mental health for all**. World Health Organization - WHO. [S.l.]. 2022.

3. MENTAL HEALTH FOUNDATION. **Stress: Are we coping?** Mental Health Foundation. [S.l.]. 2018.

4. TSUGAWA, S. et al. Recognizing Depression from Twitter Activity. **33rd Annual ACM Conference**, abril 2015. 3187-3196.

5. DE CHOUDHURY, et al. Predicting Depression via Social Media. **Proceedings of the International AAAI Conference on Web and Social Media**, Agosto 2021. 128-137.

6. BASTOS, A. Análise de dados: uma ferramenta para criar melhores estratégias de negócio. **Alura**, março 2023. Disponivel em: <https://www.alura.com.br/empresas/artigos/analise-de-dados>. Acesso em: 09 julho 2024.

7. KELLEHER, J. D.; MAC NAMEE, B.; D'ARCY, A. **Fundamentals of Machine Learning for Predictive Data Analytics**. 2ª. ed. Cambridge, Massachusetts: The MIT Press, 2020.

8. ORACLE. O que é Big Data? **Oracle**. Disponivel em: <https://www.oracle.com/br/big-data/what-is-big-data/>. Acesso em: 19 Junho 2023.

9. CHELLAPPA, S. L. Sonolência excessiva diurna e depressão: causas, implicações clínicas e manejo terapêutico. **Revista de Psiquiatria do Rio Grande do Sul**, 2009. Disponivel em: <https://www.scielo.br/j/rprs/a/dQWdBbh7yJ6ZhgNX8cs7TKH/#>. Acesso em: 09 julho 2024.

10. ZULFIKER, M. S. et al. An in-depth analysis of machine learning approaches to predict depression. **Current Research in Behavioral Sciences (CRBS) V.02**, 2021. Disponivel em: <https://www.sciencedirect.com/science/article/pii/S2666518221000310>.

11. JORDAN, D.; CESAR, V.; CARMO, L. D. Análise de Técnicas de PLN e Machine Learning na Detecção de Indícios de Depressão. **Universidade Presbiteriana Mackenzie**, 2022. Disponivel em: <https://dspace.mackenzie.br/handle/10899/31207>.

12. AGUIAR, E. J. D. et al. Análise de Sentimento em Redes Sociais para a Língua Portuguesa Utilizando Algoritmos de Classificação. **XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**, 10 Maio 2018. Disponivel em: <https://sol.sbc.org.br/index.php/sbrc/article/view/2430>.

13. MELO, A. L. B. D.; ALVES, A. L. F. Aplicação de Técnicas de Aprendizagem de Máquina para Diagnóstico de Depressão, Ansiedade e Estresse. **IX Encontro Nacional de Computação dos Institutos Federais**, 31 Julho 2022. Disponivel em: <https://sol.sbc.org.br/index.php/encompif/article/view/20429>.

14. PRIYA, A.; GARG, S.; TIGGA, N. P. Predicting Anxiety, Depression and Stress in Modern Life using Machine Learning Algorithms. **Procedia Computer Science V.167**, 2019. Disponivel em: <https://www.sciencedirect.com/science/article/pii/S1877050920309091>.

15. YADAV, S. et al. Medical Sentiment Analysis using Social Media: Towards building a Patient Assisted System. **In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)**, 2018. Disponivel em: <https://aclanthology.org/L18-1442/>.

16. CORTES, O. A. C.; MELO, W. E. D. O. Utilizando Análise de Sentimentos e SVM na Classificação de Tweets Depressivos. **XII Computer on The Beach 2021**, 2021. Disponivel em: <https://periodicos.univali.br/index.php/acotb/article/view/17388>.

17. CASANI, V. et al. DP-Symptom-Identifier: uma estratégia para classificar sintomas de depressão utilizando um conjunto de dados textuais na língua portuguesa. **XIII Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana**, 29 Novembro 2021. Disponivel em: <https://sol.sbc.org.br/index.php/stil/article/view/17794>.

18. MUZAFAR, D.; KHAN, F. Y.; QAYOOM, M. Machine Learning Algorithms for Depression Detection and Their Comparison, Janeiro 2023.

19. PACHOULY, P. S. J. et al. Depression Detection on Social Media Network (Twitter) using Sentiment Analysis. **International Research Journal of Engineering and Technology (IRJET)**, 2021. Disponivel em: <https://www.irjet.net/archives/V8/i1/IRJET-V8I1305.pdf>.

20. RACHANA, J. et al. Depression Detection from Social Media Data Using CNN and Linguistic Metadata Features. **Technix International Journal for Engineering Research (TIJER)**, Julho 2022. Disponivel em: <https://tijer.org/viewpaperforall?paper=TIJER2207006>.

21. SARKAR, D.; BALI, R.; SHARMA, T. **Practical Machine Learning with Python**. Bangalore, Karnataka, India: Apress, 2018.

22. FILHO, M. Guia Completo Sobre ROC AUC Em Machine Learning. **Mario Filho | Machine Learning**, janeiro 2023. Disponivel em: <https://mariofilho.com/guia-completo-sobre-roc-auc-em-machine-learning/>. Acesso em: 11 julho 2024.

23. INFAMOUSCODER. Depression: Twitter Dataset + Feature Extraction. **Kaggle**, 2022. Disponivel em: <https://www.kaggle.com/datasets/infamouscoder/mental-health-social-media>. Acesso em: 10 Setembro 2023.

24. ALI, H. M. Twitter Suicidal Data. **Kaggle**, 2023. Disponivel em: <https://www.kaggle.com/datasets/hosammhmdali/twitter-suicidal-data>. Acesso em: 10 setembro 2023.

25. KANT, L. twitter-suicidal-intention-dataset. **Github**, 2020. Disponivel em: <https://github.com/laxmimerit/twitter-suicidal-intention-dataset>. Acesso em: 10 setembro 2023.

26. KANT, M. Dataset for Stress Analysis in Social Media. **Kagle**, 2022. Disponivel em: <https://www.kaggle.com/datasets/monishakant/dataset-for-stress-analysis-in-social-media>. Acesso em: 13 setembro 2023.

27. MBAYE, M. Up-to-date list of Slangs for Text Preprocessing. **Kaggle**, 2000. Disponivel em: <https://www.kaggle.com/code/nmaguette/up-to-date-list-of-slangs-for-text-preprocessing>. Acesso em: 22 novembro 2023.

28. SIMÕES, L. GÍRIAS DE WHATSAPP: aprenda as principais siglas, abreviações e gírias usadas em conversas na Internet. **TV Jornal - UOL**, 2023. Disponivel em: <https://tvjornal.ne10.uol.com.br/noticias/2023/08/15576588-girias-de-whatsapp-aprenda-as-principais-siglas-abreviacoes-e-girias-usadas-em-conversas-na-internet.html>. Acesso em: 22 novembro 2023.

29. MAIS VOCÊ. Veja lista de abreviações usadas pelos jovens em troca de mensagens na internet. **gshow - Globo**, 2021. Disponivel em: <https://gshow.globo.com/programas/mais-voce/noticia/veja-lista-de-abreviacoes-usadas-pelos-jovens-em-troca-de-mensagens-na-internet.ghtml>. Acesso em: 22 novembro 2023.

30. ZVORNICANIN, E. Differences Between Porter and Lancaster Stemming Algorithms. **Baeldung**, 2024. Disponivel em: <https://www.baeldung.com/cs/porter-vs-lancaster-stemming-algorithms>. Acesso em: 20 julho 2024.

31. SANDE, N.; ADENIYI, I. O.; AKINKUNMI, A. Social Media Sentiment Analysis: A Comprehensive Analysis, fevereiro 2024. Disponivel em: <https://www.researchgate.net/publication/378150391_Social_Media_Sentiment_Analysis_A_Comprehensive_Analysis>. Acesso em: 9 Julho 2024.

32. SIPIO, R. D. A Quick Guide to AUC-ROC in Machine Learning Models. **Medium**, maio 2021. Disponivel em: <https://towardsdatascience.com/a-quick-guide-to-auc-roc-in-machine-learning-models-f0aedb78fbad>. Acesso em: 11 julho 2024.

33. PYTHON. re — Regular expression operations. **Python Documentation**. Disponivel em: <https://docs.python.org/3/library/re.html>. Acesso em: 8 outubro 2023.

34. MOURÃO, J. P. Abreviações e siglas em inglês usadas na internet. **Mundo Educação - UOL**. Disponivel em: <https://mundoeducacao.uol.com.br/ingles/abreviacoes-siglas-ingles-usadas-na-internet.htm>. Acesso em: 22 novembro 2023.

35. S., M. As abreviações mais usadas na internet para mensagens de texto e tweets. **Preply**, 2023. Disponivel em: <https://preply.com/pt/blog/as-abreviacoes-mais-usadas-na-internet-para-mensagens-de-texto-e-tweets/>. Acesso em: 22 novembro 2023.

36. SEMANTIX LAB. TF-IDF: entenda o que é Frequency Inverse Document Frequency! **Semantix**. Disponivel em: <https://semantix.ai/tf-idf-entenda-o-que-e-frequency-inverse-document-frequency/>. Acesso em: 15 maio 2024.

37. TRIPATHI, M. How to process textual data using TF-IDF in Python. **freeCodeCamp**, 2018. Disponivel em: <https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/>. Acesso em: 12 abril 2024.

38. SCIKIT-LEARN. ConfusionMatrixDisplay. **Scikit Learn**. Disponivel em: <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics.ConfusionMatrixDisplay>. Acesso em: 20 julho 2024.

39. ABDULLAH, B.; AAHIL. Métricas da avaliação. **Microsoft**, 2023. Disponivel em: <https://learn.microsoft.com/pt-br/azure/cognitive-services/language-service/custom-text-classification/concepts/evaluation-metrics>. Acesso em: 08 Junho 2023.

40. ALVES, I. N. Lemmatization vs. stemming: quando usar cada uma? **Alura**, 2021. Disponivel em: <https://www.alura.com.br/artigos/lemmatization-vs-stemming-quando-usar-cada-uma?gclid=CjwKCAjw1YCkBhAOEiwA5aN4AQfOeJFUX0F8JGXKl3L3W7iEAPcQxEA3fni7NyLOzypu6NOauUk5hBoCFXkQAvD_BwE>. Acesso em: 07 Junho 2023.

41. ARAUJO, R. Introdução ao SVM como Classificador. **LinkedIn**, 2020. Disponivel em: <https://www.linkedin.com/pulse/introdu%C3%A7%C3%A3o-ao-svm-como-classificador-rodrigo-araujo-/?originalSubdomain=pt>. Acesso em: 07 Junho 2023.

42. BRANDÃO, R. Estresse: tudo o que você precisa saber e dicas para lidar melhor. **ZenKlub**, 2020. Disponivel em: <https://zenklub.com.br/blog/para-voce/estresse/>. Acesso em: 04 Junho 2023.

43. BRUNA, M. H. V. Depressão. **Drauzio Varella - Uol**. Disponivel em: <https://drauziovarella.uol.com.br/doencas-e-sintomas/depressao/>. Acesso em: 03 Junho 2023.

44. FERREIRA, K. Análise de dados: o que é e dicas fundamentais para obter sucesso na sua análise. **RockContent**, 2019. Disponivel em: <https://rockcontent.com/br/blog/analise-de-dados/>. Acesso em: 04 Junho 2023.

45. GONÇALVES, T. PLN: o que é Processamento de Linguagem Natural? **Alura**, 2023. Disponivel em: <https://www.alura.com.br/artigos/o-que-e-pln?gclid=CjwKCAjwyeujBhA5EiwA5WD7_cap5rwuuH16ge5T6mWO0ISh1NMtKdRUtCgKYmCYABPGrt75YNq2NRoCq_YQAvD_BwE>. Acesso em: 04 Junho 2023.

46. KOJI, A.; FONSECA, C.; RODRIGUES, V. Introdução a Bag of Words e TF-IDF. **Medium**, 2020. Disponivel em: <https://medium.com/turing-talks/introdu%C3%A7%C3%A3o-a-bag-of-words-e-tf-idf-43a128151ce9>. Acesso em: 05 Junho 2023.

47. MARCEL, F. Stop Words – Como Funcionam Palavras de Parada? **Agência Mestre**, 2009. Disponivel em: <https://www.agenciamestre.com/seo/stop-words-como-funcionam-palavras-de-parada/>. Acesso em: 07 Junho 2023.

48. RODRIGUES, V. Métricas de Avaliação: acurácia, precisão, recall… quais as diferenças? **Medium**, 2019. Disponivel em: <https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c>. Acesso em: 08 Junho 2023.

49. SACRAMENTO, G. Processamento de Linguagem Natural (NLP): O que é e como funciona. **Tera**, 2021. Disponivel em: <https://blog.somostera.com/data-science/processamento-linguagem-natural-nlp>. Acesso em: 18 Junho 2023.

50. VARSHACHOUDHARY. What is Data Normalization? **Geeks for Geeks**, 2023. Disponivel em: <https://www.geeksforgeeks.org/what-is-data-normalization/>. Acesso em: 07 Junho 2023.

51. WAINER, J. MO432 - Processamento de textos. **Instituto de Computação - Unicamp**, 2018. Disponivel em: <https://www.ic.unicamp.br/~wainer/cursos/1s2021/432/cap-text.html>. Acesso em: 07 Junho 2023.

APÊNDICES

APÊNDICE A – Arquivo ‘slang_dic.py’

```
# Utilizando o notebook de: https://www.kaggle.com/code/nmaguette/up-to-date-list-of-slangs-for-text-preprocessing


abbreviations = {
    "$" : " dollar ",
    "€" : " euro ",
    "4ao" : "for adults only",
    "a.m" : "before midday",
    "a3" : "anytime anywhere anyplace",
    "aamof" : "as a matter of fact",
    "acct" : "account",
    "adih" : "another day in hell",
    "afaic" : "as far as i am concerned",
    "afaict" : "as far as i can tell",
    "afaik" : "as far as i know",
    "afair" : "as far as i remember",
    "afk" : "away from keyboard",
    "app" : "application",
    "approx" : "approximately",
    "apps" : "applications",
    "asap" : "as soon as possible",
    "asl" : "age, sex, location",
    "atk" : "at the keyboard",
    "ave." : "avenue",
    "aymm" : "are you my mother",
    "ayor" : "at your own risk",
    "b&b" : "bed and breakfast",
    "b+b" : "bed and breakfast",
    "b.c" : "before christ",
    "b2b" : "business to business",
    "b2c" : "business to customer",
    "b4" : "before",
    "b4n" : "bye for now",
    "b@u" : "back at you",
    "bae" : "before anyone else",
    "bak" : "back at keyboard",
    "bbbg" : "bye bye be good",
    "bbc" : "british broadcasting corporation",
    "bbias" : "be back in a second",
    "bbl" : "be back later",
    "bbs" : "be back soon",
    "be4" : "before",
    "bfn" : "bye for now",
    "blvd" : "boulevard",
    "bout" : "about",
    "brb" : "be right back",
    "bros" : "brothers",
    "brt" : "be right there",
    "bsaaw" : "big smile and a wink",
    "btw" : "by the way",
    "bwl" : "bursting with laughter",
    "c/o" : "care of",
```

```
    "cet" : "central european time",
    "cf" : "compare",
    "cia" : "central intelligence agency",
    "csl" : "can not stop laughing",
    "cu" : "see you",
    "cul8r" : "see you later",
    "cv" : "curriculum vitae",
    "cwot" : "complete waste of time",
    "cya" : "see you",
    "cyt" : "see you tomorrow",
    "dae" : "does anyone else",
    "dbmib" : "do not bother me i am busy",
    "diy" : "do it yourself",
    "dm" : "direct message",
    "dwh" : "during work hours",
    "e123" : "easy as one two three",
    "eet" : "eastern european time",
    "eg" : "example",
    "embm" : "early morning business meeting",
    "encl" : "enclosed",
    "encl." : "enclosed",
    "etc" : "and so on",
    "faq" : "frequently asked questions",
    "fawc" : "for anyone who cares",
    "fb" : "facebook",
    "fc" : "fingers crossed",
    "fig" : "figure",
    "fimh" : "forever in my heart",
    "ft." : "feet",
    "ft" : "featuring",
    "ftl" : "for the loss",
    "ftw" : "for the win",
    "fwiw" : "for what it is worth",
    "fyi" : "for your information",
    "g9" : "genius",
    "gahoy" : "get a hold of yourself",
    "gal" : "get a life",
    "gcse" : "general certificate of secondary education",
    "gfn" : "gone for now",
    "gg" : "good game",
    "gl" : "good luck",
    "glhf" : "good luck have fun",
    "gmt" : "greenwich mean time",
    "gmta" : "great minds think alike",
    "gn" : "good night",
    "g.o.a.t" : "greatest of all time",
    "goat" : "greatest of all time",
    "goi" : "get over it",
    "gps" : "global positioning system",
    "gr8" : "great",
    "gratz" : "congratulations",
    "gyal" : "girl",
    "h&c" : "hot and cold",
    "hp" : "horsepower",
    "hr" : "hour",
    "hrh" : "his royal highness",
    "ht" : "height",
    "ibrb" : "i will be right back",
    "ic" : "i see",
    "icq" : "i seek you",
    "icymi" : "in case you missed it",
    "idc" : "i do not care",
    "idgadf" : "i do not give a damn fuck",
```

```
    "idgaf" : "i do not give a fuck",
    "idk" : "i do not know",
    "ie" : "that is",
    "i.e" : "that is",
    "ifyp" : "i feel your pain",
    "IG" : "instagram",
    "iirc" : "if i remember correctly",
    "ilu" : "i love you",
    "ily" : "i love you",
    "imho" : "in my humble opinion",
    "imo" : "in my opinion",
    "imu" : "i miss you",
    "iow" : "in other words",
    "irl" : "in real life",
    "j4f" : "just for fun",
    "jic" : "just in case",
    "jk" : "just kidding",
    "jsyk" : "just so you know",
    "l8r" : "later",
    "lb" : "pound",
    "lbs" : "pounds",
    "ldr" : "long distance relationship",
    "lmao" : "laugh my ass off",
    "lmfao" : "laugh my fucking ass off",
    "lol" : "laughing out loud",
    "ltd" : "limited",
    "ltns" : "long time no see",
    "m8" : "mate",
    "mf" : "motherfucker",
    "mfs" : "motherfuckers",
    "mfw" : "my face when",
    "mofo" : "motherfucker",
    "mph" : "miles per hour",
    "mr" : "mister",
    "mrw" : "my reaction when",
    "ms" : "miss",
    "mte" : "my thoughts exactly",
    "nagi" : "not a good idea",
    "nbc" : "national broadcasting company",
    "nbd" : "not big deal",
    "nfs" : "not for sale",
    "ngl" : "not going to lie",
    "nhs" : "national health service",
    "nrn" : "no reply necessary",
    "nsfl" : "not safe for life",
    "nsfw" : "not safe for work",
    "nth" : "nice to have",
    "nvr" : "never",
    "nyc" : "new york city",
    "oc" : "original content",
    "og" : "original",
    "ohp" : "overhead projector",
    "oic" : "oh i see",
    "omdb" : "over my dead body",
    "omg" : "oh my god",
    "omw" : "on my way",
    "p.a" : "per annum",
    "p.m" : "after midday",
    "pm" : "prime minister",
    "poc" : "people of color",
    "pov" : "point of view",
    "pp" : "pages",
    "ppl" : "people",
```

```
    "prw" : "parents are watching",
    "ps" : "postscript",
    "pt" : "point",
    "ptb" : "please text back",
    "pto" : "please turn over",
    "qpsa" : "what happens" #"que pasa",
    "ratchet" : "rude",
    "rbtl" : "read between the lines",
    "rlrt" : "real life retweet",
    "rofl" : "rolling on the floor laughing",
    "roflol" : "rolling on the floor laughing out loud",
    "rotflmao" : "rolling on the floor laughing my ass off",
    "rt" : "retweet",
    "ruok" : "are you ok",
    "sfw" : "safe for work",
    "sk8" : "skate",
    "smh" : "shake my head",
    "sq" : "square",
    "srsly" : "seriously",
    "ssdd" : "same stuff different day",
    "tbh" : "to be honest",
    "tbs" : "tablespooful",
    "tbsp" : "tablespooful",
    "tfw" : "that feeling when",
    "thks" : "thank you",
    "tho" : "though",
    "thx" : "thank you",
    "tia" : "thanks in advance",
    "til" : "today i learned",
    "tl;dr" : "too long i did not read",
    "tldr" : "too long i did not read",
    "tmb" : "tweet me back",
    "tntl" : "trying not to laugh",
    "ttyl" : "talk to you later",
    "u" : "you",
    "u2" : "you too",
    "u4e" : "yours for ever",
    "utc" : "coordinated universal time",
    "w/" : "with",
    "w/o" : "without",
    "w8" : "wait",
    "wassup" : "what is up",
    "wb" : "welcome back",
    "wtf" : "what the fuck",
    "wtg" : "way to go",
    "wtpa" : "where the party at",
    "wuf" : "where are you from",
    "wuzup" : "what is up",
    "wywh" : "wish you were here",
    "yd" : "yard",
    "ygtr" : "you got that right",
    "ynk" : "you never know",
    "zzz" : "sleeping bored and tired",

    #--- Adições próprias

    "ptsd" : "post traumatic stress disorder",
    "tw" : "trigger warning",
    "b4" : "before",
    "wk" : "week",
    "ffs" : "for fucks sake",
    "aaf" : "as a friend",
    "aam" : "all about me",
```

```
    "abm" : "a big mistake",
    "abt" : "about",
    "aml" : "all my love",
    "g2g" : "got to go",
    "gtg" : "got to go",
    "hth" : "hope this help",
    "rotfl" : "rolling on the floor laughing",
    "np" : "no problem",
    "otoh" : "on the other hand",
    "pvt" : "private",
    "af" : "at first",
    "ywk" : "you would know",
    "ywmc" : "your wish is my command",
    "ywsyls" : "you win some, you lose some",
    "yyssw" : "yeah yeah sure sure whatever",
    "dunno" : "i dont know",
    "hifw" : "how i feel when",
    "qq" : "crying",
    "ianal" : "i am not a lawyer",
    "soml" : "story of my life",
    "oh" : "overheard",
    "gras" : "generally recognized as safe",
    "op" : "original poster",
    "prt" : "please retweet"

}
```

APÊNDICE B – Arquivo ‘contraction_dic.py’

```
contractions = {
    "gonna" : "going to",
    "wanna" : "want to",
    "gotta" : "got to",
    "aint" : "is not",
    "dont" : "do not",
    "cant" : "can not",
    "shouldnt" : "should not",
    "wouldnt" : "would not",
    "didnt" : "did not",
    "wont" : "will not",
    "arent" : "are not",
    "isnt" : "is not",
    "havent" : "have not",
    "hasnt" : "has not",
    "werent" : "were not",
    "shouldve" : "should have",
    "wouldve" : "would have",
    "couldve" : "could have",
    "mustnt" : "must not",
    "mightnt" : "might not",
    "shant" : "shall not",
    "oughtnt" : "ought not",
    "couldnt" : "could not",
    "wouldntve" : "would not have",
    "shouldntve" : "should not have",
    "mightve" : "might have",
    "mustve" : "must have",
    "couldntve" : "could not have",


    ## "Hes" : contração de "he is" ou "he has".
    ## "Shes" : contração de "she is" ou "she has".


    "theyre" : "they are",
    "ive" : "i have",
    "youve" : "you have",
    "theyve" : "they have",
    "youll" : "you will",
    "theyll" : "they will",
    "its" : "it is"

}
```

APÊNDICE C – Arquivo ‘slang_dic_pt.py’‘

```
# Português

abbreviations = {
    "add" : "adicionar",
    "agr" : "agora",
    "ajd" : "ajuda",
    "aki" : "aqui",
    "amg" : "amigo",
    "aq" : "aqui",
    "bb" : "bebe",
    "bj" : "beijo",
    "bjs" : "beijos",
    "blz" : "beleza",
    "bnt" : "bonito",
    "bpn" : "bom para nos",
    "btf" : "boto fe",
    "clr" : "celular",
    "cmg" : "comigo",
    "ctt" : "contato",
    "cvs" : "conversar",
    "dlç" : "delicia",
    "dmr" : "demorou",
    "dnv" : "de novo",
    "dps" : "depois",
    "drx" : "tudo certo", #de rocha
    "fb" : "facebook",
    "flw" : "falou",
    "fml" : "famIlia",
    "fmz" : "firmeza",
    "ft" : "foto",
    "fut" : "futebol",
    "fzr" : "fazer",
    "glr" : "galera",
    "gnt" : "gente",
    "hrs" : "horas",
    "ig" : "instagram",
    "mddc" : "meu deus do ceu",
    "mdm" : "melhor do mundo",
    "mds" : "meu deus",
    "mlk" : "moleque",
    "mlr" : "melhor",
    "mn" : "mano",
    "msg" : "mensagem",
    "msm" : "mesmo",
    "nd" : "nada",
    "ngc" : "negocio",
    "ngm" : "ninguem",
    "nn" : "nao",
    "obg" : "obrigada",
    "pcr" : "parceiro",
    "pdc" : "pode crer",
    "pft" : "perfeito",
    "pfv" : "por favor",
    "pldm" : "pelo amor de deus",
    "plmdds" : "pelo amor de deus",
    "plmds" : "pelo amor de deus",
    "plmns" : "pelo menos",
    "pls" : "por favor",
```

```
    "pprt" : "papo reto",
    "pq" : "porque",
    "pse" : "pois e",
    "pv" : "privado",
    "pvd" : "privado",
    "pvt" : "privado",
    "qria" : "queria",
    "qser" : "quiser",
    "qt" : "quando",
    "rlx" : "relaxa",
    "rs" : "risos",
    "sdds" : "saudades",
    "sla" : "sei la",
    "sl" : "sei la",
    "slk" : "voce e louco",
    "sm" : "sem",
    "smp" : "sempre",
    "sqn" : "so que nao",
    "ss" : "sim",
    "tb" : "tambem",
    "tbm" : "tambem",
    "td" : "tudo",
    "tds" : "todos",
    "tj" : "estamos juntos",
    "tlg" : "ta ligado",
    "tlgd" : "ta ligado",
    "tm" : "estamos",
    "tmj" : "estamos juntos",
    "trd" : "tarde",
    "vc" : "voce",
    "vdd" : "verdade",
    "vlw" : "valeu",
    "vms" : "vamos",
    "wpp" : "whatsapp"


}
```