UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Valentin Tiburcio

Avaliação Experimental de Classificadores para Análise de Sentimentos em Dados de Redes Sociais

Uberlândia, Brasil

2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Valentin Tiburcio

Avaliação Experimental de Classificadores para Análise de Sentimentos em Dados de Redes Sociais

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Maria Camila Nardini Barioni

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2021

Resumo

As redes sociais virtuais são ambientes cada vez mais comuns no cotidiano das pessoas em todo o mundo. Os usuários das redes sociais compartilham informações sobre sua vida constantemente para outras pessoas, tornando as redes sociais em uma possível fonte de informação sobre as emoções de um indivíduo. O *Twitter* faz parte das redes sociais mais populares e utilizadas no mundo, onde seus usuários compartilham *tweets*, que são textos curtos de até 280 caracteres, e também imagens e vídeos. Por ser uma rede social tão utilizada e por disponibilizar um grande volume de *tweets* gratuitamente, diferentes estudos sobre análise de sentimentos são realizados utilizando esses dados. A análise de sentimentos combina conceitos de aprendizagem de máquina, recuperação de informação e linguística computacional para extrair informações sobre opiniões e emoções contidas em diversas fontes de dados, e então realizar uma classificação automática para um sentimento. Neste trabalho de conclusão de curso, são abordadas técnicas de pré-processamento de texto e cinco algoritmos para classificação de sentimento encontrados na literatura, sendo eles *Random Forest*, *Logistic Regression*, *Naive Bayes Classifier*, *K-Nearest Neighbor* (KNN) e *Support Vector Machine* (SVM). Neste contexto, foram feitos experimentos e comparações de desempenho entre os algoritmos utilizando uma base de dados pública. Foram feitas análises dos resultados das predições, com o objetivo de definir o classificador que melhor realizou a tarefa de classificação de sentimento. Os resultados mostram os algoritmos SVM e *Logistic Regression* se sobressaindo sobre os demais em todos os experimentos, obtendo 77% e $76,1\%$ de acurácia respectivamente. Já o algoritmo KNN com os piores resultados de classificações de sentimento com no máximo $59,6\%$ de acurácia.

Palavras-chave: Análise de Sentimentos, *Twitter*, Pré-processamento, Classificadores de Sentimento, Métricas de Avaliação.

Lista de ilustrações

Figura 1 – Exemplo de uma frase antes e depois de ser tokenizada. 16
Figura 2 – Exemplo de uma frase antes e depois de remover suas *stopwords* 16
Figura 3 – Exemplo de uma frase antes e depois de ser lematizada. 17
Figura 4 – Exemplo de uma frase antes e depois de passar pelo processo de *stemming*. 17
Figura 5 – Exemplo de um unigrama, bigrama e trigrama de uma frase. 18
Figura 6 – Funcionamento do algoritmo *Random Forest*. 22
Figura 7 – Exemplo de curva logística. 23
Figura 8 – Exemplo de classificação de um novo dado no algoritmo 1-NN em um espaço bidimensional. 27
Figura 9 – Exemplo de classificação de um novo elemento no algoritmo K-NN. . . 28
Figura 10 – Exemplo de hiperplano unidimensional dividindo duas classes distintas e vetores de suporte delimitando as margens. 29
Figura 11 – Exemplo de validação cruzada para 5 subconjuntos. 30
Figura 12 – Matriz de confusão de duas classes. 31
Figura 13 – Curva ROC de dois classificadores. 32
Figura 14 – Etapas da metodologia para avaliação dos classificadores 38
Figura 15 – Nuvem de palavras positivas e negativas da base de dados 39
Figura 16 – Distribuição de *tweets* POSITIVOS e NEGATIVOS da base de dados . 40
Figura 17 – Cinco *tweets* antes da etapa de pré-processamento. 40
Figura 18 – Cinco *tweets* depois da etapa de pré-processamento. 42
Figura 19 – Trecho de código exemplificando a criação de uma matriz TF-IDF e sua estrutura após a criação. 43
Figura 20 – Trecho de código da função para predição de sentimento dos *tweets*. . . 44
Figura 21 – Trecho de código exemplificando o processo de treinamento e recuperação das predições dos algoritmos de classificação. 45
Figura 22 – Trecho de código apresentando as dimensões da matriz TD-IDF X, a lista de sentimentos Y e o conjunto de predições *resultados*. 45
Figura 23 – Matrizes de confusão dos cinco classificadores estudados. 47
Figura 24 – Trecho de código utilizado para treinamento dos algoritmos de classificação sem utilização de n-grama. 48
Figura 25 – Trecho de código para obtenção dos resultados das métricas de avaliação dos experimentos 1 e 2. 50
Figura 26 – Trecho de código apresentando a utilização de bigramas na função *TfidfVectorizer*. 52
Figura 27 – Curvas ROC dos quatro experimentos realizados. 54

Lista de tabelas

Tabela 1 – Tabela termo-frequência utilizando a medida TF. 19
Tabela 2 – Tabela termo-valor utilizando a medida TF-IDF. 20
Tabela 3 – Tabela comparativa entre um *tweet* da base de dados e sua versão tokenizada. 40
Tabela 4 – Tabela comparativa entre um *tweet* da base de dados e sua versão após remoção de *stopwords*. 41
Tabela 5 – Tabela comparativa entre um *tweet* da base de dados e sua versão lematizada. 41
Tabela 6 – Tabela comparativa entre um *tweet* base de dados e sua versão após processo de *stemming*. 41
Tabela 7 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 1. 49
Tabela 8 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 2. 49
Tabela 9 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 1. 50
Tabela 10 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 2. 51
Tabela 11 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 3. 52
Tabela 12 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 4. 52
Tabela 13 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 3. 53
Tabela 14 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 4. 53

Sumário

1 INTRODUÇÃO 7
1.1 Objetivos 8
1.2 Justificativa 9
1.3 Organização da Monografia 9

2 ANÁLISE DE SENTIMENTOS 10
2.1 Etapas da Análise de Sentimentos 11
2.1.1 Identificação 11
2.1.2 Classificação da Polaridade 11
2.1.3 Sumarização 12
2.2 Abordagens de Classificação de Polaridade 13
2.2.1 Abordagem Baseada em Dicionário 13
2.2.2 Abordagem baseada em Aprendizado de Máquina 14
2.2.3 Abordagens Estatísticas e Semânticas 14
2.3 Pré-Processamento de Dados 15
2.3.1 Tokenização 15
2.3.2 Remoção de *Stopwords* 16
2.3.3 Lematização 16
2.3.4 *Stemming* 17
2.4 Ponderação de Termos 17
2.4.1 N-grama 18
2.4.2 TF-IDF 18

3 ALGORITMOS DE CLASSIFICAÇÃO 21
3.1 *Random Forest* 21
3.2 *Logistic Regression* 22
3.3 *Naive Bayes Classifier* 23
3.3.1 Probabilidade condicional 24
3.3.2 Independência condicional 24
3.3.3 Teorema de Bayes 24
3.3.4 Algoritmo Naive Bayes 25
3.4 *K-Nearest Neighbor (K-NN)* 26
3.5 *Support Vector Machine (SVM)* 28
3.6 Avaliação 29
3.6.1 Validação Cruzada 30
3.6.2 Matriz de Confusão 30

3.6.3 Métricas 31

4 TRABALHOS RELACIONADOS 34

5 EXPERIMENTOS E RESULTADOS 38
5.1 Metodologia 38
5.1.1 Base de dados 38
5.1.2 Pré-Processamento de Dados 39
5.1.3 Aplicação dos modelos de classificação 42
5.1.4 Resultados 46
5.1.4.1 Experimentos sem n-grama 48
5.1.4.2 Experimentos com n-grama 51

6 CONCLUSÃO E TRABALHOS FUTUROS 55

REFERÊNCIAS 57

1 Introdução

Com o crescente aumento do uso da internet e redes sociais virtuais como o *Facebook*, *Twitter* e o *Instagram*, as pessoas estão gerando todos os dias uma quantidade enorme de dados não estruturados ou semi estruturados na rede. Como apontado em Hemmatian e Sohrabi (2019), há cerca de $2,5 \times 10^{18}$ de *bytes* sendo criados a cada dia na *internet* com a tendência desse número aumentar. Muitos desses dados contém informações relevantes de usuários expondo suas opiniões, posicionamentos políticos e emoções sobre os mais variados temas, assuntos e acontecimentos de seu dia-a-dia. Por conta disso, a área de mineração de texto ganhou bastante destaque devido às técnicas utilizadas para processar, filtrar e classificar esse grande volume de dados disponível na *web* e principalmente nas redes sociais, motivando diversas pesquisas não só na área de mineração de texto, mas também na subárea de análise de sentimento.

As redes sociais disponibilizam uma grande quantidade de informações sobre a vida, os gostos, os costumes, emoções e opiniões de um indivíduo. Esses dados são obtidos através de comentários e *feedbacks* sobre um item adquirido por uma loja ou por um serviço prestado por uma empresa, além de publicações sobre temas políticos e sociais, entre outras interações que uma pessoa possui nas redes sociais.

Empresas e organizações se utilizam desses dados para aumentar suas fontes de opinião sobre seus produtos e serviços oferecidos com o objetivo de melhorar suas estratégias de negócio. Por outro lado, pesquisadores são desafiados a estudar e desenvolver novas técnicas, métodos e ferramentas para automatizar o processo de análise desse grande volume de dados, que muitas vezes se encontra não estruturado ou semi estruturado, a fim de acelerar a tarefa de identificar e classificar os sentimentos expressos nessas informações.

O aumento do uso de redes sociais virtuais, abre portas para oportunidades de estudos sobre saúde mental dos usuários dessas plataformas a partir de suas publicações Souza e Becker (2019). Esse grande volume de dados fornece uma rica fonte de informação que pode ser explorada para reconhecer sentimentos automaticamente e posteriormente auxiliar nos critérios de diagnóstico do usuário.

A análise de sentimento, também conhecida como mineração de opinião, é uma subárea da mineração de texto (HEMMATIAN; SOHRABI, 2019) que agrega também recuperação de informação e linguística computacional, com o objetivo de identificar, extrair e analisar sentimentos, opiniões, atitudes e avaliações dos usuários expressos em textos não estruturados. O problema da análise de sentimento é estruturado por diversos autores de variadas maneiras, porém é possível modelar o problema em 3 tarefas genéricas (BECKER; TUMITAN, 2013), sendo elas:

1. Identificar as opiniões sobre um assunto contidas em um conjunto de dados;
2. Classificar a opinião utilizando alguma técnica de classificação para definir a orientação ou polaridade da opinião;
3. Apresentar os dados obtidos de forma agregada ou sumarizada utilizando textos, gráficos, entre outros.

Este tabalho faz parte de um projeto maior que tem como objetivo realizar um estudo observacional transversal retrospectivo sobre a influência das redes sociais no temperamento de discentes da UFU, já aprovado pelo CEP/UFU. Esse projeto visa extrair os dados das redes sociais virtuais *Facebook*, *Twitter* e *Instagram*, dos usuários participantes, com sua devida autorização a fim de obter uma base consistente e confiável de informação.

1.1 Objetivos

O objetivo geral deste trabalho consistiu em realizar uma avaliação experimental de diferentes classificadores de análise de sentimento e técnicas de representação de postagens do *Twitter*. O desempenho de cada classificador foi avaliado considerando critérios de avaliação propostos nos trabalhos correlatos e na literatura estudada, com o intuito de contribuir para a escolha da técnica de classificação mais apropriada para o contexto do projeto maior citado anteriormente.

Os objetivos específicos do trabalho de conclusão de curso descrito nesta monografia consistem em trabalhar com uma base de dados já existente da rede social *Twitter* na língua portuguesa para realizar a análise do algoritmo de classificação ideal para o trabalho como um todo. Entre os objetivos específicos, é possível citar:

- Estudar e separar os algoritmos de classificação de sentimentos encontrados na literatura e definir quais deles serão avaliados no trabalho;
- Aplicar os algoritmos de classificação definidos na etapa anterior e obter seus resultados iniciais nos experimentos;
- Avaliar o desempenho de cada classificador através de métricas de avaliação encontradas na literatura;
- Comparar os algoritmos de classificação com base nos resultados das métricas de avaliação;
- Sumarizar os resultados das avaliações dos classificadores para facilitar a visualização dos resultados obtidos e as métricas utilizadas para a definição do(s) algoritmo(s) de classificação de sentimento para o projeto.

1.2 Justificativa

A área de mineração de texto vem ganhando visibilidade por conta do enorme volume de dados gerado pelas pessoas na *internet*. Portanto há uma enorme necessidade de encontrar métodos para processar essas informações a fim de extrair opiniões e emoções dos usuários nesses textos (ALLAHYARI et al., 2017).

Consequentemente, diversas técnicas para classificar essas emoções e opiniões foram criadas com o objetivo de definir o sentimento de um usuário sobre um determinado produto, serviço, assunto ou evento. Além disso, o crescente uso das redes sociais virtuais, desperta o interesse de pesquisadores para estudos relacionados a saúde mental dos usuários e seus comportamentos dentro dela, com o objetivo de identificar possíveis problemas de saúde e auxiliando o diagonóstico dos usuários.

Com isso, o estudo apresentado nessa monografia foi necessário por conta do projeto sobre a influência das redes sociais no temperamento dos discentes da UFU. O trabalho descrito aqui contribuiu para a definição do algoritmo de classificação mais adequado para o contexto desse trabalho, e com isso, auxiliar nos estudos sobre o comportamento dos usuários que estão participando do projeto e assim, auxiliá-los em possíveis diagonósticos.

1.3 Organização da Monografia

Esta monografia está organizada da seguinte maneira:

1. Uma breve explicação dos conceitos e etapas empregadas na análise de sentimento de redes sociais virtuais, que foram estudadas e utilizadas na realização do trabalho de conclusão de curso, é apresentada no Capítulo 2;

2. Os classificadores selecionados para estudos e análise no trabalho de conclusão são apresentados no Capítulo 3;

3. Trabalhos relacionados com o tema estudado são apresentados no Capítulo 4;

4. No Capítulo 5 são apresentados os experimentos realizados em uma base de dados pública do *Twitter*, juntamente com a análise e apresentação dos resultados obtidos.

5. A conclusão do trabalho de conclusão de curso e os trabalhos futuros são apresentados no Capítulo 6.

2 Análise de Sentimentos

Segundo Pang e Lee (2008) pesquisas nas áreas de análise e tratamento de opiniões, sentimentos e emoções encontrados em textos, têm ganhado muita atenção por conta do grande volume de dados relevantes disponível nas redes sociais virtuais nas mais variadas formas, como notícias, comentários, blogs, publicações, bate-papos, entre muitas outras fontes. Esses ambientes propiciam e induzem usuários a expressarem fortemente suas opiniões e sentimentos sobre todo tipo de tema debatido, como eventos, produtos, serviços e situações do seu dia-a-dia (LIU; ZHANG, 2012). O conjunto de técnicas, algoritmos e modelos responsáveis por realizar o tratamento de opiniões é abordado pela área de Mineração de Opiniões (*Opinion Mining*) ou Análise de Sentimentos (*Sentiment Analysis*). Essas áreas são um campo multidisciplinar que abrange conceitos de mineração de dados, aprendizado de máquina, linguística, processamento de linguagem natural e análise textual, com o principal objetivo de analisar partes de textos para determinar a atitude, emoção, opinião ou sentimento do autor sobre algum tópico ou alvo (PANG; LEE, 2008; LIU; ZHANG, 2012).

Mineração de Opiniões e Análise de Sentimentos são frequentemente usadas como sinônimos para expressar o mesmo significado (LIU; ZHANG, 2012). Entretanto alguns pesquisadores declaram que essas duas áreas abordam problemas ligeiramente diferentes. Segundo Tsytsarau e Palpanas (2011), a Mineração de Opiniões está relacionada com à forma de determinar se um texto possui uma opinião, enquanto a Análise de Sentimentos se relaciona com a detecção da polaridade de sentimento, onde nesse caso é atribuído um sentimento positivo ou negativo para o texto que está sendo estudado.

Ainda segundo Liu e Zhang (2012), existem diferentes níveis de análise textual nas atividades de análise de sentimentos. Mais explicitamente sobre a granularidade da análise, podemos separar em três categorias:

1. **Granularidade de documento:** Essa tarefa consiste em classificar se a opinião contida em um documento possui um sentimento positivo, negativo ou neutro. Por exemplo, uma ferramenta de análise de sentimentos irá determinar se um comentário sobre um produto, expressa um opinião positiva ou negativa. Essa análise assume que cada documento possui opiniões sobre um único alvo;

2. **Granularidade de sentença:** Nesta tarefa o texto é dividido em sentenças menores e a análise é realizada sobre essas unidades a fim de definir o sentimento positivo, negativo ou neutro individualmente em cada sentença;

3. **Granularidade de entidade e de aspecto:** Nesta tarefa a análise é mais específica por conta da granularidade de documento e na granularidade de sentença se serem

capaz de relacionar a entidade com seu sentimento, ou seja, não é possível descobrir o que a pessoa gosta ou não gosta. Por exemplo, "*O produto não é dos melhores, mas o serviço prestado foi ótimo*", se essa sentença for analisada com granularidade de documento ou de sentença, ela poderia ser classificada como negativa mesmo que não seja totalmente negativa, por conta da relação à entidade "produto" possuir opinião negativa, porém em relação ao "serviço" o autor possui um opinião positiva.

2.1 Etapas da Análise de Sentimentos

A análise de sentimentos pode ser caracterizada em termos de três tarefas, sendo elas: identificar tópicos e sentenças com opiniões e sentimentos, classificar a polaridade desses sentimentos e sumarizar seus resultados (BECKER; TUMITAN, 2013). As próximas seções descrevem cada uma dessas tarefas.

2.1.1 Identificação

A tarefa de identificação possui como objetivo encontrar tópicos e aspectos existentes nos textos, com isso associá-los com o conteúdo subjetivo em um conjunto de textos extraídos de alguma fonte, como as redes sociais, plataformas de avaliações de produtos e serviços, entre outros. Essa etapa está muito relacionada e dependente do tipo de granularidade escolhida para análise por conta de sua forma de identificar as entidades, aspectos e sentimentos do conteúdo textual (TSYTSARAU; PALPANAS, 2011).

As aplicações mais frequentes em mineração de opiniões e análise de sentimentos, se concentram em revisões de produtos e serviços. Isso ocorre por conta do alvo ser mais facilmente identificado, uma vez que assume-se que todo documento se refere a somente uma única entidade.

Como a complexidade da identificação do alvo da opinião depende muito da mídia considerada, em jornais, blogs e postagens, não se conhece intuitivamente as entidades que estão envolvidas, podendo envolver inclusive diversas entidades em um mesmo trecho de texto. Em situações simples, podemos restringir a identificação a entidades pré-definidas, por exemplo a busca por nomes famosos ou marcas.

2.1.2 Classificação da Polaridade

A tarefa de classificação de sentimento, também conhecida como classificação de polaridade, é frequentemente abordada como um problema de classificação binário, ou seja, dado um determinado texto, a classificação é feita para uma das duas classes, sendo elas positivo ou negativo. Entretanto, em análises mais robustas onde é necessário aumentar o nível de detalhes de resultados, classes adicionais podem ser consideradas nas

análises de sentimento. Com isso estas classes podem ser estendidas em classificações de intervalos numéricos ou em diferentes graus de intensidades (TSYTSARAU; PALPANAS, 2011).

Outra abordagem que pode ser considerada é a categoria neutra, que abrange textos sem uma direção clara com relação a sua polaridade ou apenas textos sem sentimento expresso. Sendo assim, a etapa de classificação de polaridade é responsável por identificar sentimentos nos textos. Entretanto termos sem uma polaridade definida podem ser descartados na etapa de identificação (TSYTSARAU; PALPANAS, 2011).

Diferentes abordagens são propostas na literatura para classificação da polaridade. Essas abordagens serão apresentadas posteriormente na seção 2.2 desta monografia. Contudo, independente da abordagem utilizada, a classificação da polaridade possui alguns desafios, sendo eles:

- Palavras de sentimento podem enganar, pois uma vez que existam opiniões sem o uso de palavras de sentimento e vice-versa, a polaridade de alguns termos ainda depende do contexto em que estão inseridos;
- Existem muitos domínios que são caracterizados pelo uso recorrente de sarcasmos e ironias, onde o sentido contido no texto é exatamente oposto ao sentimento declarado;
- A polaridade de conteúdo pode depender do observador, ou seja, o trecho de texto "As ações da empresa X subiram" pode ser considerada positiva para quem possui ações desta empresa, por outro lado negativa para quem deixou de investir nessas ações.

2.1.3 Sumarização

Para conseguirmos identificar as opiniões médias ou predominantes de um grupo de pessoas sobre um determinado tópico ou alvo, a opinião ou sentimento de apenas um usuário não é suficiente, sendo assim necessário analisar uma grande quantidade de opiniões (CARDIE, 2014). A criação de métricas e sumários para quantificar a variedade de opiniões encontradas sobre um mesmo alvo se faz necessário. O objetivo desta etapa consiste em criar métricas que representem o sentimento geral, as quais podem ser visualizadas ou servir como entradas para outras aplicações.

Em revisões de produtos, um sumário pode ajudar um consumidor a identificar os pontos positivos e negativos sobre um determinado produto, levando em consideração as experiências anteriores de outros usuários que expressaram suas opiniões sobre o produto. Por exemplo, no *Google Shopping* podemos encontrar recursos deste tipo, uma vez que a ferramenta extrai, analisa e agrega automaticamente revisões de produtos disponibilizados por diferentes lojas de comércio eletrônico. Outra forma de sumarização comum em

aplicações, resume-se em extrair de mídias sociais os sentimentos do público sobre uma determinada entidade, como marcas, políticos e celebridades, e com isso associá-los a informações geográficas ou temporais. Este tipo de abordagem se dá normalmente quando queremos saber o que as pessoas pensam sobre um alvo, dado um evento, por exemplo o lançamento de um novo produto no mercado.

2.2 Abordagens de Classificação de Polaridade

As abordagens de classificação podem ser divididas em quatro grupos, sendo eles: léxicas, utilizando dicionários de sentimentos; aprendizado de máquina, utilizando em sua maioria, técnicas de classificação ou regressão; estatística, onde são utilizadas técnicas para avaliar a co-ocorrência de termos; e semântica, onde definem-se a polaridade de palavras em função de sua proximidade semântica com outras polaridades conhecidas. Segundo Tsytsarau e Palpanas (2011), há uma predominância das duas primeiras abordagens citadas, sem que nenhuma técnica se sobressaia em termos de desempenho.

2.2.1 Abordagem Baseada em Dicionário

A abordagem baseada em dicionário é também conhecida como léxica ou linguística. Essa abordagem se caracteriza por utilizar dicionários de sentimentos, ou seja, compilações de palavras ou expressões de sentimentos associadas à respectiva polaridade. Uma das técnicas mais comuns utilizadas na abordagem linguística, é o da co-ocorrência entre alvo e sentimento, não levando em consideração a ordem dos termos dentro de um documento, nem suas relações léxico-sintáticas. Para a classificação do sentimento em um texto, basta que exista uma palavra de sentimento, onde sua polaridade é dada por um dicionário de sentimentos. Esse método é amplamente utilizado para o vínculo de um sentimento a uma entidade em uma sentença.

O método por co-ocorrência apresenta bons resultados quando o nível de análise textual é de granularidade pequena (TSYTSARAU; PALPANAS, 2011), pois a palavra detentora do sentimento está próxima à entidade que qualifica. Sendo assim, este método é usualmente utilizado em análises de nível de sentença, cláusula ou até em documentos com poucos caracteres, como um *tweet*, porém quando aplicada em um nível de granularidade maior, é necessário estabelecer algum tipo de média sobre as palavras de sentimento encontradas. A maioria dos léxicos existentes são dependentes de idioma e em sua maioria para a língua inglesa, entretanto existem léxicos disponíveis na língua portuguesa, como o OpLexicon (SOUZA; VIEIRA, 2011) e o SentiLex-PT (SILVA; CARVALHO; SARMENTO, 2012), sendo eles respectivamente para a língua portuguesa do Brasil e para português de Portugal.

2.2.2 Abordagem baseada em Aprendizado de Máquina

As abordagens baseadas em aprendizado de máquina possuem como principal objetivo, descobrir automaticamente regras gerais em grandes conjuntos de dados que permitam extrair informações contidas nos textos. De forma geral as técnicas de aprendizagem de máquina podem ser divididas em dois tipos, sendo elas: aprendizado supervisionado e aprendizado não supervisionado (TAN; STEINBACH; KUMAR, 2005).

Na área de mineração de opiniões ou análise de sentimentos, há um predomínio do uso de métodos supervisionados de aprendizagem de máquina, especificamente métodos de classificação e regressão. Nestas circunstâncias, o problema da classificação é dividido em dois passos, sendo eles: aprender um modelo de classificação sobre um corpus de treinamento rotulado com as classes consideradas previamente, por exemplo positivo e negativo; prever a polaridade de novas porções de texto com base no modelo obtido da primeira etapa. Dentre os algoritmos mais usados nesta área, podemos citar o *Support Vector Machine*, Naïve Bayes, *Maximum Entropy* e algoritmos baseados em redes neurais.

A qualidade do modelo resultante da etapa de aprendizagem é medido através de métricas como acurácia, ou seja, a capacidade do modelo prever corretamente, precisão, correspondente ao número de instâcias previstas corretamente em uma dada classe, ou revocação, correspondente ao número de instâncias de uma dada classe previstas na classe correta.

Um dos grandes problemas na utilização do aprendizado supervisionado para a tarefa de polaridade está relacionada a necessidade de dados rotulados previamente para treino, pois seu desempenho é afetado não só pela quantidade, mas também pela qualidade dos dados de treinos disponíveis.

Em trabalhos que envolvem revisões de produto, a classificação dada pelos usários em forma de notas é utilizada como rótulo para o texto correspondente (PANG; LEE; VAITHYANATHAN, 2002), entretanto esse recurso não está disponível em todos os outros domínios, neste caso as alternativas propostas costumam ser a utilização de métodos léxicos ou probabilísticos.

2.2.3 Abordagens Estatísticas e Semânticas

As abordagens estatísticas, por vezes chamadas de não supervisionadas (CARDIE, 2014), se baseiam no argumento de que palavras que traduzem opiniões frequentemente são encontradas juntas no corpus dos textos, ou seja, se a palavra ocorre mais frequentemente junto a palavras positivas/negativas no mesmo contexto, então é provável que seja positiva/negativa, entretanto se ocorre com uma frequência igual, a palavra deve ser neutra. Nesse cenário a polaridade de um termo desconhecido pode ser determinado calculando a co-ocorrência com uma palavra notadamente positiva/negativa, como por exemplo, “excelente” ou “péssimo”.

A abordagem semântica se assemelha bastante com a abordagem estatística, exceto o princípio das técnicas nesta categoria, consiste em palavras semanticamente próximas, onde devem possuir a mesma polaridade. A abordagem semântica também pode ser usada como complemento às outras abordagens, como forma de expansão ou de aquisição de vocabulário específico, na ausência de bons dicionários de sentimento.

2.3 Pré-Processamento de Dados

Postagens em redes sociais não se encontram de forma estruturada e são fortemente propensos a ruídos, imperfeições e inconsistências, dificultando a aplicação de algoritmos para análise de sentimentos. Por conta desse fator, a qualidade dos dados analisados poderá ser prejudicada, levando a resultados errados ou de baixa qualidade. O pré-processamento de dados é realizado para eliminar ou minimizar os problemas citados, com o objetivo de preparar os dados para a aplicação dos algoritmos de classificação. O pré-processamento de dados possui diferentes técnicas e etapas para realizar a limpeza dos dados, contudo nessa monografia, serão apresentados as técnicas utilizadas nos experimentos. Uma visão geral sobre as técnicas de pré-processamento empregadas nos trabalhos correlatos pode ser encontrada em Hu e Liu (2012).

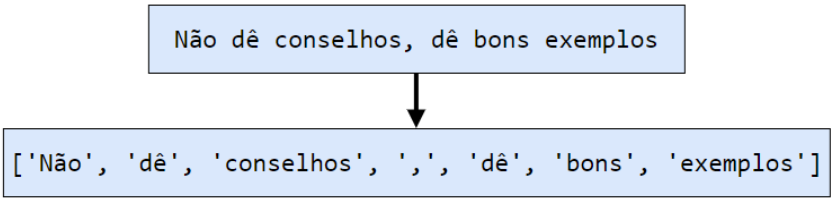
2.3.1 Tokenização

A tokenização é uma técnica para separar palavras, quebrando a sequência de caracteres de um texto encontrando os pontos onde uma palavra termina e outra começa, com isso as palavras identificadas são chamadas de *tokens*.

Segundo Palmer (2010), existe uma diferença na aplicação da tokenização nas chamadas linguagens delimitadas por espaço para as linguagens não segmentadas. No primeiro caso, de forma intuitiva percebe-se que a abordagem da tokenização consiste em separar as palavras por espaços em branco, muito utilizada para linguagens européias. Já na segunda abordagem, a separação das palavras necessita de informações léxicas e morfológicas adicionais, onde são utilizadas para separar palavras em chinês e tailandês.

Quanto a tokenização em liguaguens delimitadas por espaço, caracteres como vírgulas, pontos finais, ponto e vírgula são tratados como *tokens* separados, como é apresentado na Figura 1, entretanto existem casos onde esses caracteres devem ser incorporados a outro *token*. Um exemplo na língua portuguesa, seria o uso de pontos para abreviação de palavras como, Dr. ou Dra (doutor ou doutora) ou Av. (avenida).

Figura 1 – Exemplo de uma frase antes e depois de ser tokenizada.

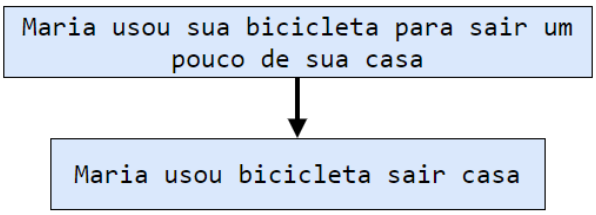


Fonte: Elaboração própria.

2.3.2 Remoção de *Stopwords*

A remoção de *stopwords* é uma técnica para limpeza de palavras que possuem um elevado número de ocorrências em uma sentença ou documento (BARION, 2008). Na maioria dos casos, nomes, advérbios, adjetivos, artigos, conjunções e preposições, são consideradas como *stopwords*, pois não possuem valores úteis ou relevantes para uma análise, sendo assim diminuindo o tamanho das estruturas de indexação. Na prática, é utilizado uma lista de palavras consideradas irrelevantes no idioma em que se deseja trabalhar, e então quando houver a ocorrência de uma palavra existente na lista, esse termo será eliminado. A Figura 2 exemplifica o processo de remoção de *stopwords* de uma frase.

Figura 2 – Exemplo de uma frase antes e depois de remover suas *stopwords*



Fonte: Elaboração própria.

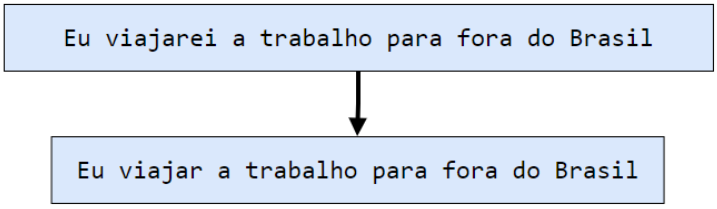
2.3.3 Lematização

A lematização é um processo onde os termos são resolvidos para seu lema, com o objetivo de agrupar diferentes palavras que possuem uma forma base em comum, retirando a conjugação verbal caso se trate de um verbo e altera os substantantivos e os adjetivos para o singular masculino, colocando a palavra na sua forma de dicionário. Por exemplo, palavras como “andar”, “anda”, “andou” e “andando” possuem como mesma forma base ou lema o termo “andar”.

Da mesma forma a lematização considera o contexto em que a palavra se encontra para resolver problemas onde palavras idênticas possuem significados diferentes

dependendo do contexto. A Figura 3 mostra um exemplo de uma frase após passar pelo processo de lematização, onde o termo **viajarei** foi alterado para **viajar**.

Figura 3 – Exemplo de uma frase antes e depois de ser lematizada.



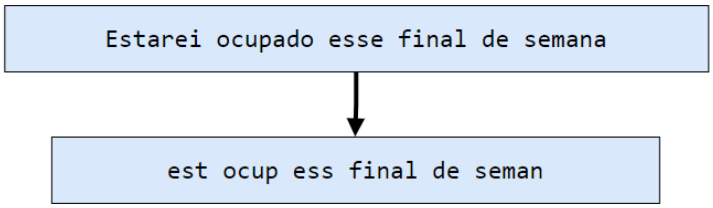
Fonte: Elaboração própria.

2.3.4 *Stemming*

O *stemming* é um processo utilizado para a limpeza de todas as intercorrências de palavras, mantendo somente a raiz do termo. Esse processo permite obter a raiz morfológica de uma palavra eliminando prefixos, sufixos, plurais e gerúndios com o objetivo de diminuir o número de termos a serem indexados nas estruturas de indexação.

A técnica de *stemming*, tem por base fazer uma representação da palavra, excluindo gêneros, tempos verbais específicos e diminutivos, por exemplo, as palavras química, químicas, químico e químicos são todos reduzidos a palavra químic durante o processo de *stemming*. A Figura 4 apresenta um exemplo mais claro da aplicação da técnica de *stemming* em uma frase.

Figura 4 – Exemplo de uma frase antes e depois de passar pelo processo de *stemming*.



Fonte: Elaboração própria.

2.4 Ponderação de Termos

Antes de iniciar qualquer tipo de tarefa de classificação, é necessário realizar a ponderação dos dados dos documentos que serão utilizados nos classificadores posteriormente. Em Aggarwal e Zhai (2012), os dados de um texto ou documento, podem ser representados de duas maneiras distintas. A primeira forma seria representar o dado diretamente como uma cadeia de termos/palavras, onde cada documento é uma sequência de

palavras. A segunda forma de representação, e a mais utilizada, consiste em uma técnica conhecida como *Bag Of Words* (BOW). Essa representação modela os termos/palavras de um documento em vetores númericos para em seguida, lidar com eles com operações algébricas lineares. Nesta monografia foram utilizadas duas técnicas de ponderação de termos que serão abordadas em seguida.

2.4.1 N-grama

O n-grama é uma cadeia de palavras, letras, sílabas ou fonemas de n itens interligados de um texto. Essa interligação de palavras, por exemplo, pode conter mais informação do que a palavra isolada em si, consequentemente, pode-se gerar atributos com maior poder de predição.

Por exemplo, ao considerar as palavras **São** e **Paulo** individualmente, pode-se agregar pouco conhecimento, uma vez que **São** pode ser referido como o verbo **ser** e **Paulo** é um nome próprio comum. Entretanto, o termo composto **São Paulo**, pode agregar mais informações sobre o texto, pois se refere a cidade ou estado de São Paulo. A Figura 5 exemplifica o funcionamento da técnica em uma frase, onde os retângulos representam os *tokens* da frase para unigramas, bigramas e trigramas.

Figura 5 – Exemplo de um unigrama, bigrama e trigrama de uma frase.

Fonte: (CHOLLET, 2010)

O n-grama pode ser aplicado em modelos de linguagem probabilística, para predizer o próximo item de um n-grama, com base nos itens já existentes. Um modelo de linguagem n-grama, utiliza das $n-1$ palavras imediatamente precedentes para estimar a probabilidade de ocorrência da próxima palavra. Esse modelo é geralmente utilizado para correção ortográfica de textos, reconhecimento de fala e tradução automática de textos. Na prática, o valor de n se limita a 2 ou 3, chamados de modelo bigrama e trigrama respectivamente (ZHAO, 1999).

2.4.2 TF-IDF

O método TF-IDF (*Term Frequency - Inverse Document Frequency*) é uma técnica estatística da área de PLN (Processamento de linguagem natural) com o objetivo de men-

surar a importância de um termo em um documento não estruturado ou semi estruturado (BAEZA-YATES; RIBEIRO-NETO, 2011). A abordagem consiste em atribuir um peso a cada termo de um documento a partir de sua frequência no próprio texto e em todos os documentos da base, assinalando assim sua importância.

O cálculo do TF-IDF pode ser separado em duas partes, sendo elas TF (*Term Frequency* ou Frequência do Termo) e IDF (*Inverse Document Frequency* ou Frequência Inversa do Documento). O TF, consiste na razão entre o número de ocorrências de um atributo ou termo dentro de um documento pelo número de termos no documento. Essa medida é definida pela Equação 2.1, onde $f(t_i, d_j)$ é a frequência do termo t_i no documento d_j.

$$tf(t_i, d_j) = f(t_i, d_j) \tag{2.1}$$

Na Tabela 1 é apresentado um exemplo utilizando a medida TF. Supondo que cada documento na tabela possua exatamente 100 termos, apenas para facilitação dos cálculos, no Documento 7 o termo **vacinação** recebe o valor 0,05 já que o termo ocorre 5 vezes, com isso tf(vacinação, Documento 7) $= 5/100$.

Tabela 1 – Tabela termo-frequência utilizando a medida TF.

	tristeza	**diversão**	**futebol**	**política**	**vacinação**	**família**
Documento 1	0,00	0,06	0,05	0,00	0,00	0,00
Documento 2	0,00	0,00	0,04	0,03	0,03	0,07
Documento 3	0,00	0,00	0,01	0,02	0,02	0,03
Documento 4	0,00	0,01	0,00	0,01	0,04	0,04
Documento 5	0,00	0,03	0,02	0,00	0,00	0,05
Documento 6	0,02	0,00	0,00	0,05	0,01	0,02
Documento 7	0,07	0,00	0,00	0,04	0,05	0,03

O IDF é utilizado para que os atributos ou termos que aparecem na maioria dos documentos tenham um peso de representação menor (BAEZA-YATES; RIBEIRO-NETO, 2011). Sendo assim, a ponderação IDF é inversamente proporcional ao logaritmo do número de documentos em que o termo aparece no número total N de documentos. A Equação 2.2 apresenta o cálculo do IDF, onde N e $n(t_i)$ representam o número total de documentos e o número de documentos em que o termo t_i aparece respectivamente.

$$idf(t_i) = \log \frac{N}{n(t_i)} \tag{2.2}$$

O cálculo do TF-IDF é determinado através do produto de TF, que representa a frequência do termo em um documento, com IDF, representando a relevância de um

termo em uma coleção de documentos. Dessa forma, a Equação 2.3 apresenta o cálculo da medida TF-IDF.

$$tfidf(t_i, d_j) = f(t_i, d_j) \times \log \frac{N}{n(t_i)} \tag{2.3}$$

Na Tabela 2 é apresentado um exemplo da medida TF-IDF. O atributo **tristeza** recebe o fator de ponderação $idf(\text{tristeza}) = \log \frac{7}{2} \simeq 0,544$ por aparecer em dois dos sete documentos (veja na Tabela 1). Com isso no Documento 7, o atributo **tristeza** recebe o valor de $tfidf(\text{tristeza, Documento 7}) = 0,07 \times 0,544 \simeq 0,038$. Os valores de todos os termos em todos os documentos são calculados da mesma maneira.

Tabela 2 – Tabela termo-valor utilizando a medida TF-IDF.

	tristeza	**diversão**	**futebol**	**política**	**vacinação**	**família**
Documento 1	0,000	0,022	0,012	0,000	0,000	0,000
Documento 2	0,000	0,000	0,010	0,004	0,004	0,005
Documento 3	0,000	0,000	0,002	0,003	0,003	0,002
Documento 4	0,000	0,004	0,000	0,001	0,006	0,003
Documento 5	0,000	0,011	0,005	0,000	0,000	0,003
Documento 6	0,011	0,000	0,000	0,007	0,001	0,001
Documento 7	0,038	0,000	0,000	0,006	0,007	0,002

3 Algoritmos de Classificação

Nesta seção serão apresentados os classificadores de sentimento estudados, empregados e analisados neste trabalho de conclusão de curso. A escolha dos algoritmos foi feita com base literatura estudada onde foram selecionados alguns dos algoritmos mais utilizados para estudos na área, entretanto existem diversos outros algoritmos para classificação de sentimentos na literatura.

3.1 *Random Forest*

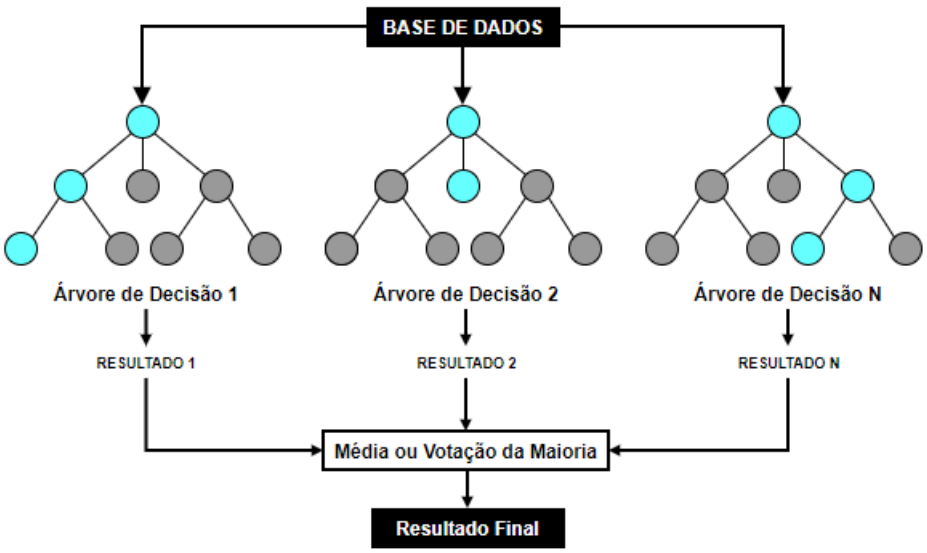
Random Forest, em português, Floresta Aleatória, é um algoritmo supervisionado de classificação baseado em árvores de decisão (*Decision Trees*). As árvores de decisão utilizam a tática de dividir para conquistar, ou seja, separar um problema complexo em problemas mais simples e repetir o processo para obter vários modelos diferentes, até que seja possível encontrar a solução do problema complexo, assemelhando-se a uma árvore. As árvores de decisão são geralmente utilizadas para solução de problemas de classificação e regressão.

As árvores de decisão, determinam regras para tomada de decisão, onde em cada nó da árvore, é verificada uma condição, caso seja atendida o fluxo segue por um ramo da árvore, caso contrário, segue por outro, indo até o próximo nó, até que a árvore seja completamente finalizada. O classificador *Random Forest*, segundo Breiman (2001) consiste na criação de uma coleção de árvores de decisão, estruturadas com k vetores aleatórios independentes e identicamente distribuídos, como é exemplificado na Figura 6. A criação das árvores se inicia a partir da definição do primeiro nó da árvore (nó raiz), onde será testada a primeira condição para dar origem aos dois primeiros ramos.

Cada nó interno da árvore corresponde a um teste sobre alguma característica da base de dados. A escolha deste primeiro nó será definida de forma aleatória (*random*) pelo algoritmo que irá escolher dois ou mais valores disponíveis na base de dados. Com isso, o algoritmo realiza cálculos com base nos dados selecionadas, para definir quais valores serão utilizados no nó raiz. Para a escolha do próximo nó, serão aplicados os passos anteriores, excluindo os dados selecionados anteriormente, e então o processo de escolha será repetido.

Para a criação de novas árvores, os processos citados são repetidos, entretanto por conta da aleatoriedade da escolha dos nós, essa nova árvore tende a ser diferente da primeira, tanto na seleção de amostras quanto na seleção de dados. Com isso ao final do algoritmo, para problemas de classificação, o resultado que ocorrer um maior número de vezes será escolhido.

Figura 6 – Funcionamento do algoritmo *Random Forest*.



Fonte: (TIBCO, 2021)

3.2 *Logistic Regression*

A *Logistic Regression*, ou Regressão Logística, é uma técnica estatística preditiva, com o objetivo de descrever dados e explicar a relação entre uma variável binária dependente e uma ou mais variáveis independentes. A Regressão Logística tem por função descrever a relação entre uma variável de resposta discreta, onde na maior parte dos casos, assume-se que a variável possua dois valores (valores binários), e uma ou mais variáveis independentes, usualmente chamadas de váriáveis explicativas ou preditoras.

O uso da Regressão Logística é conveniente em diversos cenários por conta de possibilitar a análise do efeito de uma ou mais váriáveis independentes (X_i) sobre uma variável binária (Y), onde essa pode assumir valores como 1 e 0, sim ou não, possui ou não possui. Com isso, o modelo tem o poder de descrever o valor provável de Y a partir da Equação 3.1 (FIGUEIRA, 2006):

$$P(Y = 1) = \frac{1}{1 + e^{-g(x)}} \tag{3.1}$$

Dado que $g(x) = B_0 + B_1X_1 + ... + B_pX_p$, os coeficientes $B_0, B_1, ..., B_p$ são estimados a partir do conjunto de dados, pelo método da máxima verossimilhança, encontrando assim uma combinação de coeficientes que aumenta a probabilidade da amostra ter sido observada. A Regressão Logística possui como característica o formato da letra S, isso ocorre por conta de sua curva logística possuir comportamento probabilístico ao se combinar determinados coeficientes e variar os valores de X. Por conta desta característica, a técnica da Regressão Logística é bastante generalista (MESQUITA, 2014), dando a ela

aspectos importantes, como:

- Caso $g(x) \to +\infty$, então $P(Y = 1) \to 1$;
- Caso $g(x) \to -\infty$, então $P(Y = 1) \to 0$.

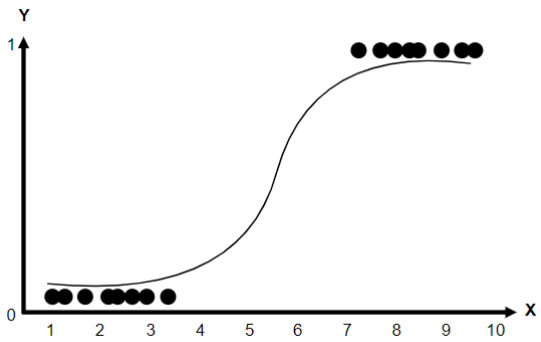
Da mesma maneira que é possível estimar diretamente a probabilidade de um evento ocorrer na Equação 3.1, a probabilidade de um evento não ocorrer pode ser estimado como na Equação 3.2:

$$P(Y = 0) = 1 - P(Y = 1) \tag{3.2}$$

Como a Regressão Logística estima probabilidades da ocorrência de eventos em intervalos de 0 e 1, como é apresentada na Figura 7, e usualmente é utilizada uma regra de classificação para discriminar um limite para esse intervalo, sendo essa regra:

- Se P(Y=1) > 0,5 então classifica-se Y=1;
- caso contrário classifica-se Y=0.

Figura 7 – Exemplo de curva logística.



Fonte: (NADEEM, 2021)

3.3 *Naive Bayes Classifier*

O algoritmo *Naive Bayes* é um classificador probabilístico baseado em conceitos como probabilidade condicional (veja na Seção 3.3.1), indepedência condicional (veja na Seção 3.3.2), distribuição conjunta de probabilidades e principalmente no Teorema de Bayes (veja na Seção 3.3.3).

3.3.1 Probabilidade condicional

A probabilidade condicional, é um conceito relevante aplicado ao Teorema de Bayes. Em Leon-Garcia e Leon-Garcia (c2008.), define-se a probabilidade condicional como sendo um segundo evento que ocorre após ocorrência de um primeiro evento, isto é, a probabilidade condicional indica a probabilidade de um evento B dada a ocorrência de um evento A, como apresentada na Equação 3.3.

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \tag{3.3}$$

Onde $P(B|A)$ é a probabilidade dos eventos A e B ocorrerem simultâneamente, ou seja, a probabilidade conjunta de A e B.

3.3.2 Independência condicional

A independência condicional, outro conceito utilizado no classificador *Naive Bayes*, é definido como dois eventos A e B são condicionalmente independentes dado um terceiro evento C. Isto é, sabendo-se que o evento C ocorre, saber a ocorrência do evento A não gera informações sobre a ocorrência do evento B. Logo saber a ocorrência de B também não gera informações sobre a probabilidade de ocorrência de A (LEON-GARCIA; LEON-GARCIA, c2008.), como apresentada na Equação 3.4.

$$P(A \cap B|C) = P(A|C)P(B|C) \tag{3.4}$$

Para o classificador *Naive Bayes*, a indenpedência condicional é utilizada nos valores dos atributos de uma classe, onde a classe pode ser representada pelo evento C e os atributos são representados pelos eventos A e B, dado na definição da Equação 3.4. Sendo assim, a independência condicional é dada como, o efeito do valor de um atributo sobre uma classe é independente dos valores dos outros atributos dessa mesma classe, isto é, todos os atributos da classe possuem a mesma importância para a classificação da mesma.

3.3.3 Teorema de Bayes

O classificador *Naive Bayes* é baseado essencialmente no Teorema de Bayes com o objetivo de encontrar a probabilidade a posteriori. O teorema determina que a probabilidade de um evento A ocorrer, não depende apenas da relação entre A e B, mas também da probabilidade de observar-se A independentemente de observar-se B (MITCHELL, 1997), dada pela Equação 3.5:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.5}$$

Onde P(A) e P(B) são as probabilidades a priori dos eventos A e B, $P(A|B)$ e $P(B|A)$ são as probabilidades a posteriori de A condicional a B e de B condicional a A respectivamente.

O intuito do Teorema de Bayes é alterar as probabilidades a priori levando em consideração novas evidências, a fim de obter probabilidades a posteriori. Portanto o classificador *Naive Bayes* assume que a presença ou ausência de uma característica particular está relacionada com a presença ou ausência de qualquer outro elemento, tendo em conta a classe variável.

Por exemplo, uma fruta pode ser considerada como uma maçã se for vermelha, redonda e com cerca de 5 cm de diâmetro. Um classificador *Naive Bayes* considera cada um desses atributos para contribuir de forma independente para a probabilidade de que esta é uma fruta maçã independente da presença ou ausência de outras características.

3.3.4 Algoritmo Naive Bayes

O objetivo do classificador *Naive Bayes* é encontrar $P(A|B)$, apresentada na Equação 3.5, que corresponde a probabilidade a posteriori da ocorrência da classe A dado o evento B. Para que seja possível chegar a probabilidade, é necessário calcular a probabilidade condicional $P(B|A)$, ou seja, a probabilidade de ocorrência do evento B dada uma classe A. Além disso, encontrar a probabilidade $P(A)$ a priori de ocorrência da classe A a partir dos dados de treinamento e a probabilidade $P(B)$ do evento B ocorrer dentro do conjunto de treinamento (DUDA; HART; STORK, 2000).

Em geral, o classificador *Naive Bayes* é utilizado para encontrar a classe com o máximo valor a posteriori A_{MAP} (do inglês, *Maximum A Posteriori*), apresentado na Equação 3.6:

$$\begin{aligned} A_{MAP} &= arg\max_i P(A_i|B) \\ &= arg\max_i \frac{P(B|A_i)P(A_i)}{P(B)} \\ &= arg\max_i P(B|A_i)P(A_i) \end{aligned} \tag{3.6}$$

Onde $argmax_i$ retorna a classe A_i com a maior probabilidade de estar associada a B. A probabilidade $P(B)$ pode ser desconsiderada no cálculo, pois seu valor é o mesmo para todas as classes.

Para encontrar a probabilidade da classe A ocorrer (P(A)), a priori, é preciso saber a frequência da classe em todo o conjunto de amostras associados a ela, dada pela Equação 3.7, onde N representa o total de amostras de treinamento e N_i o número de amostras

associadas a classe.

$$P(A_i) = \frac{N_i}{N} \tag{3.7}$$

Outro parâmetro necessário é a probabilidade ($P(B|A_i)$) do evento B dada a classe A, onde assume-se que os valores dos atributos do evento B são independentes entre si dada a classe A (DUDA; HART; STORK, 2000). Com isso é possível decompor a probabilidade $P(B|A_i)$ no produto $P(B_1|A_i)\times...\times P(B_d|A_i)$, onde B_j é o j-ésimo atributo do evento B, dado pela Equação 3.8:

$$P(B|A) = \prod_{j=1}^{d} P(B_j|A_i) \tag{3.8}$$

Logo o classificador conhecido como *Naive Bayes* é dado pela Equação 3.8 e pelo máximo valor a posteriori, apresentado na Equação 3.6.

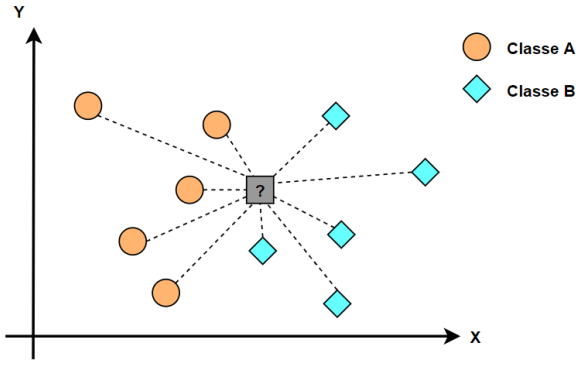
3.4 *K-Nearest Neighbor (K-NN)*

K-Nearest Neighbors ou K-Vizinhos Mais Próximos em português, é um algoritmo de classificação supervisionado baseado em técnicas de predição através da proximidade entre os dados (CARVALHO et al., 2011). Essas técnicas possuem a premissa de que dados similares tendem a se concentrar na mesma região no espaço de entrada, consequentemente dados dissimilares estarão mais distantes entre si.

K-NN é um algoritmo de aprendizagem lenta, conhecido como algoritmo preguiçoso (*lazy*). Isto ocorre pelo fato da técnica não precisar de dados de treinamento para gerar um modelo. Sendo assim, a fase de treino do algoritmo é realizada com mais velocidade, dado que ele apenas memoriza os dados que serão utilizados na fase de teste. Em contra partida, sua fase de testes é mais custosa e lenta, pois para que se possa classificar um novo objeto, é necessário efetuar o cálculo de sua distância com relação a todos os outros objetos do espaço de entrada.

O algoritmo de classificação K-NN é uma ampliação do algoritmo 1-NN. O funcionamento desse algoritmo é bastante simples, para se classificar um novo dado no espaço de entrada representado pelo quadrado cinza na Figura 8, é apenas necessário que os dados de treinamento já rotulados estejam memorizados, este dados estão representados pelos losangolos e círculos, cada um representando duas classes distintas. Com isso, é possível calcular a distância entre o novo dado e todos os dados de treinamento rotulados. O novo dado então, irá receber o mesmo rótulo/classe do dado de treinamento que possuir a menor distância calculada entre todos os dados de treinamento.

Figura 8 – Exemplo de classificação de um novo dado no algoritmo 1-NN em um espaço bidimensional.



Fonte: (INFERIR, 2019)

Existem vários métodos para se calcular a distância do novo dado no espaço de entrada para os dados de treinamento rotulados, dos quais os métodos mais utilizados são as distâncias Euclidiana, Manhattan, Minkowski e Chebyshev. As Equações 3.9, 3.10, 3.11 e 3.12 apresentam o cálculo das distância respectivamente listadas e extraídas de Carvalho et al. (2011). Em todas as equações abaixo, $d(x_i, x_j)$ representa a distância de um dado novo x_i^l para um outro dado x_j^l, que correspondem aos valores da coordenada l.

- A **Distância Euclidiana** é calculada como a raiz quadrada da soma das diferenças quadráticas entre um novo ponto x_i e um ponto existente x_j;

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{d} (x_i^l - x_j^l)^2} \tag{3.9}$$

- A **Distância de Manhattan** é calculada somando o valor absoluto da diferença entre um novo ponto x_i e um ponto existente x_j;

$$d(x_i, x_j) = \sum_{l=1}^{d} \left| x_i^l - x_j^l \right| \tag{3.10}$$

- A **Distância de Minkowski** é considerada uma generalização das distância de Manhattan e Euclidiana, onde a escolha de diferentes valores para p, com $1 \leq p \leq \infty$, define variações para a métrica. Se o valor de p for igual a 1, então o resultado será o mesmo da distância de Manhattan e se o valor de p for igual a 2, então o resultado será o mesmo da distância Euclidiana;

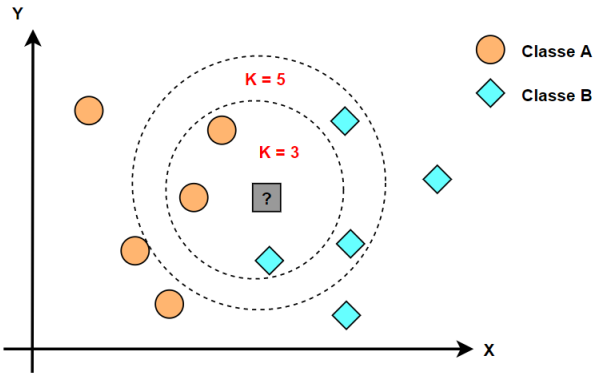
$$d(x_i, x_j) = \sqrt[p]{\sum_{l=1}^{d} \left| x_i^l - x_j^l \right|^p} \tag{3.11}$$

- A **Distância de Chebyshev** é calculada pela diferença máxima entre quaisquer valores da coordenada l entre um novo ponto x_i e um ponto existente x_j;

$$d(x_i, x_j) = \max_{1 \leq l \leq d} \left| x_i^l - x_j^l \right| \tag{3.12}$$

O K-NN se difere do 1-NN por levar em consideração não apenas um vizinho mais próximo, mas sim K vizinhos. Dessa forma, a classificação de um novo objeto no espaço de entrada, será semelhante a apresentada anteriormente, com a diferença de que o novo objeto irá receber o rótulo/classe que mais ocorrer entre todos os K vizinhos mais próximos, ou seja, através do cálculo da moda do rótulo/classe dos K vizinhos mais próximos. Em problemas binários, onde só existem dois tipos de classes possíveis, o número de K vizinhos deve ser ímpar, para que não haja empate no momento de classificação, como é exemplificado na Figura 9.

Figura 9 – Exemplo de classificação de um novo elemento no algoritmo K-NN.



Fonte: (NADEEM, 2021)

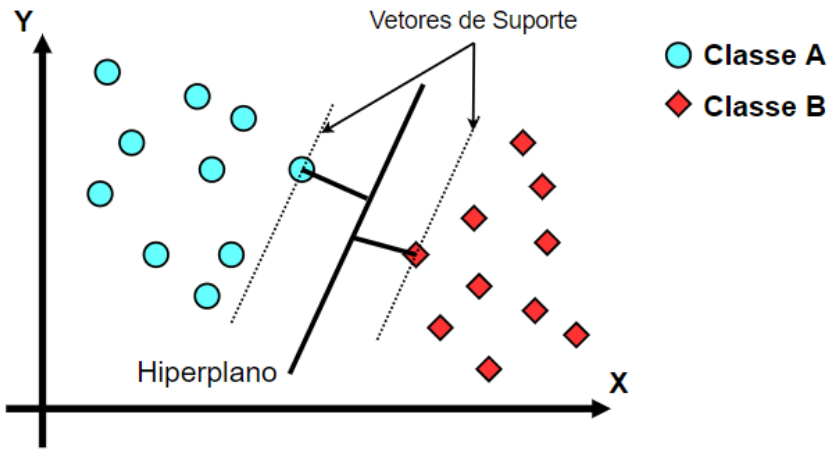
No exemplo da Figura 9, ao se considerar os três vizinhos mais próximos ($K = 3$) do novo elemento no espaço de entrada, o elemento irá receber o rótulo/classe A. Entretanto ao se considerar os cinco vizinhos mais próximos ($K = 5$), o elemento irá receber o rótulo/classe B.

3.5 *Support Vector Machine (SVM)*

Support Vector Machine, ou Máquina de Vetores de Suporte em português, é um algoritmo supervisionado utilizado em problemas de classificação e regressão, com o objetivo de classificar um conjunto de dados mapeado, em um espaço de características multidimensional utilizando uma função *kernel* (VAPNIK, 1995). O algoritmo representa cada dado como um ponto em um espaço n-dimensional, onde n é o número de objetos disponíveis no conjunto e o valor de cada objeto, representa o valor de uma coordenada.

O algoritmo determina a classificação de um dado, a partir da definição de um hiperplano que melhor diferencia as classes. Além disso, o algoritmo também determina uma margem limite de cada classe conhecida como vetor de suporte, onde não há nenhum ponto dentro dessa margem. Isso pode ser visto na Figura 10, onde o hiperplano está limitando a área de classificação da classe A e da classe B.

Figura 10 – Exemplo de hiperplano unidimensional dividindo duas classes distintas e vetores de suporte delimitando as margens.



Fonte: (ICHI.PRO, 2020)

Entretanto, existem problemas onde nem sempre é possível realizar a separação dos dados facilmente. Para isso, o algoritmo necessita encontrar a separação ótima dos dados, conhecido como truque de *kernel*. Esse método aumenta a dimensão do problema, fazendo com que o hiperplano seja traçado em um espaço de dimensão superior, sendo assim, é possível converter um problema não separável em um problema separável. Usualmente essa tática é utilizada para realizar a separação dos dados de forma não linear.

3.6 Avaliação

Métricas de avaliação são fundamentais para avaliar qualquer método de classificação proposto para um conjunto de dados, e assim determinar a qualidade do modelo escolhido. Usualmente uma métrica de avaliação não é melhor que outra, sendo assim o que deve ser levado em consideração é a escolha correta de cada técnica para a resolução do problema trabalhado (CARVALHO et al., 2011).

Para que seja possível compreender as Métricas de Avaliação, será necessário abordar alguns tópicos sobre técnicas de avaliação de performance dos algoritmos de classificação, sendo elas a Validação Cruzada e a Matriz de Confusão, uma vez que essas estão sendo utilizadas para os classificadores de sentimentos abordados no trabalho descrito nesta monografia.

3.6.1 Validação Cruzada

O método de validação cruzada, ou *cross-validation*, consiste em uma técnica de divisão de dados para treinamento e teste de algoritmos de classificação. A abordagem consiste em subdividir um conjunto de dados inicial em r subconjuntos de tamanho aproximadamente igual (CARVALHO et al., 2011).

O objetivo dessa subdivisão é treinar o algoritmo de classificação com $r - 1$ subconjuntos, e os dados restantes são utilizado para teste, repetindo-se o processo r vezes, alternando os subconjuntos de treino e teste, como é exemplificado na Figura 11. Com isso, o desempenho de um classificador é dado a partir da média dos resultados de cada iteração da validação cruzada.

Figura 11 – Exemplo de validação cruzada para 5 subconjuntos.

Fonte: (SANTANA, 2020)

3.6.2 Matriz de Confusão

A matriz de confusão é uma técnica para visualização de performance de um algoritmo de classificação, onde são apresentadas as quantidades de classificações corretas contra as classificações incorretas de cada classe em um conjunto de dados. A partir dessa matriz, é possível identificar quais classes um algoritmo de classificação possui maior facilidade e dificuldade de determinar. Em Carvalho et al. (2011) define-se a dimensão de uma matriz de confusão M_c como $k \times k$, onde k representa as classes, e um elemento m_{ij} representa o número de exemplos de uma classe i classificados na classe j na matriz de confusão M_c.

Para problemas de duas classes ilustrado na Figura 12, via de regra são definidas as classes como negativa (-) e positiva (+), logo são obtidos quatro valores representando as quantidades de classificações corretas e incorretas do modelo, sendo eles:

Figura 12 – Matriz de confusão de duas classes.

		Classe predita	
		+	-
Classe verdadeira	+	VP	FN
	-	FP	VN

Fonte: (CARVALHO et al., 2011)

- Verdadeiros Positivos (VP): representa o número total de exemplos classificados para a classe positiva corretamente;
- Verdadeiros Negativos (VN): representa o número total de exemplos classificados para a classe negativa corretamente;
- Falsos Positivos (FP): representa o número total de exemplos classificados para a classe positiva incorretamente;
- Falsos Negativos (FN): representa o número total de exemplos classificados para a classe negativa incorretamente.

3.6.3 Métricas

Na literatura é possível encontrar um acervo de métricas de avaliação, levando em conta a matriz de confusão apresentada na seção 3.6.2. A próxima seção apresenta as métricas utilizadas no trabalho descrito aqui baseado em Carvalho et al. (2011).

A **acurácia** é uma métrica dada a partir da soma dos acertos do modelo, ou seja, Verdadeiros Positivos e Verdadeiros Negativos, dividida pelo número total de dados classificados na matriz. Essa métrica é utilizada para definir o números de acertos do modelo, independente da classe (veja na Equação 3.13).

$$acurácia = \frac{VP + VN}{n} \tag{3.13}$$

A **precisão** é uma métrica calculada a partir do número de classificações corretas para a classe positiva, dividido pelo total de dados classificados como positivo. Essa métrica é utilizada para definir a proporção de acertos do modelo, ou seja, dos dados classificados como positivo, quantos são realmente positivos (veja na Equação 3.14).

$$precisão = \frac{VP}{VP + FP} \tag{3.14}$$

A **revocação** (ou *recall*) é uma métrica onde, a partir da razão entre o número de classificações corretas para a classe positiva, e total de dados que são de fato positivos,

busca-se encontrar a quantidade de dados corretamente classificados como positivos, essa métrica também é conhecida por ser a taxa de verdadeiros positivos (TVP) (veja na Equação 3.15).

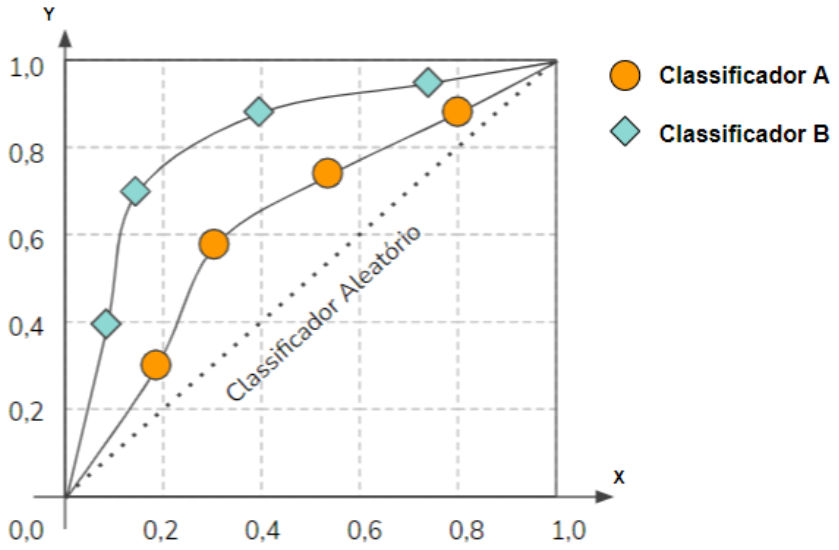
$$revocação = TVP = \frac{VP}{VP + FN} \tag{3.15}$$

A ***F1-score*** leva em consideração a precisão e a revocação para calcular a média harmônica entre elas. Essa métrica tende a demonstrar a qualidade de um modelo, pois um modelo com alta precisão, ou seja, capaz de acertar as predições, e alta revocação, isto é, capaz de recuperar os exemplos da classe, aumentam o valor do *F1-score*. Entretanto, caso a precisão ou a revocação tendam a 0, o valor dessa métrica também será reduzido (veja na Equação 3.16).

$$F1 - score = 2 \times \frac{precisão \times revocação}{precisão + revocação} \tag{3.16}$$

A **Curva ROC** é uma métrica de avaliação de classificadores para comparação de algoritmos em problemas binários. A curva ROC (*Receiving Operating Characteristcs*) é plotada em um gráfico onde a taxa de verdadeiros positivos (TVP ou revocação) corresponde ao eixo Y e a taxa de falsos positivos (TFP) corresponde ao eixo X. A Figura 13 apresenta a Curva ROC de dois classificadores em um espaço bidimensional.

Figura 13 – Curva ROC de dois classificadores.



Fonte: (BROWNLEE, 2018)

O espaço do gráfico ROC é compreendido em um intervalo de valores 0 e 1, possuindo uma diagonal dos pontos (0, 0) e (1, 1) representando os classificadores aleatórios. O desempenho de um classificador é caracterizado por um ponto no espaço do gráfico

ROC, representado pelos losangolos e círculos na Figura 13, onde busca-se sempre estar mais próximo do ponto (0, 1), conhecido como *céu ROC*.

Por outro lado, classificadores com o desempenho próximo ao ponto (1, 0) conhecido como *inferno ROC*, tendem a ser piores que os classificadores aleatórios. Outras características do gráfico ROC seriam os pontos (0, 0), representando a tendência de um classificador rotular sempre como negativo, e o ponto (1, 1) onde o classificador tende a rotular sempre como positivo.

Habitualmente a qualidade de um modelo é calculado através da área compreendida abaixo da curva ROC, conhecida como AUC (*Area Under ROC Curve*), onde quanto mais próxima a curva estiver do *céu ROC*, ou seja, no ponto (1, 0), maior será a sua área sob a curva no espaço do gráfico ROC, consequentemente, melhor será o desempenho do classificador.

4 Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados com o mesmo tema ou que se utilizam de abordagens semelhantes as que foram abordadas no trabalho de conclusão de curso descrito nesta monografia.

A análise de sentimento é um campo de estudo com o objetivo de analisar partes de textos a fim de determinar a atitude, emoção, opinião, avaliação ou sentimento de um indivíduo sobre determinado assunto ou alvo (PANG; LEE, 2008). Desde que a análise de sentimentos ganhou espaço e começou a atrair a atenção da indústria e da academia, diversas ferramentas e métodos foram propostos para identificação de sentimentos em diversos ambientes baseando-se em diferentes estratégias.

O trabalho apresentado em Becker, Moreira e dos Santos (2017) realiza uma série de experimentos para investigar a adequação da classificação de emoções multilíngue baseadas em tradução para classificação de sentimentos. Nesse trabalho são feitos três tipos de análises, através de quatro experimentos utilizando técnicas de aprendizado supervisionado, sendo essas análises:

- Verificar se a tradução automática de textos, afeta o desempenho da tarefa de classificação de emoção;
- Verificar se a combinaçao de palavras em diferentes línguas tem potencial de melhorar os resultados da etapa de classificação de emoção;
- Mostrar se os efeitos da tradução automática dos textos são diferentes comparados as tarefas de classificação de emoção por polaridade.

Com relação aos experimentos, o trabalho os abordou da seguinte maneira. O primeiro experimento classifica as emoções dos textos em idiomas diferentes utilizando tradução automática, com o objetivo de verificar se é possível adaptar uma abordagem de classificação de emoções multilíngue. Outros dois experimentos investigam se combinar textos em diferentes idiomas melhora a qualidade da classificação de emoção através de comparações de modelos testados em um único idioma com modelos testados em vários idiomas. O quarto experimento verifica se o comportamento dos algoritmos usados são semelhantes, através de um conjunto de textos contendo a polaridade e emoção comparado com os resultados da classificação de emoção e polaridade.

O trabalho apresentado em Tausczik e Pennebaker (2010), cria uma ferramenta para análise de texto que, dado um texto, estima seus componentes emocionais, cogniti-

vos e estruturais baseando-se em um dicionário de palavras categorizadas. Além disso a ferramenta detecta a pontuação de polaridade positiva e negativa de um texto.

Em Skowron et al. (2016), é proposto um novo método de estudo dos traços de personalidade dos usuários de redes sociais. O trabalho realiza os estudos a partir dos textos e imagens produzidos pelos usuários participantes, em duas redes sociais distintas, sendo elas, o *Twitter* e o *Instagram*. O problema abordado no trabalho, consiste na afirmação de que a grande parte dos estudos encontrados na literatura, utilizam apenas uma rede social para estudar os traços de personalidade dos usuários. Com isso, são utilizados dados de duas redes sociais, para obter uma diminuição nos erros de previsão de cada traço de personalidade.

A coleta de dados neste trabalho, se deu a partir de um questionário, onde primeiramente, era necessário haver o consentimento dos usuários para a realização da pesquisa. O questionário então foi utilizado para inferir cinco traços de personalidade de cada usuário, sendo pontuados de 1 a 5. Finalizada essa etapa, os usuários com menos de 30 imagens no *Instagram* ou menos de 30 *tweets*, foram filtrados, restando ao final 62 usuários para o estudo, sendo todos falantes da língua inglesa e localizados nos Estados Unidos.

No trabalho foi utilizado método de regressão *Random Forest Regression*, e então calculada a média sobre as árvores de decisão de regressão para chegar a um modelo de características dos traços de personalidades. Foram feitas 5 execuções independentes utilizando validação cruzada divindo os dados em 10 subconjuntos (10-*fold*)

Como resultado, o trabalho Skowron et al. (2016) chegou a uma nova abordagem de reconhecimento de personalidade de um usuário de redes sociais, de fácil adaptação a outras redes socias, com resultados animadores para o potencial da análise de dados de usuários gerados em diferentes redes sociais.

Em Quercia et al. (2011), foi feito um estudo para correlacionar o comportamento do mundo real com o comportamento dentro das redes sociais. O estudo foi realizado na rede social *Twitter*, com a justificativa de que não há tantos trabalhos que analisam as personalidades dos usuários, na mesma escala com que é feito na rede social do *Facebook*.

O trabalho coletou dados da personalidade de 335 usuários através de um aplicativo do *Facebook* chamado *myPersonality*, onde os usuários da plataforma, podem fazer um teste de personalidade gratuitamente e então compartilharem suas pontuações nas redes sociais, incluindo o *Twitter*.

A partir dos dados dos traços de personalidades dos usuários, os autores realizaram uma análise de regressão utilizando árvores de decisões, onde para cada traço de personalidade, o algoritmo passou por 10 interações independentes e separado em 10 subconjuntos (10-*fold*) para validação cruzada.

A conclusão do trabalho mostrou que todos os usuários são emocionalmente está-

veis e que a maioria deles são extrovertidos. Além disso, foi possível prever com facilidade e efetividade a personalidade de um usuário, a partir de seus dados públicos, como o número de seguidores, número de pessoas que o usuário segue e o número de *tweets* do usuário que foi salvo por outros usuários. Por conta disso, o trabalho ainda deixa em aberto discussões sobre mecanismos e proteção a privacidade nas redes sociais.

Em Wang et al. (2012), são utilizadas estratégias de aprendizado de máquina para extrair e analisar o sentimento do público sobre as eleições presidenciais dos EUA em 2012. Nesse trabalho é feita uma abordagem que combina processamento de dados em tempo real do Twitter e modelagem de sentimento estatística informada pelo autor a fim de compreender a cultura das práticas políticas do público na plataforma.

Em Hemmatian e Sohrabi (2019), é apresentada uma revisão completa sobre análise de sentimento e mineração de opinião. Nesse artigo são apresentados diferentes métodos de classificação de sentimento comparando suas vantagens e desvantagens. O trabalho explica, categoriza, resume e compara as técnicas de classificação de sentimento propostas na literatura e as divide em dois principais grupos, sendo eles:

- Aprendizado de máquina: onde são apresentados três tipos de técnicas utilizadas para classificar os sentimentos (supervisionada, semi supervisionada e não supervisionada);
- Baseadas em léxico: onde são apresentados pelo autor dois tipos de abordagens (baseada em dicionário e baseada em corpus).

Ao final do artigo são apresentados os critérios de avaliação de desempenho dos classificadores de sentimentos, que consideram o número de resultados corretos ou incorretos diagnosticados correta ou incorretamente, variando entre 0 e 1, onde quanto mais próximo de 1, melhores os resultados.

Um artigo exploratório sobre as técnicas tradicionais de mineração de texto e processamento de linguagem natural, com o objetivo de auxiliar na extração e classificação de padrões de comportamentos nas informações compartilhadas nas redes sociais pode ser visto em Lima e Castro (2019). O trabalho enfatiza a importância de extrair os dados dos perfis dos usuários nas redes sociais de forma passiva, por conta de um provável viés que um questionário proporcionaria ao usuário, sabendo que está sendo avaliado explicitamente. A pesquisa se utiliza da rede social *Twitter* como base de estudo, por conta da plataforma disponibilizar seus dados abertamente para coleta e análise. Com isso, esses dados são mapeados utilizando os tipos psicológicos de MBTI[1] (*Myers–Briggs Type Indicator*) e os Temperamentos de Keirsey[2].

[1] MBTI é marca registrada da *Consulting Psychologist Press, Inc.*EUA.
[2] Disponível em: <http://www.keirsey.com>

O trabalho realiza uma série de avaliações de classificadores a fim de mapear o comportamento dos usuários de redes sociais através de suas postagens no *Twitter*, e com isso criar uma estrutura denominada *Temperament Classification Framework* (TECLA). A estrutura foi avaliada usando um conjunto de mais de um milhão de *tweets* de cerca de 1.500 usuários e cinco algoritmos de classificação foram avaliados, sendo eles: *Naive Bayes*, *Support Vector Machine*, Árvore de Decisão, *Multilayer Perceptron* e *KNN*.

No trabalho de conclusão de curso descrito aqui, visou a utilização dos algoritmos de classificação *Naive Bayes*, *Support Vector Machine*, *Random Forest*, *Logistic Regreesion* e *KNN*, por serem alguns dos classificadores mais estudados e empregados para a classificação e predicação de sentimento, como apresentado nos trabalhos relacionados citados, assim como a escolha das métricas de avaliação foi baseada nas métricas mais utilizadas na literatura correlata.

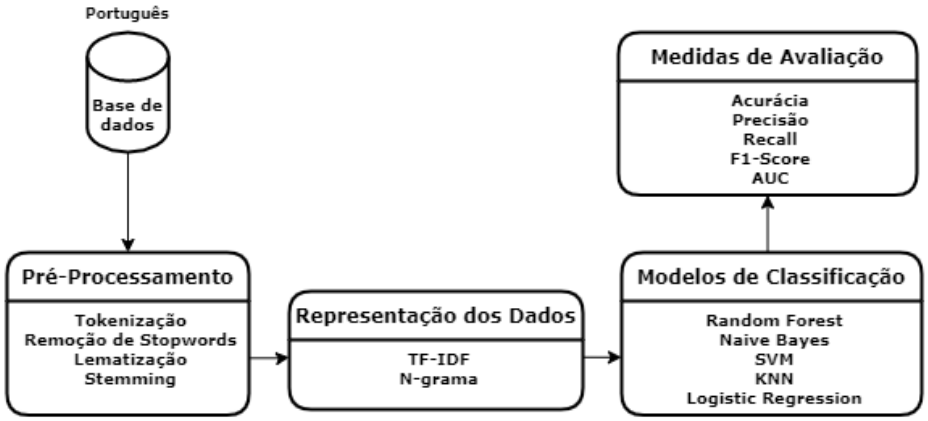
5 Experimentos e Resultados

Nesse capítulo, são apresentados os experimentos realizados nessa monografia, assim como os resultados obtidos. A motivação por trás dos experimentos é observar o desempenho de alguns algoritmos que utilizam diferentes técnicas de classificação, e assim determinar as vantagens e desvantagens do uso deles para a análise de sentimentos em dados de redes sociais.

5.1 Metodologia

Nesta seção são apresentados os métodos aplicados em cada etapa do desenvolvimento do trabalho descrito nesta monografia. As etapas consistem no processamento e preparação dos dados para a realização da análise de sentimentos, a representação dos dados pré-processados, assim como a aplicação dos algoritmos de classificação e seus processos de treinamento e aplicação das medidas de avaliação do desempenho dos classificadores utilizados. A Figura 14 exemplifica as principais etapas da metodologia empregada neste trabalho de conclusão de curso a fim de avaliar os classificadores para análise de sentimento em dados de redes sociais.

Figura 14 – Etapas da metodologia para avaliação dos classificadores



5.1.1 Base de dados

Para a análise dos classificadores de sentimentos, foi escolhida uma base de dados na língua portuguesa da rede social *Twitter* intitulada *Portuguese Tweets for Sentiment Analysis*, disponibilizada na plataforma *Kaggle*[1]. Optou-se por utilizar uma base de dados pública semelhante a base de dados que está sendo coletada no projeto maior no qual

[1] Disponível em <https://www.kaggle.com/augustop/portuguese-tweets-for-sentiment-analysis>.

o trabalho de conclusão de curso está relacionado (veja na Seção 1), pois a etapa de coleta dos dados está em andamento. O conjunto de dados conta com 50 mil registros de postagens de usuários (*tweets*), obtidos através da API[2] disponibilizada pela própria rede social no ano de 2018. Essa base de dados foi escolhida por conta de seu grande volume de dados já rotulados com os sentimentos 1 e 0, POSITIVO e NEGATIVO respectivamente, o que facilita as fases de treinamento e testes dos modelos de classificação. A Figura 15 apresenta a nuvem de palavras positivas e negativas da base de dados pública estudada.

Figura 15 – Nuvem de palavras positivas e negativas da base de dados

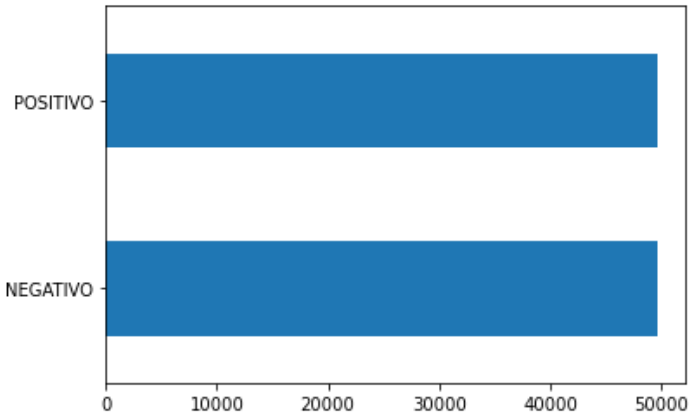
Além disso, a distribuição de *tweets* rotulados como POSITIVO e NEGATIVO está balanceada, isto é, o conjunto de dados possui 25 mil *tweets* positivos e 25 mil *tweets* negativos. Entretanto, após a remoção dos *tweets* repetidos existentes na base de dados, o conjunto ficou com 24.638 *tweets* positivos e 24.811 *tweets* negativos. A Figura 16 apresenta um gráfico com a distribuição de *tweets* da base, obtido através do uso da biblioteca *Pandas*[3] na liguagem de programação *Python*[4].

5.1.2 Pré-Processamento de Dados

O primeiro passo no pré-processamento dos dados foi a remoção dos *tweets* repetidos na base de dados estudada e a conversão dos textos para minúsculo, para que palavras escritas de formas diferentes sejam reconhecidas como iguais. Em seguida foi feito uma primeira limpeza de informações irrelevantes contidas em cada *tweet*, como os *links* de páginas *web*, nomes de usuários na rede social, identificados pelas palavras iniciadas com

[2] Disponível em: <https://developer.twitter.com/en/docs>.
[3] A biblioteca *Pandas* pode ser acessada em: <https://pandas.pydata.org/>.
[4] *Website* oficial da linguagem *Python*: <https://www.python.org/>

Figura 16 – Distribuição de *tweets* POSITIVOS e NEGATIVOS da base de dados



o caractere @ e as *hashtags*, escritas com o símbolo #, que são utilizadas para indexar tópicos ou palavras no *Twitter*[5]. A Figura 17 apresenta cinco *tweets* retirados aleatoriamente antes da etapa de pré-processamento dos dados.

Figura 17 – Cinco *tweets* antes da etapa de pré-processamento.

```
0   Porque simplesmente as coisas que me deixam chateada são aquelas coisas que eu quero mesmo muitooo ver e me iludir :) http
s://t.co/zEiHgH2oJz
1   A partir de segunda começa uma nova rotina, e assim dá perfeitamente para consolidar com a escola de condução :p
2   Meu amigo parece estar apaixonado por vc — é recíproco! :) https://t.co/FLnOcbarXY
3   @Goncla_Silva és um gajo bacano, vê lá se não me aleijas mais, espero que tu e a Silva sejam muito felizes :))
4   tou a ver videos de cabrinhas bebés :D
```

O próximo passo foi aplicar as técnicas de pré-processamento de dados apresentadas na Seção 2.3, iniciando pela técnica de tokenização, com o objetivo de converter os *tweets* em *tokens* e assim realizar com mais facilidade a segunda limpeza de *tokens* desnecessários. Foi utilizado o *word-tokenize*, um módulo do *NLTK*[6] para dividir frases em *tokens*. A Tabela 3, mostra o processo de tokenização de um *tweet* da base de dados estudada.

Tabela 3 – Tabela comparativa entre um *tweet* da base de dados e sua versão tokenizada.

***Tweet* Original**	***Tweet* Tokenizado**
Como me deixa feliz assistir a esse vídeo!	[’Como’, ’me’, ’deixa’,’feliz’, ’assistir’, ’a’, ’esse’, ’vídeo’, ’!’]

Com os *tweets* tokenizados, iniciou-se a tarefa de remover as *stopwords* indesejadas de cada sentença, com isso reduzindo o número de palavras que serão indexadas pelos classificadores nas próximas etapas. Nesta etapa foi utilizado o módulo *NLTK* que possui uma lista pronta das *stopwords* na língua portuguesa. A Tabela 4 apresenta o mesmo *tweet* da Tabela 3 sem os termos que não agregam na sentença.

[5] Disponível em: <https://help.twitter.com/pt/using-twitter/how-to-use-hashtags>.
[6] Documentação da biblioteca *NLTK* em: <https://www.nltk.org/index.html>.

Tabela 4 – Tabela comparativa entre um *tweet* da base de dados e sua versão após remoção de *stopwords*.

***Tweet* Original**	***Tweet* sem stopwords**
Como me deixa feliz assistir a esse vídeo!	feliz assistir vídeo

Logo em seguida, os *tweets* da base de dados passam pelas etapas de lematização e *stemming*. Cada palavra do *tweet* será resolvida para seu lema, ou seja, cada termo da sentença será deflexionado para sua forma de dicionário. Isso é necessário para que seja feito um agrupamento das formas flexionadas de uma palavra, e assim conseguir analisá-las como um único item. Para fazer a lematização, foi utilizado o módulo *Spacy*[7] que possui a funcionalidade de identificar o lema de uma sentença para a língua portuguesa. A Tabela 5 apresenta um *tweet* da base de dados que passou pelas etapas de tokenização, remoção de *stopwords* e foi lematizado.

Tabela 5 – Tabela comparativa entre um *tweet* da base de dados e sua versão lematizada.

Tweet Original	**Tweet Lematizado**
A busca de Deus é a busca da alegria. O encontro com Deus é a própria alegria. Bom dia! :)	buscar deus buscar alegria encontrar deus próprio alegria

A última etapa do pré-processamento de dados, foi a aplicação da técnica de *stemming*, para que sejam removidos os tempos verbais, gêneros e diminutivos de cada palavra do *tweet* e assim diminuindo ainda mais o número de termos que serão indexados pelos classificadores. Nesta etapa foi utilizado o módulo *SnowballStemmer* também do *NLTK*, que possui o recurso de *stemming* para a língua portuguesa. A Tabela 6 apresenta um *tweet* que passou pelas etapas de tokenização, remoção de *stopwords*, lematização e *stemming*.

Tabela 6 – Tabela comparativa entre um *tweet* base de dados e sua versão após processo de *stemming*.

***Tweet* Original**	***Tweet* após *stemming***
A busca de Deus é a busca da alegria. O encontro com Deus é a própria alegria. Bom dia! :)	busc deus busc alegr encontr deus própr alegr

Após todas as etapas de pré-processamento citadas anteriormente, a base de dados está pronta para a etapa de representação dos dados e aplicação dos modelos de classificação, previstos na metodologia (veja Figura 15). Abaixo a Figura 18 apresenta os mesmos *tweets* extraídos aleatoriamente da Figura 17, após passarem pelas técnicas de pré-processamento realizadas no trabalho de conclusão de curso descrito aqui.

[7] Disponível em: <https://spacy.io/>.

Figura 18 – Cinco *tweets* depois da etapa de pré-processamento.

```
0    simples  deix  chat  muitoo  ilud
1    comed  rotin  perfeit  consolid  escol  conduçã
2    apaixon  recíproc
3    gaj  bacan  ver  aleij  esper  silv  feliz
4    tou  vid  cabr  beb
```

5.1.3 Aplicação dos modelos de classificação

Para que os algoritmos de classificação sejam capazes de interpretar os *tweets*, é necessário realizar a conversão desses textos em uma estrutura de ocorrências de termos. Para essa tarefa, foi utilizada a biblioteca *Scikit-learn*[8] que possui pacotes como o *feature-extraction.text* para extração de características para representação de textos. Esse pacote possui métodos para realizar esse trabalho, como o *CountVectorizer*, que converte uma coleção de textos em uma matriz de ocorrências de termos, e o *TfidfVectorizer*, que converte uma coleção de textos em uma matriz TF-IDF diretamente. O *TfidfVectorizer* foi escolhido para a extrair a representação dos textos, por conta de sua transformação direta em TF-IDF e de disponibilizar a funcionalidade de definir n-grama no momento de sua utilização.

A Figura 19 ilustra como é feita a conversão de uma coleção de *tweets* em uma matriz TF-IDF. No primeiro bloco, estão sendo utilizados os *tweets* pré-processados no exemplo apresentado anteriormente (veja Figura 18) para criar a coleção de textos chamada ***tweets***. Em seguida é instanciada uma variável ***vectorize*** que será utilizada para criar a matriz de frequências e com a função *fit_transform*, a matriz de frequências de cada *tweet* será criada e atribuída a variável **X**. O segundo bloco está apenas sendo apresentada a matriz para que seja possível a visualização das frequências dos termos em cada *tweet*. É possível ver que cada posição da matriz possui um conjunto de frequências que correspondem a cada *tweet*, e cada um deles possui exatamente 24 valores de frequências de termos. Isso ocorre por conta da coleção de ***tweets*** ter ao todo 24 palavras distintas. É possível ver também que cada palavra que não aparece no *tweet* recebe o valor 0.0, análogo ao que ocorre no exemplo dado na Tabela 2 na apresentação dos conceitos do TF-IDF.

Neste trabalho, utilizou-se cinco classificadores para análise de sentimento, sendo eles *Random Forest* (veja na Seção 3.1), *Logistic Regression* (veja na Seção 3.2), *Naive Bayes Classifier* (veja na Seção 3.3), *K-Nearest Neighbor* (veja na Seção 3.4) e *Support Vector Machine* (veja na Seção 3.5). Para realizar o treinamento dos classificadores, é necessário padronizar os dados para que eles possam ser interpretados, no nosso caso em uma matriz TF-IDF, e suas respectivas classes. O conjunto de dados de treino, no contexto de aprendizado de máquina, normalmente é conhecido como X e suas classes como Y.

[8] Documentação da biblioteca *Scikit-learn* disponível em: <https://scikit-learn.org/stable/>.

Figura 19 – Trecho de código exemplificando a criação de uma matriz TF-IDF e sua estrutura após a criação.

```
In [65]: from sklearn.feature_extraction.text import TfidfVectorizer
         tweets = [
             'simples  deix  chat  muitoo  ilud',
             'comed  rotin  perfeit  consolid  escol  conduçã',
             'apaixon  recíproc',
             'gaj  bacan  ver  aleij  esper  silv  feliz',
             'tou  vid  cabr  beb'
         ]
         vectorizer = TfidfVectorizer()
         X = vectorizer.fit_transform(tweets)

In [67]: print(X.toarray())

[[0.         0.         0.         0.         0.         0.4472136
  0.         0.         0.         0.4472136  0.         0.
  0.         0.         0.4472136  0.4472136  0.         0.
  0.         0.         0.4472136  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.40824829 0.40824829 0.40824829 0.         0.40824829 0.
  0.         0.         0.         0.         0.40824829 0.
  0.40824829 0.         0.         0.         0.         0.        ]
 [0.         0.70710678 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.70710678
  0.         0.         0.         0.         0.         0.        ]
 [0.37796447 0.         0.37796447 0.         0.         0.
  0.         0.         0.         0.         0.         0.37796447
  0.37796447 0.37796447 0.         0.         0.         0.
  0.         0.37796447 0.         0.         0.37796447 0.        ]
 [0.         0.         0.         0.5        0.5        0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.5        0.         0.5       ]]
```

Ainda para a etapa de treinamento, é necessário realizar uma validação cruzada (veja na Seção 3.6.1) com r igual a 5 subconjuntos em todos os classificadores, por conta de 5 ser o valor padrão de subconjuntos para validação cruzada. Para isso, foi utilizado o pacote *sklearn.model_selection* também do *Scikit-learn*, que fornece ferramentas para a validação cruzada como os métodos *cross_val_predict* e *cross_validate*.

Esses dois métodos são muito parecidos, sendo necessário apenas passar alguns parâmetros para utilizá-los como, o classificador instanciado e treinado, o conjunto de dados de treino (X), as classes de treino (Y) e o número r de subconjuntos. A diferença entre os métodos está principalmente no seu valor de retorno, onde *cross_val_predict* retorna uma lista de predições de classes para o conjunto de dados, enquanto o *cross_validate* retorna o tempo de execução para a realização da classificação dos dados e os valores das métricas de avaliação passadas como parâmetro. Entretanto somente com o primeiro método já é possível obter os valores das métricas de avaliação, sendo necessário passar a lista de predições de classes para outros métodos disponíveis no *Scikit-learn*.

Foi feita uma função para criar, treinar e realizar a validação cruzada de uma lista de classificadores passada como parâmetro, além do conjunto de dados de treino (X) e (Y). O parâmetro X são os *tweets* já transformados em uma estrutura de matriz, como apresentada na Figura 19 e Y é uma estrutura de lista, contendo a classe/sentimento de cada *tweet*. Também é passado como parâmetro uma lista de rótulos para identificar cada

classificador. Todo classificador passado para a função, deve ajustar os dados de treino (X) a partir do método *fit*, para só então ser capaz de utilizar a função *cross_val_predict* para treinar o classificador. A função irá retornar a lista de predições de classes de cada classificador passado como parâmetro.

Figura 20 – Trecho de código da função para predição de sentimento dos *tweets*.

```
def _classificarTweets(classificadores, rotulos, X, Y):

    resultados = dict()
    for rotulo, classificador in zip(rotulos, classificadores):
        classificador.fit(X, Y)

        resultados[rotulo] = cross_val_predict(classificador, X, Y, cv = 5)

    return resultados
```

A Figura 21 exemplifica o uso da função ***_classificarTweets*** onde no primeiro bloco é criada a lista de classificadores e atribuída a variável ***classificadores***, já a lista de rótulos para cada classificador é atribuída a variável ***rotulos***. Para cada algoritmo de classificação de sentimento, foram utilizados os parâmetros padrões disponíveis na biblioteca *Scikit-learn*, dentre eles:

- ***KNN***: foi utilizado o parâmetro *n_neighbors* para definir os 5 vizinhos mais próximos de cada dado de treino;
- ***SVM***: foi utilizada a função *kernel* linear (SVC);
- ***Naive Bayes***: foi utilizado o modelo BernulliNB por estarmos trabalhando com dados que possuem apenas dois valores possíveis para classe (POSITIVO e NEGATIVO);
- ***Logistic Regression***: foi utilizado o parâmetro *max_iter* que corresponde ao máximo de interações do algoritmo;
- ***Random Forest***: foi utilizado o parâmetro *n_estimators* para definir o número de árvores e *random_state* que é um número aleatório para controlar a aleatoriedade de dados utilizados para inicializar a árvore.

No segundo bloco de código, é possível recuperar os *tweets* já pré-processados de um *Data Frame*, converter os valores vazios por uma *string* sem nenhum caractere com o método *fillna* e os transformar em uma matriz TF-IDF. O parâmentro *ngram_range* é usado para informar que serão utilizados bigramas para a classificação e então a matriz criada é atribuída a variável ***X***. Em seguida a variável ***Y*** recebe a classe/sentimento de cada *tweet* e a variável ***resultados*** recebe o retorno da função ***_classificarTweets***.

A Figura 22 apresenta o conteúdo de cada variável apresentada, onde no primeiro bloco a função *shape* é utilizada para mostrar as dimensões da matriz TF-IDF da variável

Figura 21 – Trecho de código exemplificando o processo de treinamento e recuperação das predições dos algoritmos de classificação.

```
classificadores = [
    KNeighborsClassifier(n_neighbors=5),
    SVC(kernel="linear"),
    BernoulliNB(),
    LogisticRegression(max_iter=200),
    RandomForestClassifier(n_estimators=100, random_state=100)
]
rotulos = ['KNN', 'SVM', 'BernoulliNB', 'LogisticRegression', 'RandomForestClassifier']
```

```
tweets = df['Tweet Processado'].fillna(' ')

vectorizer = TfidfVectorizer(ngram_range=(1,2))
X = vectorizer.fit_transform(tweets)
Y = df['sentiment']

resultados = _classificarTweets(classificadores, rotulos, X, Y)
```

$\boldsymbol{X}$ que apresenta 49.449 linhas e 117.303 colunas. No segundo bloco o conteúdo da lista da variável $\boldsymbol{Y}$ é apresentado, onde cada linha representa o sentimento de cada *tweet*, sendo o valor 1 representando o sentimento POSITIVO e 0 o sentimento NEGATIVO, tendo ao todo 49.449 linhas. Já no terceiro bloco são apresentadas as predições de sentimento dos *tweets* de cada classificador, onde cada um deles apresenta 49.449 predições. Para as próximas etapas da análise dos classificadores, as predições contidas na variável ***resultados*** serão muito importantes para a aplicação das métricas de avaliação.

Figura 22 – Trecho de código apresentando as dimensões da matriz TD-IDF X, a lista de sentimentos Y e o conjunto de predições *resultados*.

```
X.shape
```

```
(49449, 177303)
```

```
Y
```

```
0        1
1        1
2        1
3        1
4        1
        ..
49444    0
49445    0
49446    0
49447    0
49448    0
Name: sentiment, Length: 49449, dtype: int64
```

```
for rotulo, classificador in zip(rotulos, classificadores):
    print(rotulo, resultados[rotulo], "Length:", len(resultados[rotulo]))
```

```
KNN [1 0 1 ... 0 0 0] Length: 49449
SVM [1 1 1 ... 0 1 0] Length: 49449
BernoulliNB [1 1 1 ... 0 0 0] Length: 49449
LogisticRegression [1 1 1 ... 0 1 0] Length: 49449
RandomForestClassifier [1 1 1 ... 0 0 0] Length: 49449
```

Com as predições já feitas, é possível extrair diversas informações sobre cada classificador utilizado e construir matrizes de confusão para realizar uma análise mais detalhada

das validações. A Figura 23a apresenta a matriz de confusão do classificador *KNN*, a Figura 23b, do *SVM*, a Figura 23c, do *Naive Bayes*, a Figura 23d, da *Logistic Regression* e a Figura 23e do *Random Forest*. Cada figura é um plano bidimensional onde o eixo x representa os valores de classes/sentimentos previstos pelo classificador e o eixo y representa os valores de classes/sentimentos reais, no caso estudado no trabalho as classes são 1 e 0.

Como já abordado na Seção 3.6.2, a matriz de confusão é utilizada para avaliar um classificador, sendo possível identificar qual classe o algoritmo possui mais facilidade para classificar. Na matriz de confusão para problemas binários, temos quatro tipos de valores diferentes de predição, Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN).

As predições corretas de um classificador se encontram na diagonal principal das matrizes de confusão apresentadas na Figura 23, onde o primeiro valor representa os VN e o segundo valor os VP, já as predições incorretas do classificador se encontram na diagonal secundária das matrizs, onde o primeiro valor representa os FP e o segundo valor os FN.

Por exemplo, na matriz de confusão da *Logistic Regression* (Figura 23d), a diagonal principal possui ao todo 15.905 *tweets* classificados corretamente com a classe/sentimento 0 ou NEGATIVO (VN) e 19.063 classificados corretamente com 1 ou POSITIVO (VP), enquanto a diagonal secundária possui 8.906 *tweets* classificados incorretamente com a classe/sentimento 1 (FP) e 5.575 classificados incorretamente com classe/sentimento 0 (FN).

A partir dos dados obtidos das matrizes de confusão e utilizando as métricas de avaliação apresentadas na Seção 3.6.3, já será possível avaliar a qualidade dos classificadores estudados neste trabalho. A aplicação das métricas de avaliação serão demonstradas nos experimentos realizados na Seção 5.1.4.

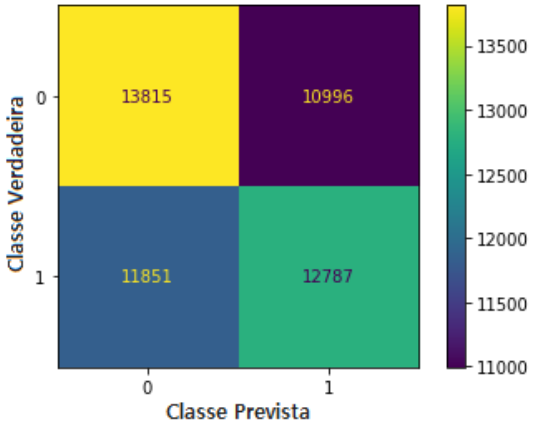
5.1.4 Resultados

No trabalho de conclusão de curso descrito aqui optou-se por realizar quatro tipos de experimentos para avaliar o desempenho de cada classificador de sentimento estudado. Em todos os experimentos os passos apresentados na Seção 5.1.3 como a definição da função de classificação dos *tweets* e a transformação dos dados em uma matriz TF-IDF (veja na Figura 21), se mantém. Cada experimento se diferencia pela variação da utilização dos n-grama e pelas etapas de pré-processamento que foram utilizadas nos dados de treino.

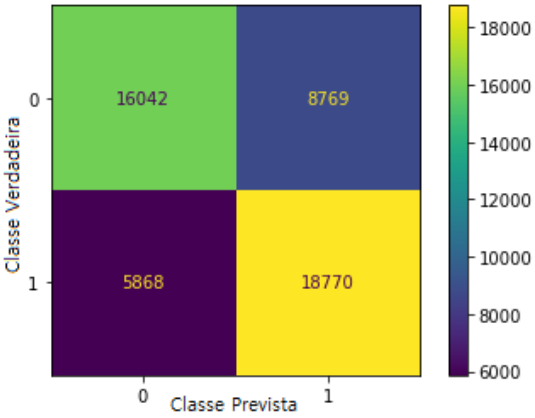
Foi feita a escolha de variar as técnicas de pré-processamento com o objetivo de constatar a implicação do processamento de texto sobre os resultados das métricas de avaliação, uma vez que essas técnicas influenciam diretamente na base de dados, diminuindo o número de termos existentes e aumentando a frequência de termos mais relevantes dos

Figura 23 – Matrizes de confusão dos cinco classificadores estudados.

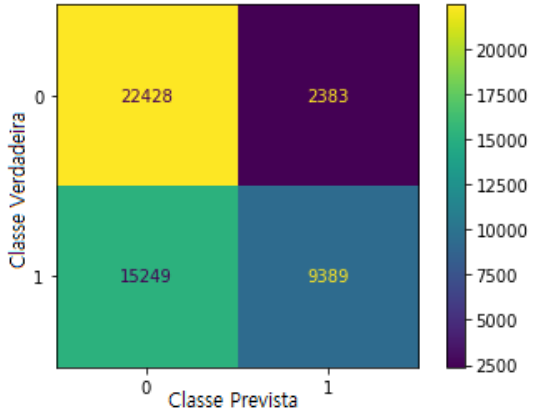
(a) *KNN*



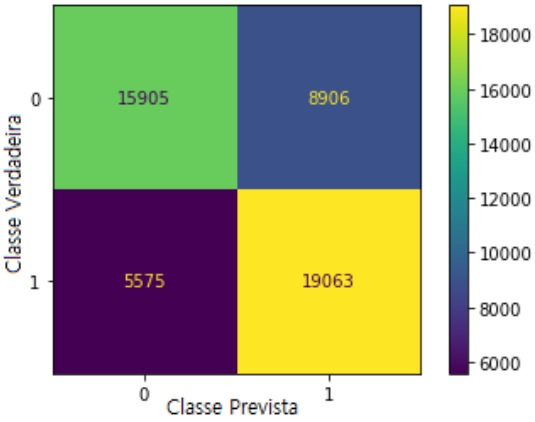
(b) *SVM*



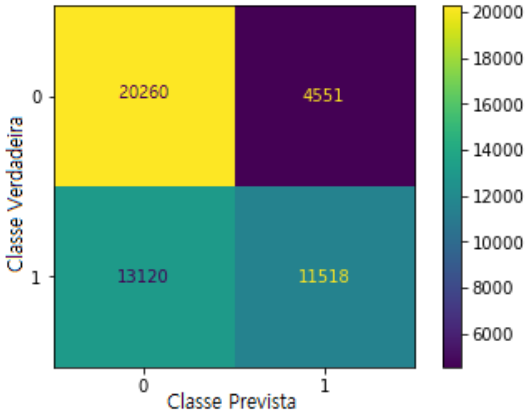
(c) *Naive Bayes*



(d) *Logistic Regression*



(e) *Random Forest*



textos. A escolha de realizar experimentos variando a técnica de n-grama possui o mesmo objetivo dos experimentos variando as técnicas de pré-processamento, constatar o seu efeito nos resultados das métricas de avaliação, visto que essa técnica irá disponibilizar uma gama maior de *tokens* para treinar os algoritmos de classificação.

Para facilitar a apresentação dos resultados obtidos, os experimentos são divididos em dois tipos, sendo eles, os experimentos em que foram utilizadas a técnica de n-grama e os experimentos que não utilizaram. Em cada um deles, foram feitos dois testes diferentes alternando a utilização das técnicas de pré-processamento, sendo eles, um teste utilizando os *tweets* sem nenhum tipo de pré-processamento, ou seja, usando os dados brutos e um teste com os *tweets* pré-processados.

5.1.4.1 Experimentos sem n-grama

Os primeiros experimentos foram feitos sem a utilização de n-grama, com isso a etapa de treinamento de cada classificador se manteve muito parecida com o que foi aplicado na Figura 20, como a definição e parametrização dos classificadores e os rótulos. A Figura 24 apresenta a aplicação da etapa de treinamento dos classificadores dos dois primeiros experimentos feitos no trabalho descrito aqui.

Figura 24 – Trecho de código utilizado para treinamento dos algoritmos de classificação sem utilização de n-grama.

```
tw_bruto = df['tweet_text'].fillna(' ')
tw_procs = df['Tweet Processado'].fillna(' ')

vectorizer = TfidfVectorizer()

X_bruto = vectorizer.fit_transform(tw_bruto)
X_procs = vectorizer.fit_transform(tw_procs)

Y = df['sentiment']

y_predicoes_bruto = _classificarTweets(classificadores, rotulos, X_bruto, Y)
y_predicoes_procs = _classificarTweets(classificadores, rotulos, X_procs, Y)
```

Os *tweets* sem nenhum tipo de pré-processamento foram obtidos a partir do *Data Frame* ***df*** e atribuídos a variável ***tw_bruto***, já os *tweets* pré-processados foram atribuídos a variável ***tw_procs*** e serão utilizados para treinar cada classificador. Esses dados de treinamento foram transformados em matrizes TF-IDF e armazenados nas váriáveis ***X_bruto*** e ***X_procs***, enquanto as classes/sentimentos foram armazenadas na variável ***Y***.

Em seguida foi feita a chamada da função ***_classificarTweets*** para classificar os dados de treinamento passados como parâmetro, assim como a lista de classificadores, a lista de rótulos dos modelos e as classes/sentimentos dos *tweets*. As predições dos ex-

perimentos com os dados brutos são armazenados na variável ***y_predicoes_bruto*** e as predições com os dados pré-processados na variável ***y_predicoes_procs***.

Com as predições prontas, é possível obter as matrizes de confusão de cada classificador nos dois primeiros experimentos. Para melhorar a visualização dos valores das matrizes de confusão, as Tabelas 7 e 8 foram criadas, onde apresentam os resultados das predições do experimento com os dados brutos e as predições do experimento com os dados pré-processados, respectivamente.

Tabela 7 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 1.

	KNN	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Verdadeiro Negativo	6.583	18.349	21.579	18.039	18.111
Falso Negativo	2.089	5.354	10.771	5.050	7.690
Verdadeiro Positivo	22.549	19.284	13.867	19.588	16.948
Falso Positivo	18.228	6.462	3.232	6.772	6.700
Total de Acertos	29.132	**37.633**	35.446	37.627	35.059
Total de Erros	20.317	11.816	14.003	11.822	14.390
Total de Predições	49.449	49.449	49.449	49.449	49.449

Tabela 8 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 2.

	KNN	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Verdadeiro Negativo	15.321	16.088	20.414	16.126	17.314
Falso Negativo	11.324	6.218	11.377	5.899	9.439
Verdadeiro Positivo	13.314	18.420	13.261	18.739	15.199
Falso Positivo	9.490	8.723	4.397	8.685	7.497
Total de Acertos	28.635	34.508	33.675	**34.865**	32.513
Total de Erros	20.814	14.941	15.774	14.584	16.936
Total de Predições	49.449	49.449	49.449	49.449	49.449

Observando a Tabela 7, fica mais claro identificar quais classificadores tiveram melhores resultados levando em consideração seu número total de acertos para o primeiro experimento. Com isso é possível ordenar os classificadores para esse experimento, com *KNN* em quinto, *Random Forest* em quarto, *Naive Bayes* em terceiro, *Logistic Regression* em segundo e o *SVM* em primeiro, sendo esses dois últimos com total de acertos muito próximos. Já na Tabela 8, para o experimento com os dados pré-processados sem a utilização de n-grama, não houveram tantas mudanças na ordem dos classificadores que obtiveram melhores resultados, desta vez o algoritmo *Logistic Regression* se saiu melhor que o algoritmo *SVM*, enquanto os outros algoritmos mantiveram sua ordenação anterior. A principal diferença notável neste experimento, fica por conta do aumento do número total de erros de todos os classificadores, dando destaque para o *SVM* que neste experimento classificou incorretamente 3.125 *tweets* a mais que no experimento anterior.

Para avaliar a qualidade dos classificadores foram utilizadas cinco métricas de avaliação, sendo elas ***acurácia***, ***precisão***, ***revocação***, ***F1-score*** e ***Curva ROC***, todas apresentadas na Seção 3.6.3. Com as predições obtidas na fase de treinamento, é utilizado o pacote de métricas *sklearn.metrics* do *Scikit-learn* para avaliar os modelos. A aplicação das métricas é bastante simples, bastando apenas passar a lista de classes/sentimentos reais que foram utilizadas para treino (variável ***Y***) e a lista de predições de cada algoritmo. A Figura 25 apresenta os resultados obtidos com a utilização das funções das métricas de avaliação.

Figura 25 – Trecho de código para obtenção dos resultados das métricas de avaliação dos experimentos 1 e 2.

```
accuracy_bruto = dict()
precision_bruto = dict()
recall_bruto = dict()
f1_bruto = dict()
roc_bruto = dict()
for rotulo, clf in zip(rotulos, classificadores):
    accuracy_bruto[rotulo] = accuracy_score(Y , y_predicoes_bruto[rotulo])
    precision_bruto[rotulo] = precision_score(Y , y_predicoes_bruto[rotulo])
    recall_bruto[rotulo] = recall_score(Y , y_predicoes_bruto[rotulo])
    f1_bruto[rotulo] = f1_score(Y , y_predicoes_bruto[rotulo])
    roc_bruto[rotulo] = roc_auc_score(Y , y_predicoes_bruto[rotulo])
```

```
accuracy_procs = dict()
precision_procs = dict()
recall_procs = dict()
f1_procs = dict()
roc_procs = dict()
for rotulo, clf in zip(rotulos, classificadores):
    accuracy_procs[rotulo] = accuracy_score(Y , y_predicoes_procs[rotulo])
    precision_procs[rotulo] = precision_score(Y , y_predicoes_procs[rotulo])
    recall_procs[rotulo] = recall_score(Y , y_predicoes_procs[rotulo])
    f1_procs[rotulo] = f1_score(Y , y_predicoes_procs[rotulo])
    roc_procs[rotulo] = roc_auc_score(Y , y_predicoes_procs[rotulo])
```

No primeiro e no segundo bloco de código, são obtidos os resultados de cada métrica para as predições feitas nos dois primeiros experimentos, sendo as funções ***accuracy_score*** para acurácia, ***precision_score*** para precisão, ***recall_score*** para revocação e ***f1_score*** para F1-score. Para a Curva ROC é utilizada a função ***roc_auc_score*** para obter a área sob a curva, indicando que quanto maior a área, melhor a qualidade do classificador. Nas Tabelas 9 e 10 são apresentados os resultados do primeiro e do segundo experimento respectivamente, das métricas de avaliação para cada algoritmo em porcentagem, com excessão a Curva ROC que será apresentada graficamente na Figura 27.

Tabela 9 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 1.

Métrica (%)	***KNN***	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Acurácia	58,9	**76,1**	71,7	**76,1**	70,8
Precisão	55,2	74,9	**81,0**	74,3	71,6
Revocação	**91,5**	78,2	56,2	79,5	68,7
F1-score	68,9	76,5	66,4	**76,8**	70,1

Tabela 10 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 2.

Métrica (%)	***KNN***	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Acurácia	57,9	69,7	68,1	**70,5**	65,7
Precisão	58,3	67,8	**75,0**	68,3	66,9
Revocação	54,0	74,7	53,8	**76,0**	61,6
F1-score	56,1	71,1	62,7	**71,9**	64,2

Na Tabela 9 é possível observar que para cada métrica de avaliação, um classificador que se saiu melhor que os outros trabalhando com os dados sem pré-processamento. Para a acurácia os algoritmos *SVM* e *Logistic Regression* ficaram empatados com $76,1\%$ de acerto nas classificações de sentimentos, como no total de acertos apresentado na Tabela 7. Para a precisão o algoritmo *Naive Bayes* se saiu melhor com 81%, mostrando a capacidade de acerto do classificador quando ele faz uma predição para a classe/sentimento POSITIVO, já para a revocação, o algoritmo *KNN* se destacou com $91,5\%$, indicando que o algoritmo possui uma boa capacidade de classificar para a classe/sentimento POSITIVO quando o sentimento é realmente POSITIVO. O *F1-score* mostra novamente o algoritmo *Logistic Regression* como o que se desempenhou melhor com $76,8\%$, indicando que para esse experimento, o classificador possui uma melhor capacidade de acertar suas predições e classificar como sentimento POSITIVO quando de fato é POSITIVO.

Na Tabela 10 percebe-se uma queda nos valores de todas as métricas de avaliação para todos os classificadores após os dados de treinamento passarem pelas técnicas de pré-processamento. A principal diferença nos resultados, pode-se destacar a perda de $37,5\%$ de revocação do *KNN*, porém um aumento de $3,1\%$ em sua precisão. Após esses dois primeiros experimentos, podemos ranquear os classificadores com melhor desempenho levando em consideração todas as métricas aplicadas, tendo em quinto o algoritmo *KNN*, em quarto o *Naive Bayes*, em terceiro o *Random Forest*, em segundo o *SVM* e o *Logistic Regression* como o classificador que obteve melhores resultados.

5.1.4.2 Experimentos com n-grama

Os dois últimos experimentos utilizam a técnica de n-grama no momento de treinar os classificadores, como já abordado anteriormente nesta seção. Para esses experimentos, todas as etapas de treinamento e obtenção das predições (veja na Figura 24) e depois a aplicação das métricas de avaliação (veja na Figura 25), serão repetidas para esses novos experimentos. A única diferença será na utilização do parâmetro *ngram_range* na função *TfidfVectorizer* informando que serão utilizados bigramas nos treinamentos dos classificadores, como é apresentado na Figura 26.

Realizada a execução do treinamento dos classificadores utilizando bigramas e obtendo suas respectivas predições de classes/sentimentos, novamente é possível obter

Figura 26 – Trecho de código apresentando a utilização de bigramas na função *TfidfVectorizer*.

```
vectorizer = TfidfVectorizer(ngram_range=(1,2))
```

as matrizes de confusão dos algoritmos. As Tabelas 11 e 12 apresentam os resultados das predições utilizando bigrama do experimento com os dados brutos e com os dados pré-processados respectivamente.

Tabela 11 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 3.

	KNN	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Verdadeiro Negativo	6.862	18.381	22.940	17.831	18.701
Falso Negativo	2.016	4.917	13.791	4.805	9.270
Verdadeiro Positivo	22.622	19.721	10.847	19.833	15.368
Falso Positivo	17.949	6.430	1.871	6.980	6.110
Total de Acertos	29.484	**38.102**	33.787	37.664	34.069
Total de Erros	19.965	11.347	15.662	11.785	15.380
Total de Predições	49.449	49.449	49.449	49.449	49.449

Tabela 12 – Tabela gerada a partir das matrizes de confusão de cada algoritmo de classificação para o experimento 4.

	KNN	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Verdadeiro Negativo	13.815	16.042	22.428	15.905	20.260
Falso Negativo	11.851	5.868	15.249	5.575	13.120
Verdadeiro Positivo	12.787	18.770	9.389	19.063	11.518
Falso Positivo	10.996	8.769	2.383	8.906	4.551
Total de Acertos	26.602	34.812	31.817	**34.968**	31.778
Total de Erros	22.847	14.637	17.632	14.481	17.671
Total de Predições	49.449	49.449	49.449	49.449	49.449

Semelhante ao que ocorreu nos primeiros experimentos, o número total de acertos de todos os classificadores tiveram uma queda no número de acertos após os *tweets* passarem pelas etapas de pré-processamento. Além disso, não houveram tantas mudanças na ordenação dos classificadores que obtiveram um maior número de acertos, onde para o experimento em que não houve pré-processamento dos dados, tivemos em ordem descrecente os algoritmos *KNN*, *Naive Bayes*, *Random Forest*, *Logistic Regression* e *SVM*. Já no experimento em que houve pré-processamento dos dados, a ordenação dos classificadores ficou com *KNN*, *Random Forest*, *Naive Bayes*, *SVM* e *Logistic Regression* como os classificadores com mais acertos.

Seguindo os mesmos passos dos experimentos anteriores, os resultados das métricas de avaliação dos classificadores são obtidos e então apresentados nas Tabelas 13 e 14. Fazendo uma análise dos resultados obtidos, nota-se que a utilização dos bigramas não

teve um efeito tão significativo na performance dos classificadores, porém ainda assim percebe-se um pequeno aumento de acertividade em todas as métricas para todos os algoritmos. A excessão foi com o algoritmo *Naive Bayes*, onde houve uma queda notável na sua revocação, indo de $56,2\%$ e $53,8\%$ no primeiro e segundo experimentos sem n-grama (veja as tabelas 9 e 10) para 44% e $38,1\%$ nos experimentos com n-grama, e por conta do *f1-score* utilizar a revocação, essa métrica também possuiu uma queda em seus valores.

Tabela 13 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 3.

Métrica (%)	***KNN***	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Acurácia	59,6	**77,0**	68,3	76,1	68,8
Precisão	55,7	75,4	**85,2**	73,9	71,5
Revocação	**91,8**	80,0	44,0	80,4	62,3
F1-score	69,3	**77,6**	58,0	77,0	66,6

Tabela 14 – Resultados das métricas de avaliação de cada algoritmo de classificação para o experimento 4.

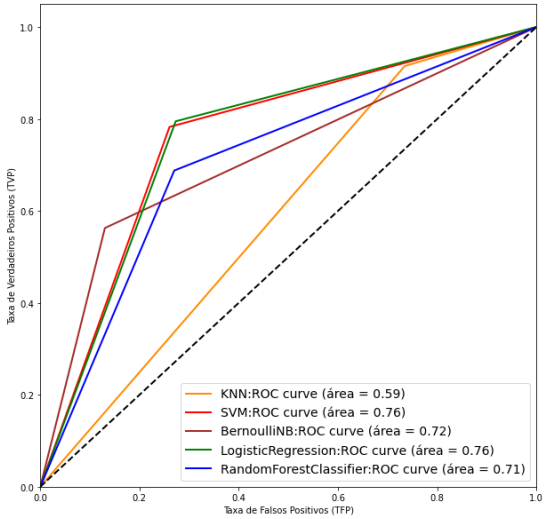
Métrica (%)	***KNN***	***SVM***	***Naive Bayes***	***Logistic Regression***	***Random Forest***
Acurácia	53,7	70,3	64,3	**70,7**	64,2
Precisão	53,7	68,1	**79,7**	68,1	71,6
Revocação	51,8	76,1	38,1	**77,3**	46,7
F1-score	52,8	71,9	51,5	**72,4**	56,5

Considerando os valores obtidos com a utilização das métricas de avaliação, o ranquemento dos classificadores que melhor desempenharam a tarefa de classificação para o experimento sem pré-processamento e utilização de bigramas foram em ordem decrescente os algoritmos *KNN*, *Naive Bayes*, *Random Forest*, *Logistic Regression* e *SVM*, enquanto para o experimento com pré-processamento dos dados e utilização de bigramas o algoritmo *Logistic Regression* se saiu melhor do que o *SVM*. É interessante notar que para os quatro experimentos realizados, os algoritmos *Logistic Regression* e *SVM* se sobressaíram sobre os outros três algoritmos, isso fica mais nítido quando é gerada a Curva ROC de cada classificador com suas predições. Nas Figuras 27a, 27b, 27c e 27d, são apresentadas as Curvas ROC do primeiro, segundo, terceiro e quarto experimento respectivamente, onde no eixo X de cada figura representa a Taxa de Falsos Positivos (TFP) e o eixo Y a Taxa de Verdadeiros Positivos (TVP), como foi apresentado na explicação dos conceitos da Curva ROC (veja a Seção 3.6.3).

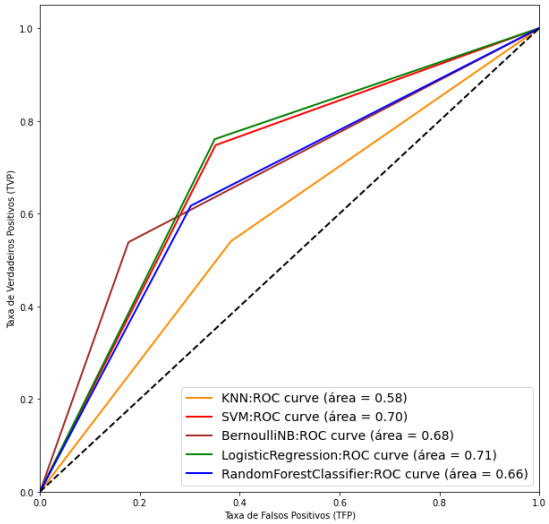
Nas quatro figuras é possível notar como os classificadores *Logistic Regression* e *SVM* se destacaram perante os outros algoritmos em todos os experimentos, obtendo uma área sob a curva entre 0.76 e 0.77 nos experimentos onde não foram utilizadas as técnicas de pré-processamento (figuras 27a e 27c) e 0.70 e 0.71 nos experimentos em

Figura 27 – Curvas ROC dos quatro experimentos realizados.

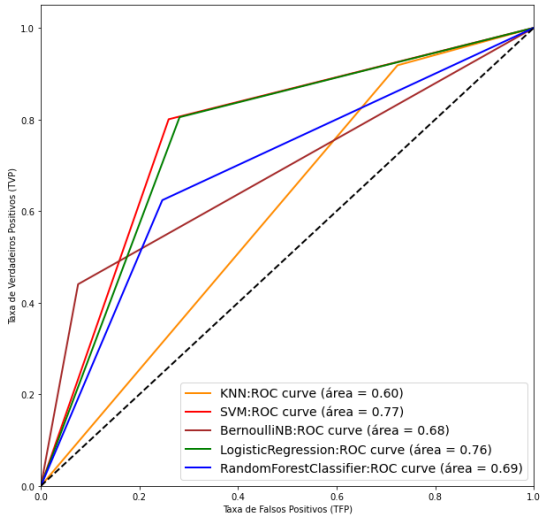
(a) Experimento 1



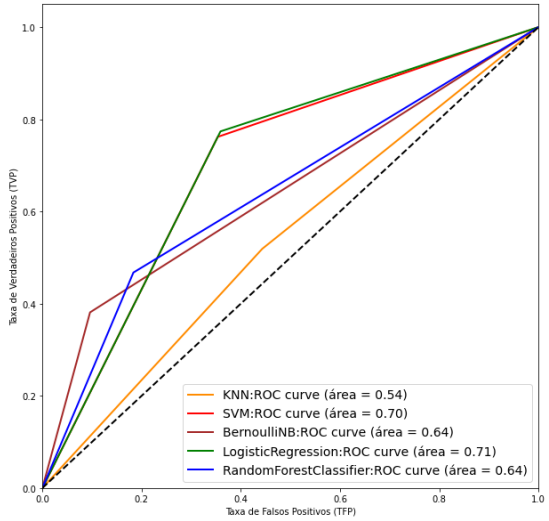
(b) Experimento 2



(c) Experimento 3



(d) Experimento 4



que as técnicas de pré-processamento de texto foram utilizadas (figuras 27b e 27d). É interessante notar também que todos os classificadores sofreram uma redução em sua Taxa de Verdadeiros Positivos (TVP), consequentemente uma diminuição da área sob a curva, após os *tweets* passarem pelas etapas de pré-processamento ou pela técnica de *n*-grama. Dadas as informações das tabelas das matrizes de confusão e pelos resultados das métricas de avaliação, tanto o algoritmo *SVM* quanto o *Logistic Regression* obtiveram os melhores resultados para classificação de sentimento da rede social do *Twitter*, portanto, poderão ser utilizados no projeto para realizar a classificação de sentimento dos usuários.

6 Conclusão e Trabalhos Futuros

O trabalho de conclusão de curso descrito nessa monografia teve como objetivo explorar os principais conceitos da análise de sentimento no contexto das redes sociais. A facilidade de obtenção dos dados do *Twitter*, fez com que essa rede social tenha sido escolhida como o objeto de estudo neste trabalho.

Outro objetivo foi estudar e separar os diferentes algoritmos de classificação de sentimento encontrados na literatura e definir quais seriam utilizados no trabalho de conclusão de curso, além de explorar as implementações desses algoritmos na linguagem de programação *Python*.

Durante o desenvolvimento do trabalho foi necessário realizar diversos experimentos com o conjunto de dados, como as etapas de pré-processamento dos dados, que tipicamente pretende otimizar a classificação de textos. Entretanto nem sempre a utilização dessas técnicas afetam de forma positiva nos resultados se analisada separadamente, como foi visto nos resultados deste trabalho. Além disso, foi preciso estudar técnicas de transformação de textos como TF-IDF e n-grama para que os algoritmos de classificação consigam processar as informações dos *tweets*.

Outros conceitos e técnicas importantes estudadas neste trabalho foram as métricas de avaliação dos algoritmos. Com essas métricas foi possível realizar uma análise mais detalhada do desempenho de cada classificador estudado e chegar a definição dos algoritmos ideiais para classificação de sentimentos do projeto maior citado anteriormente.

Com relação aos resultados obtidos e a comparação dos algoritmos de classificação, tivemos algumas situações interessantes que ocorreram. De modo geral os classificadores de sentimento *Logistic Regression* e *SVM* obtiveram resultados muito semelhantes em todos os experimentos que foram realizados nesse trabalho de conclusão de curso. O algoritmo *SVM* teve um melhor desempenho nos experimentos onde não foi utilizado as técnicas de pré-processamento de texto, enquanto o algoritmo *Logistic Regression* se saiu melhor nos experimentos com pré-processamento de texto. As etapas de treinamento e testes dos algoritmos são feitas rapidamente para a base de dados com 50.000 *tweets*, levando em média 10 segundos por algoritmo para serem realizados nos experimentos sem utilização de bigramas e 20 segundos nos experimentos com bigramas. A excessão neste caso fica para o algoritmo *SVM* que levou cerca de 13 minutos para os experimentos sem bigramas e 30 minutos com bigramas.

Foi possível observar também o comportamento dos classificadores ao aplicar os bigramas na fase de treinamento. O desempenho de todos os algoritmos foi menor se comparados isoladamente com os experimentos sem bigramas, em especial o algoritmo

KNN. Já os algoritmos *Naive Bayes* e *Random Forest* não tiveram o desempenho tão ruim se comparado ao *KNN*, porém não conseguiram superar o *SVM* e *Logistic Regression* muito por conta de sua baixa revocação. O algoritmo *Naive Bayes* ainda assim na maioria dos experimentos, alcançava classificações com mais de 80% de precisão em suas predições nos sentimentos dos *tweets*.

Diferentemente do que é encontrado na literatura, a aplicação de técnicas para limpeza dos dados e pré-processamento de texto não teve o efeito esperado nos resultados apresentados, como por exemplo um aumento nas previsões corretas dos algoritmos de classificação. O mesmo ocorreu com a utilização da técnica de n-grama, onde esperava-se que também ocorreria um pequeno aumento nas previsões corretas dos algoritmos. Isso ocorreu por conta das etapas de pré-processamento dos *tweets* não realizarem uma limpeza mais profunda de caracteres, como a remoção de *emoticons* e *emotes*. Outro fator importante que impactou nas classificações, seria a não utilização de um dicionário informal contendo palavras abreviadas e gírias que auxiliariam em uma melhor limpeza dos dados.

Nesse trabalho de conclusão de curso foi utilizado uma base de dados com publicações da rede social *Twitter* do ano de 2018. Uma melhor predição dos algoritmos de classificação pode ser obtida a partir da análise de outras representações de dados encontradas nas postagens, como a análise dos *emoticons*, *emotes* e também análise de *hashtags*.

Além disso, seria interessante refinar o pré-processamento para que não apareçam palavras abreviadas ou gírias comumente utilizadas nas redes sociais. Assim os classificadores de sentimentos receberiam um conjunto de dados de treino só com termos que realmente são relevantes no texto e consequentemente, melhores resultados serão alcançados.

Outra melhoria que pode ser feita nesse trabalho de conclusão de curso, seria o estudo de diferentes tipos de bases de dados, como bases artificais e bases reais. Além disso estudar bases de dados que possuem mais de dois tipos de classes/sentimentos, uma vez que neste trabalho foram abordadas apenas as classes POSITIVO e NEGATIVO, com o objetivo de verificar o comportamento dos algoritmos de classificação de sentimento e comparar os resultados das predições variando o uso das técnicas de pré-processamento de texto e n-grama.

Por fim, seria interessante também explorar outros tipos de algoritmos de classificação de sentimentos encontrados na literatura, como os algoritmos de *deep learning*, por exemplo as Redes Neurais Artificiais e então compará-los com os algoritmos inicialmente estudados.

Referências

AGGARWAL, C. C.; ZHAI, C. A survey of text classification algorithms. In: *Mining text data*. [S.l.]: Springer, 2012. p. 163–222. Citado na página 17.

ALLAHYARI, M.; POURIYEH, S.; ASSEFI, M.; SAFAEI, S.; TRIPPE, E. D.; GUTIERREZ, J. B.; KOCHUT, K. *A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques*. 2017. Citado na página 9.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval: The Concepts and Technology behind Search*. 2nd. ed. USA: Addison-Wesley Publishing Company, 2011. ISBN 9780321416919. Citado na página 19.

BARION, D. L. E. C. N. "mineração de textos. *Revista de Ciências Exatas e Tecnologia*, III, n. 3, p. 123 – 140, 2008. Disponível em: <https://revista.pgsskroton.com/index.php/rcext/article/view/2372/2276>. Citado na página 16.

BECKER, K.; MOREIRA, V. P.; dos Santos, A. G. Multilingual emotion classification using supervised learning: Comparative experiments. *Information Processing Management*, v. 53, n. 3, p. 684–704, 2017. ISSN 0306-4573. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306457316307099>. Citado na página 34.

BECKER, K.; TUMITAN, D. Introdução à mineração de opiniões: Conceitos, aplicações e desafios. *Simpósio brasileiro de banco de dados*, v. 75, 2013. Citado 2 vezes nas páginas 7 e 11.

BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: <https://doi.org/10.1023/A:1010933404324>. Citado na página 21.

BROWNLEE, J. *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*. 2018. <https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/>. Citado na página 32.

CARDIE, C. Sentiment analysis and opinion mining bing liu (university of illinois at chicago) morgan & claypool (synthesis lectures on human language technologies, edited by graeme hirst, 5(1)), 2012, 167 pp; paperbound, isbn 978-1-60845-884-4. *Computational Linguistics*, v. 40, n. 2, p. 511–513, 2014. Disponível em: <https://doi.org/10.1162/COLI_r_00186>. Citado 2 vezes nas páginas 12 e 14.

CARVALHO, A.; FACELI, K.; LORENA, A.; GAMA, J. Inteligência artificial–uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, p. 45, 2011. Citado 5 vezes nas páginas 26, 27, 29, 30 e 31.

CHOLLET. *Deep Learning with Python*. 2010. <https://colab.research.google.com/github/iagml/iagml.github.io/blob/master/assets/uploads/chollet_6_1.ipynb#scrollTo=X3mEy_mOiYu8>. Citado na página 18.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. USA: Wiley-Interscience, 2000. ISBN 0471056693. Citado 2 vezes nas páginas 25 e 26.

FIGUEIRA, C. V. *Modelos de Regressão Logística*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006. Citado na página 22.

HEMMATIAN, F.; SOHRABI, M. K. A survey on classification techniques for opinion mining and sentiment analysis. *Artificial Intelligence Review*, v. 52, n. 3, p. 1495–1545, Oct 2019. ISSN 1573-7462. Disponível em: <https://doi.org/10.1007/s10462-017-9599-6>. Citado 2 vezes nas páginas 7 e 36.

HU, X.; LIU, H. Text analytics in social media. In: *Mining text data*. [S.l.]: Springer, 2012. p. 385–414. Citado na página 15.

ICHI.PRO. *Máquina de vetores de suporte (SVM) explicada*. 2020. <https://ichi.pro/pt/maquina-de-vetores-de-suporte-svm-explicada-97743104690915>. Citado na página 29.

INFERIR. *ALGORITMO KNN PARA CLASSIFICAÇÃO*. 2019. <https://inferir.com.br/artigos/algoritimo-knn-para-classificacao/>. Citado na página 27.

LEON-GARCIA, A.; LEON-GARCIA, A. *Probability, statistics, and random processes for electrical engineering /*. 3rd ed.. ed. Upper Saddle River, NJ :: Pearson/Prentice Hall,, c2008. Disponível em: <http://www.loc.gov/catdir/toc/ecip084/2007046492.html>. Citado na página 24.

LIMA, A. C. E. S.; CASTRO, L. N. de. Tecla: A temperament and psychological type prediction framework from twitter data. *PLOS ONE*, Public Library of Science, v. 14, n. 3, p. 1–18, 03 2019. Disponível em: <https://doi.org/10.1371/journal.pone.0212844>. Citado na página 36.

LIU, B.; ZHANG, L. A survey of opinion mining and sentiment analysis. *Springer US*, p. 415–463, 2012. Citado na página 10.

MESQUITA, P. S. B. *Um Modelo de Regressão Logística para Avaliação dos Programas de Pós-Graduação no Brasil*. Dissertação (Mestrado) — Universidade Estadual do Norte Fluminense, Campo dos Goytacazes, 2014. Citado na página 22.

MITCHELL, T. M. *Machine Learning*. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077. Citado na página 24.

NADEEM. *Logistic Regression Using Python*. 2021. <https://medium.com/analytics-vidhya/logistic-regression-using-python-a5044843a504>. Citado 2 vezes nas páginas 23 e 28.

PALMER, D. D. Text preprocessing. In: INDURKHYA, N.; DAMERAU, F. J. (Ed.). *Handbook of Natural Language Processing, Second Edition*. Chapman and Hall/CRC, 2010. p. 9–30. Disponível em: <http://www.crcnetbase.com/doi/abs/10.1201/9781420085938-c2>. Citado na página 15.

PANG, B.; LEE, L. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, Now Publishers Inc., Hanover, MA, USA, v. 2, n. 1–2, p. 1–135, jan. 2008. ISSN 1554-0669. Disponível em: <https://doi.org/10.1561/1500000011>. Citado 2 vezes nas páginas 10 e 34.

PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs up? sentiment classification using machine learning techniques. In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, 2002. p. 79–86. Disponível em: <https://www.aclweb.org/anthology/W02-1011>. Citado na página 14.

QUERCIA, D.; KOSINSKI, M.; STILLWELL, D.; CROWCROFT, J. Our twitter profiles, our selves: Predicting personality with twitter. In: IEEE. *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*. [S.l.], 2011. p. 180–185. Citado na página 35.

SANTANA, R. *Validação Cruzada: Aprenda de forma simples como usar essa técnica*. 2020. <https://minerandodados.com.br/validacao-cruzada-aprenda-de-forma-simples-como-usar-essa-tecnica/>. Citado na página 30.

SILVA, M. J.; CARVALHO, P.; SARMENTO, L. Building a sentiment lexicon for social judgement mining. In: CASELI, H.; VILLAVICENCIO, A.; TEIXEIRA, A.; PERDIGÃO, F. (Ed.). *Computational Processing of the Portuguese Language*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 218–228. ISBN 978-3-642-28885-2. Citado na página 13.

SKOWRON, M.; TKALcIc, M.; FERWERDA, B.; SCHEDL, M. Fusing social media cues: Personality prediction from twitter and instagram. In: *Proceedings of the 25th International Conference Companion on World Wide Web*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016. (WWW '16 Companion), p. 107–108. ISBN 9781450341448. Disponível em: <https://doi.org/10.1145/2872518.2889368>. Citado na página 35.

SOUZA, J. N. V.; BECKER, K. Characterization of anxiety, depression, and their comorbidity from texts of social networks. Instituto de Informatica – Universidade Federal do Rio Grande do Sul (UFRGS), 2019. Citado na página 7.

SOUZA, M.; VIEIRA, R. Construction of a portuguese opinion lexicon from multiple resources. *Anais do Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana, 2011, Brasil.*, 2011. Citado na página 13.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367. Citado na página 14.

TAUSCZIK, Y. R.; PENNEBAKER, J. W. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*, v. 29, n. 1, p. 24–54, 2010. Disponível em: <https://doi.org/10.1177/0261927X09351676>. Citado na página 34.

TIBCO. *What is a Random Forest?* 2021. <https://www.tibco.com/reference-center/what-is-a-random-forest>. Citado na página 22.

TSYTSARAU, M.; PALPANAS, T. Survey on mining subjective data on the web. *KDD*, v. 24, p. 478–514, 01 2011. Citado 4 vezes nas páginas 10, 11, 12 e 13.

VAPNIK, V. N. *The Nature of Statistical Learning Theory*. Berlin, Heidelberg: Springer-Verlag, 1995. ISBN 0387945598. Citado na página 28.

WANG, H.; CAN, D.; KAZEMZADEH, A.; BAR, F.; NARAYANAN, S. A system for real-time Twitter sentiment analysis of 2012 U.S. presidential election cycle. In: *Proceedings of the ACL 2012 System Demonstrations*. Jeju Island, Korea: Association for Computational Linguistics, 2012. p. 115–120. Disponível em: <https://www.aclweb.org/anthology/P12-3020>. Citado na página 36.

ZHAO, J. The impact of cross-entropy on language modeling. *Mississippi State University*, 1999. Citado na página 18.