UNIVERSIDADE FEDERAL DO PARANÁ

JOSUÉ LEAL EVANGELISTA

APRENDIZADO DE MÁQUINA APLICADO A ANÁLISE DE SENTIMENTO EM AVALIAÇÕES DE USUÁRIOS DE APLICATIVOS DO GOOGLE PLAY

CURITIBA

2022

JOSUÉ LEAL EVANGELISTA

APRENDIZADO DE MÁQUINA APLICADO A ANÁLISE DE SENTIMENTO EM AVALIAÇÕES DE USUÁRIOS DE APLICATIVOS DO GOOGLE PLAY

Trabalho de Conclusão de Curso apresentado ao curso de Pós-Graduação em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2022

Aprendizado de Máquina Aplicado a Análise de Sentimento em Avaliações de Usuários de Aplicativos do Google Play

Josué Leal Evangelista
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil
lealjosue@hotmail.com

Jaime Wojciechowski
Setor de Educação Profissional e Tecnológica
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil
jaimewo@ufpr.br

***Resumo*—O estudo apresentado neste trabalho trata da técnica de aprendizado de máquina supervisionada aplicada a análise de sentimento. Essa técnica foi utilizada para predição de sentimentos em avaliações textuais de usuários de aplicativos do Google Play. Para isso, foram usados para previsão classificadores como o Decision Tree, K-Nearest Neighbor (K-NN), Nayve Bayes e o Extreme Gradient Boosting (XGBoost). Esses classificadores foram aplicados em dois datasets: um com saída de duas classes (binário) e outro de três classes (ternário). Além disso, utilizou-se em cada classificação a vetorização Count Vectorizer (CV) e a Term Frequency-Inverse Document Frequency (TF-IDF). A linguagem de programação Python com algumas de suas bibliotecas e módulos foi utilizada nas etapas do trabalho. Os melhores resultados na classificação binária foram obtidos pelo modelo XGBoost com acurácias de 0,9380 com a vetorização CV e 0,9382 com a TF-IDF. Em segundo lugar, o Naive Bayes com acurácias de 0,9312 com a vetorização CV e 0,9264 com a TF-IDF. Na classificação multiclasse, manteve-se as colocações, uma vez que o XGBoost obteve acurácias de 0,7281 com a vetorização CV e 0,728 com a TF-IDF e o Naive Bayes alcançou acurácias de 0,7081 e 0,7078 com a vetorização CV e TF-IDF respectivamente.**

Palavras-chave*—*análise de sentimento, aprendizado de máquina, Extreme Gradient Boosting, Naive Bayes.

Abstract*— *The study presented in this paper is about the supervised machine learning technique applied to sentiment analysis. This technique was used for sentiment prediction in textual user ratings of Google Play applications. To do this, classifiers such as Decision Tree, K-Nearest Neighbor (K-NN), Nayve Bayes, and Extreme Gradient Boosting (XGBoost) were used for prediction. These classifiers were applied to two datasets: one with two classes (binary) and the other with three classes (ternary). In addition, Count Vectorizer (CV) and Term Frequency-Inverse Document Frequency (TF-IDF) were used in each classification. The Python programming language with some of its libraries and modules was used in the steps of the work. The best results in binary classification were obtained by the XGBoost model with accuracies of 0.9380 with CV vectorization and 0.9382 with TF-IDF. In second place was the Naive Bayes, with accuracies of 0.9312 with CV vectorization and 0.9264 with TF-IDF. In the multi-class classification, the placements were maintained, as XGBoost obtained accuracies of 0.7281 with CV vectorization and 0.728 with TF-IDF, and Naive Bayes achieved accuracies of 0.7081 and 0.7078 with CV vectorization and TF-IDF respectively.

Keywords*— *sentiment analysis, machine learning, Extreme Gradient Boosting, Naïve Bayes.

I. DESENVOLVIMENTO

As opiniões podem ser fundamentais para o processo de tomada de decisões das pessoas [1]. Para as empresas, obter as opiniões de seus consumidores ou público-alvo acerca de produtos e serviços é algo de grande relevância [2].

Com o crescente uso das redes sociais, blogs e sites de e-commerce, percebe-se o grande volume de opiniões que estão sendo geradas. Em sites de compras como Amazon, Mercado livre e AliExpress, por exemplo, podem ser notados milhares de comentários e avaliações de clientes, que revelam suas opiniões em relação a produtos ou serviços. Essas avaliações são realizadas geralmente através de expressões textuais e notas (*ratings*) dos próprios clientes mostrando o nível de satisfação com um item ou serviço adquirido.

Essas expressões textuais geradas nas avaliações desses sites, ou em outros meios de comunicação da Internet, podem ser de grande importância para se obter a real opinião de pessoas. Porém, são necessárias técnicas de análise de textos para extrair o sentimento ou opinião a partir dessas expressões. Para isso, tem-se um campo de estudo denominado de análise de sentimento, que dispõe de técnicas para extração de sentimentos ou opiniões de textos.

A análise de sentimento, ou mineração de opiniões, é uma área voltada para análise de textos que expressam sentimentos, opiniões e emoções de pessoas acerca de produtos, serviços, organizações, indivíduos, eventos, questões e tópicos [2]. Conforme [2][3], a análise de sentimento tem o objetivo de identificar as opiniões negativas e positivas nos textos em relação a algo. Por fim, [3] conclui que a análise de sentimento

tem por objetivo detectar expressões de sentimento, polaridade e força das expressões e sua ligação com o assunto. Conforme [4], a polaridade é um conceito que representa o grau do quanto um texto é positivo ou negativo ou também pode se tratar de uma solução discreta de saída binária (positivo ou negativo) ou ternária (positivo, negativo ou neutro).

Uma das técnicas para se obter as opiniões ou saídas (binária ou ternária) de um texto é a de aprendizagem de máquina supervisionada para classificação [4]. Nessa técnica, um modelo de aprendizado de máquina é treinado com sentenças rotuladas previamente. Esse modelo aprende com as características das sentenças já rotuladas e depois torna-se capaz de prever o sentimento de novas sentenças [4]. Com isso, é possível realizar a previsão de sentimentos de opiniões textuais de forma automática.

Neste contexto, para este estudo foram utilizadas e comparadas quatro técnicas para aprendizado de máquina supervisionado – *Decison Tree, K-Nearest Neighbor (K-NN), Nayve Bayes e* o *Extreme Gradient Boosting (XGBoost)* – para predição binária e ternária ou multiclasse de sentimentos em textos de avaliações. Essas técnicas foram aplicadas a dados coletados de opiniões de usuários em relação a usabilidade dos principais aplicativos de *marketplace* no Goolgle Play. As principais ferramentas utilizadas nessas atividades foram a linguagem de programação Python e o Anaconda Jupyter Notebook.

A. Descrição dos Dados

Neste trabalho, a base de dados foi criada a partir da coleta de dados de avaliações de usuários de alguns dos principais aplicativos do Google Play. Foi criada uma base inicial com 2.300.618 registros de comentários. Dessa base, foram amostradas 319.143 *reviews* para realizar os experimentos. O conjunto da amostra possui 106.381 registros de cada classe (negativo, neutro e positivo).

Os atributos iniciais do conjunto de dados descrevem os detalhes de cada registro de avaliações feitas por usuários a respeito do aplicativo de compra utilizado. Para este experimento, apenas os atributos *content* e *score*, especificados na Tabela I, foram relevantes para as etapas que serão descritas a seguir. Os comentários dos usuários trabalhados neste experimento estão contidos no atributo *content*.

B. Métodos

O estudo foi realizado através de procedimentos divididos em algumas etapas. Ao todo, a metodologia foi composta por 5 etapas principais: 1) Montagem da base de dados, 2) pré-processamento dos dados, 3) preparação do treinamento, 4) treinamento e 5) predição do modelo. Essas etapas foram compostas por um conjunto de atividades inerentes a cada uma delas. O resultado deste trabalho foi um modelo de aprendizado de máquina capaz de classificar o sentimento de novas sentenças de avaliações. O fluxograma do método do trabalho está apresentado na Figura 1.

1) Montagem da base de dados: Essa etapa consistiu na coleta e criação de uma base de dados para ser utilizada na fase de pré-processamento.

Para este experimento, foi preciso coletar os dados para criar a base de dados que foi utilizada no processo de classificação supervisionada de inferência de máquina. Em alguns casos, é preciso extrair dados de alguma fonte para criação de uma base de dados necessária para o estudo. Pode-se observar isso no estudo realizado por Crescensio, Gonçalves e Todesco [5] em que os dados usados foram extraídos do site TripAdivisor para que pudessem ser utilizados na aprendizagem de máquina.

Neste trabalho, como já mencionado, a base de dados foi criada a partir da extração de registros com comentários de usuários de aplicativos de compras do Google Play. Para isso, foram utilizadas a linguagem de programação Python e funções do módulo *google-play-scraper*. Esse módulo permite realizar a raspagem e obter os dados de avaliações de aplicativos do Google Play. Com suas funções é possível coletar detalhes, *reviews*, permissões e realizar pesquisas de um aplicativo específico.

TABELA I
ATRIBUTOS INICIAIS DO CONJUNTO DE DADOS.

Atributo	Descrição
reviewId	Identificador exclusivo da avaliação.
userName	Nome do usuário que fez a avaliação.
userImage	Link contendo a imagem do usuário.
content	Campo com o comentário do usuário.
score	Nota dada pelo usuário de 1 a 5.
thumbsUpCount	Número de usuários que marcaram o comentário da avaliação como "Gostei".
reviewCreatedVersion	Versão de criação do comentário.
at	Data e horário de criação do comentário do usuário.
replyContent	Resposta ao comentário do usuário.
repliedAt	Data da resposta ao comentário do usuário.

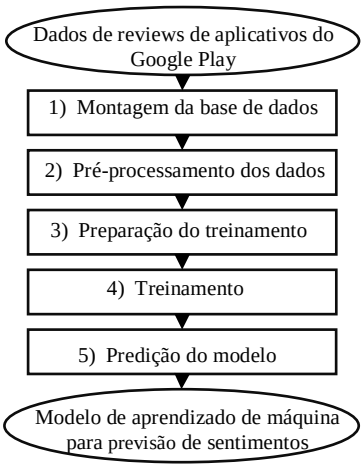

Fig. 1. Etapas do método aplicado no estudo.

Através da função de busca de *reviews* (reviews_all) do módulo citado e da linguagem Python, foram obtidas 2.300.618 avaliações com comentários de usuários desses aplicativos. A Figura 2 (a) apresenta esse total de avaliações distribuído em cada nota. Essas notas dadas pelos usuários dos aplicativos são de 1 a 5 estrelas. Dessa forma, utilizou-se uma formulação de problema de classificação descrita em [2], em que os registros com notas 1 e 2 foram rotulados como negativos, os comentários com nota 3 como neutros, e os de nota 4 e 5 ganharam a rotulação de positivos, conforme mostra a Figura 2 (b) a seguir. Neste estudo a classe neutro será descartada para o problema de classificação binária, mas será utilizada no ternário.

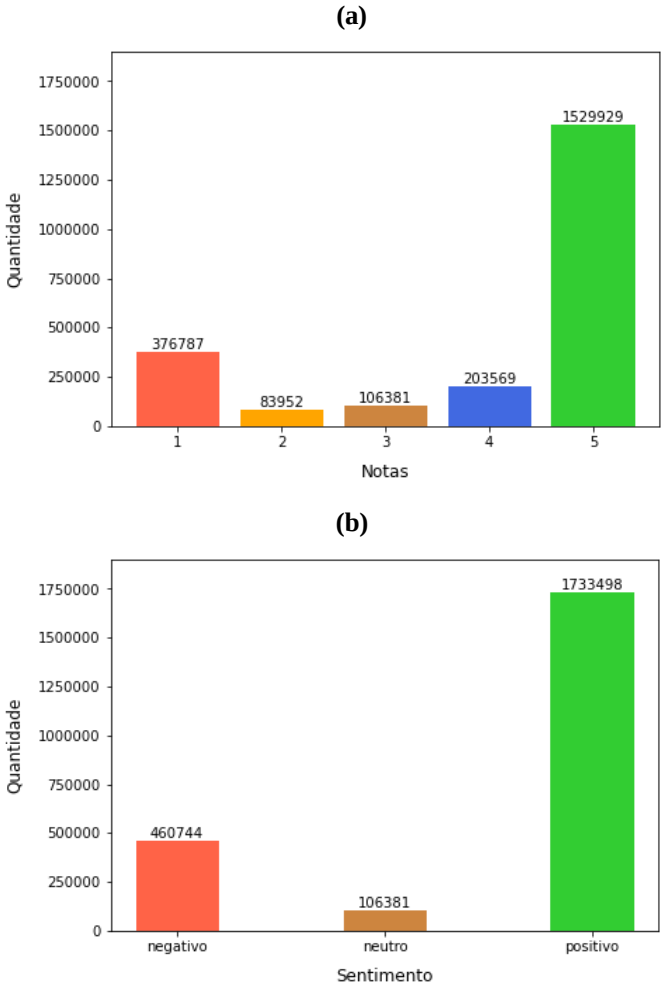

Fig. 2. (a) Quantidade de avaliações por notas; (b) Quantidade de avaliações por sentimento após transformação dos rótulos.

Em seguida, para criação dos *datasets*, foram coletados todos os dados da classe neutra que totalizavam 106.381 e amostrados 106.381 para cada uma das classes restantes (negativo e positivo) como é possível observar na Figura 3. Esses registros totalizaram 319.143 *reviews* com a soma das três classes citadas. A partir disso, foram formados dois *datasets* para previsão de duas (negativo e positivo) e três classes (negativo, neutro e positivo). No total, o conjunto de dados binário ficou com 212.762 registros e conjunto ternário com 319.143.

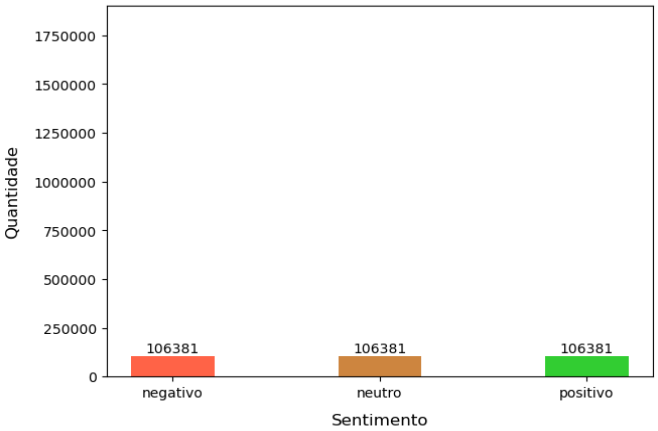

Fig. 3. Quantidade de avaliações por sentimento balanceada.

2) Pré-processamento dos dados: Essa etapa recebeu o conjunto de dados e teve como objetivo a prepararação dos dados para as fases de preparação do treinamento, treino e predição do modelo. Nessa etapa de pré-processamento, foram realizadas cinco atividades: tokenização, lematização, *bag of words* , vetorização - *Count Vectorizer (CV)* e *Term Frequency-Inverse Document Frequency (TF-IDF ou tf-idf*) - e seleção de variáveis.

A tokenização consiste na divisão de uma entrada de texto em unidades básicas ou *tokens* [6]. No caso, cada texto de entrada foi dividido em palavras. Durante esse processo, foram empregadas quatro etapas para limpeza dos textos de opiniões. Iniciou-se com a remoção de caracteres desnecessários como acentos, pontuações e números, e com a conversão em letras minúsculas. Em seguida, foram realizadas a correção ortográfica e de digitação através de dicionários formados com palavras coletadas dos textos das avaliações. Na sequência, foi criada uma lista para exclusão de palavras não úteis para o experimento e de erros de digitação que não puderam ser corrigidos. Outra correção realizada na sequência foi a de palavras alongadas do texto. Por último, foram removidas as *stopwords*, que conforme [4], são palavras não relevantes para o sentimento do texto podendo ser representadas

Geralmente as *stopwords* podem ser preposições, conectivos, pronomes ou artigos. Para removê-las, foi criada uma função que recebeu como base uma lista formada a partir das *stopwords* em português da biblioteca NLTK e de outras palavras que foram acrescentadas a essa lista.

Na Figura 4 (a) e (b) é possível observar o antes e o depois das atividades de limpeza e eliminação das *stopwords*. Após a remoção das *stopwords*, foi necessário verificar se os *datasets* possuíam algum registro em branco no campo de texto, pois algumas das atividades citadas consistiram em remoção de palavras. Após essas atividades, as sentenças processadas foram utilizadas na etapa de lematização descrita a seguir.

(a)

(b)

Fig. 4. Nuvens de palavras: (a) Antes do pré-processamento e (b) Após limpeza e remoção das *stopwords*.

Lematização é o processo em que as palavras como adjetivos ou substantivo são representadas pelo seu masculino no singular e os verbos pelo seu infinitivo [7]. Essa etapa contribui para minimizar a alta dimensionalidade, pois permite que um conjunto de palavras venha a ser representado por uma única palavra. Isso faz com que ocorra uma redução da quantidade de atributos formados pelas palavras dos textos, conforme apresentado na Tabela II. Para aplicar a lematização uma função foi desenvolvida e teve como base um dicionário composto por 850.264 mil pares para substituição de adjetivos e substantivos pelo masculino singular e os verbos pelo infinitivo. Esse dicionário está disponibilizado em um banco de dados no GitHub [8]. O resultado do processo alimenta a etapa de *Bag of Words* descrita a seguir.

TABELA II
REDUÇÃO DO NÚMERO DE ATRIBUTOS

Atividade	Quantidade de *features*
Início	60.439
Remoção das stopwords	53.680
Lematização	37.022

A *Bag of Words* é um método em que a coleção de sentenças (textos) é transformada em *tokens* de palavras formando um único vetor em que as palavras não se repetem. Esse vetor pode ser representado por unigramas (uma palavra) ou n-gramas (duas ou mais palavras) como é o caso do bigrama (duas palavras).

Neste estudo, foi utilizada a representação de unigrama apenas, que consiste na separação do texto em *tokens* de uma palavra. No caso de bigramas, que também é amplamente utilizado, o texto também é separado em *tokens*, porém com duas palavras cada, sendo cada palavra do texto unida a sua sucessora do mesmo texto ou sentença.

Após a *Bag of Words*, tem-se a etapa de vetorização, que visa a transformação dos unigramas em uma matriz na qual as palavras são representadas numericamente. Os valores de cada palavra na matriz baseiam-se em sua ocorrência em um mesmo texto ou também em sua frequência no total de comentários. Na sequência, essa matriz resultante é utilizada para treinar os modelos de aprendizado de máquina.

Neste estudo, foram utilizados dois tipos de vetorização presentes na documentação da biblioteca scikit-learn [9], a de contagem de palavras *Count Vectorizer (CV)* com a função *CountVectorizer* e a *Term Frequency-Inverse Document Frequency* (*tf-idf)* com a função *TfidfTransformer*. A primeira se baseia na vetorização mais simples que é a frequência do termo no documento. Nesse caso, uma matriz de representação numérica das palavras é formada através da frequência do termo (palavra) no documento (comentário do usuário). Já a segunda baseia-se no modelo *tf-idf*, que além de utilizar a frequência do termo no documento *tf*, também incorpora o *idf* que é a frequência de documento inversa do termo. Nesse modelo *tf-idf*, cada termo ou palavra recebe um peso calculado, que tem como objetivo reduzir o impacto de palavras que ocorrem com muita frequência no total de textos, e que por isso são consideradas menos significativas. Para utilizar a função *TfidfTransformer* da biblioteca scikit-learn é necessário que antes seja utilizada a função *CountVectorizer* para vetorizar as palavras pela frequência do termo no documento.

No *tf-idf* clássico, visto no estudo de Paltoglou e Thelwall [10], o resultado do *tf-idf* é obtido conforme Equação 1, sendo que w_i é o peso atribuído ao termo i no documento, tf_i é o número de vezes que o termo i ocorre no documento, idf_i é a frequência de documento inversa do termo i, N é o número total de documentos ou textos e df_i é o número de documentos que possui o termo i .

$$w_i = tf_i \cdot idf_i = tf_i \cdot log\frac{N}{df_i} \quad (1)$$

No modelo *tf-idf* da biblioteca scikit-learn [9], o cálculo pode ser feito através de duas equações que possuem diferenças no *idf* em relação a Equação 1. Na Equação 2, o valor 1 é adicionado ao idf_i e na Equação 3, além da adição de 1 ao idf_i, o numerador e denominador também são adicionados em 1, conforme as equações a seguir:

$$w_i = tf_i \cdot idf_i = tf_i \cdot log\frac{N}{df_i} + 1 \quad (2)$$

$$w_i = tf_i \cdot idf_i = tf_i \cdot log\frac{1+N}{1+df_i} + 1 \quad (3)$$

Conforme consta na biblioteca [9], para que a função *TfidfTransformer* utilize a Equação 2 ou 3, é necessário que o hiperparâmetro *smooth_idf* da função seja acionado com valores booleanos (smooth_idf=False ou smooth_idf=True). Quando o *smooth_idf* é igual a "*false",* o idf_i da Equação 2 é usado. Já quando o *smooth_idf* é "*True*" (padrão na função *TfidfTransformer*), o idf_i da Equação 3 é utilizado.

Neste trabalho, foi utilizado *smooth_idf* igual a "*True*" que aciona o uso da Equação 3, que é a padrão usada para vetorização *tf-idf* da função *TfidfTransformer* da biblioteca scikit-learn [9]. Com a aplicação dessa função, um novo *array* com vetores numéricos é formado sobre o da vetorização *CountVectorizer*. O *TfidfTransformer*, além de formar esses novos vetores numéricos, aplica uma normalização em cada um desses vetores (linhas do *array*) utilizando-se da norma euclidiana conforme Equação 4 a seguir.

$$v_{norm} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{v_1^2+v_2^2+\cdots v_n^2}} \quad (4)$$

Na Equação 4, v é o vetor (linhas do *array*) formado após o cálculo do *tf-idf pela* função *TfidfTransformer*, $\|v\|_2$ a norma euclidiana e v_{norm} o vetor normalizado resultante.

Por último, foi feita a seleção de características ou atributos utilizando a função de seleção de variáveis *SelectKBest* presente na biblioteca scikit-learn [9]. Essa função possui como hiperparâmetros o *score_func*, que permite a escolha da estatística a ser utilizada para seleção de variáveis e o hiperparâmetro *k* que é a quantidade de variáveis a serem selecionadas.

Para o *score_func*, utilizou-se a opção *chi2* que representa o teste chi-quadrado que pode ser utilizado em atividades de seleção de variáveis em classificação de texto, conforme descrito na biblioteca scikit-learn [9]. Com a escolha do teste chi-quadrado no hiperparâmetro *score_func*, as variáveis foram selecionadas pela função através da melhor pontuação obtida com base no resultado do *p-value* de cada variável.

Já para o hiperparâmetro *k* foram utilizados os valores de 20.000 para seleção de variáveis no dataset binário que continha 28.123 variáveis e de 30.000 para seleção no multiclasse que possuía 37.022. Esses valores de *k* foram escolhidos visando a redução dos atributos sem que houvesse uma possível perda de atributos importantes, o que pode acontecer caso sejam escolhidos valores muito baixos.

Além de escolher os atributos mais relevantes, a atividade de seleção, assim como a lematização, contribuiu para redução do número de variáveis. Após essas atividades de pré-processamento, os dados ficaram prontos para serem utilizados nas etapas de treinamento e predição dos modelos descritas neste trabalho.

3) Preparação para o treinamento: A etapa de preparação para treino dos modelos teve início com a definição do método para divisão de treino e teste dos dados pré-processados e as métricas para avaliação de desempenho dos modelos treinados.

O método de divisão aplicado aos conjuntos de dados no treinamento foi uma divisão de treino seguindo a proporção 70/30 (70% do *dataset* para treino e 30% para teste). Conforme [11], deve-se considerar um valor de divisão acima de 50% para treino, sendo os valores típicos de 2/3 para treino e 1/3 para teste. Também de acordo com [11] não existe fundamento teórico a respeitos desses valores. Em [12], por exemplo, é informado que no serviço Amazon *Machine Learning* utiliza-se uma estratégia geralmente de 70% a 80% de dados para treino e de 20% a 30% para teste. Então, baseado no que foi mencionado, foi aplicada a proporção escolhida de 70/30 tanto para classificação binária quanto para multiclasse, conforme mostra a Tabela III a seguir.

TABELA III
DIVISÃO DOS DADOS DE TREINO E TESTE

Dataset	Divisão 70/30		
	Total	Treino	Teste
Binário	211.593	148.115	63.478
Ternário	316.911	221.837	95.074

Nessa etapa também foram definidas a aplicação de técnicas de busca em *grid* (*grid search*) e validação cruzada estratificada (*estratificate cross validation*) que foram aplicadas aos dados de treino. A partição (*folds)* para validação cruzada foi de 10, que é o valor normalmente utilizado de acordo com [13]. Na validação cruzada estratificada, as partições são criadas com a mesma distribuição de classes do conjunto de dados em que foi aplicada. Por exemplo, ao aplicar em um conjunto de dados de treino em que 70% das observações são da classe "positivo" e 30% da classe "negativo", cada *fold* seria composto 70% de observações da classe "positivo" e 30% da "negativo".

Neste estudo, utilizou-se algumas métricas de desempenho usadas em problemas supervisionados de classificação conforme [10]. Portanto, foram usadas métricas como a acurácia (*accuracy*), sensibilidade (*recall)*, precisão (*precision*), *f1-score* e especificidade, que são calculadas a partir da matriz de confusão.

A matriz de confusão é usada nos problemas de classificação para que se possa verificar a quantidade acertos do modelo de aprendizado de máquina na previsão de cada classe [11]. Essa matriz faz com que seja possível a avaliação do resultado entre as classes preditas e as classes de referência (conhecidos) como mostra a Figura 4. Em um problema de classificação supervisionado de duas classes, por exemplo, tem-se as classes de referência positivo (P) e negativo (N). Em uma previsão correta das classes positivo e negativo, são obtidos como resultados os verdadeiros positivos (TP) e os verdadeiros negativos (TN) respectivamente. Para predições erradas das classes positivos e negativos são obtidos falsos positivos (FP) e os falsos negativos (FN). O falso positivo é o erro cometido quando o modelo prevê positivo quando na verdade a classe era negativo. Já o falso negativo é o erro que o modelo comete quando prevê a classe negativo sendo que na verdade era positivo.

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	Verdadeiros Positivos TP	Falsos Negativos FN
	Negativo	Falsos Positivos FP	Verdadeiros Negativos TN

Fig. 4. Matriz de confusão binária.

Após o resultado da matriz de confusão, é possível realizar o cálculo das métricas - acurácia, sensibilidade, precisão e *F1-score* - para avaliar o desempenho dos modelos. Os cálculos para as medidas estão descritos a seguir conforme equações.

- Acurácia: medida que informa o percentual de acerto do modelo [10]. Essa medida é definida pela razão da soma dos acertos pelo total da amostra.

$$acurácia = \frac{(TP+TN)}{n} \quad (5)$$

- Sensibilidade: mede a capacidade do modelo de detectar de forma correta a classe de positivos e pode ser resumida como a taxa de acertos de verdadeiros positivos conforme [10].

$$sensibilidade\ (recall) = \frac{TP}{(TP+FN)} \quad (6)$$

- Precisão: Conforme [10] é a medida que mostra a proporção de classificação correta da classe positivo dentre todos os que foram preditos como positivos.

$$precisão = \frac{TP}{(TP+FP)} \quad (7)$$

- *f1-score*: de acordo com [10] é a média harmônica ponderada entre precisão e sensibilidade.

$$f1-score = \frac{2\ (precisão \cdot recall)}{(precisão+recall)} \quad (8)$$

4) Treinamento: Neste experimento, as técnicas de aprendizado de máquina utilizadas foram *Decision Tree, K-Nearest Neighbor (K-NN), Nayve Bayes e o Extreme* G*radient* B*oosting (XGBoost).* Para o treinamento, todas as técnicas foram testadas com validação cruzada estratificada. A busca em *grid* de hiperparâmetros foi utilizada em todos os métodos, exceto o *Naive Bayes*. Os hiperparâmetros dos algoritmos foram definidos conforme descrito a seguir.

A etapa de treinamento iniciou-se com a técnica *Decision tree* ou Árvore de Decisão. Para essa técnica, testou-se o hiperparâmetro de critério de medidas de qualidade de divisão (*criterion*). Para o *criterion* foram testados "*gini*", "*entropy*". Os outros hiperparâmetros do *Decision Tree* foram testados com seus valores padrão como consta na biblioteca scikit-learn [9].

Na técnica *K-NN* foi utilizado na busca em *grid* apenas o hiperparâmetro k-vizinhos mais próximos (*n_neighbors)*. Conforme [13], o valor de k usado é frequentemente pequeno e ímpar, pois valores ímpares evitam empates na votação da classe em problemas de classificação. Com isso, o k foi determinado a partir de testes com k = 5, 9, 15, 25 e 35. Para os outros hiperparâmetros como, por exemplo, o de distância, optou-se por deixar o padrão que é a distância de Minkowski, que é uma generalização da distância de Manhattan e Euclidiana também utilizadas pelo algoritmo.

A técnica *Nayve Bayes* foi utilizada e treinada na forma padrão conforme hiperparâmetros do algoritmo na documentação da biblioteca scikit-learn [9]. Para essa técnica não se utilizou a busca em *grid* para obter seus melhores hiperparâmetros, apenas utilizou-se a validação cruzada estratificada. Neste estudo, foi utilizado o classificador *Multinomial Naive Bayes* (*MultinomialNB*) disponibilizado como um algoritmo apropriado para a atividade de classificação de textos conforme consta na biblioteca [9].

Por fim, para o uso do algoritmo *Extreme Gradient Boosting*, foram testados alguns dos seus hiperparâmetros conforme observados na Tabela IV.

TABELA IV

HIPERPARÂMETROS DO XGBOOST

Hiperparâmetros	**Descrição**
n_estimators	Quantidade de classificadores ou estimadores.
learning_rate	Taxa de aprendizagem. Intervalo: [0,1].
colsample_bytree	Proporção de subamostra de coluna ao construir cada árvore. Intervalo: (0, 1).
max_depth	Profundidade máxima de árvore. Intervalo: [0,∞].
subsample	Amostragem aleatória dos dados antes de cultivar as árvores. Intervalo: (0,1].
reg_alpha	Termo de regularização sobre pesos.
objective	Função objetivo ou função de perda. • **binary:logistic** para classificação de saída binária; • **multi:softmax** para classificação de saída multiclasse.

No *XGBoost* foram utilizados alguns valores em seus hiperparâmetros, conforme descrito a seguir. Para *n_estimators* de usou-se 100, 300, 700, 1000 e 1100 árvores, *learning_rate* de 0.01, 0.2 e o padrão 0.3, *colsample_bytree* de 0.7, *max_depth* padrão de 6, *subsample* de 0.7 e *reg_alpha* de 0.05. Já para o *objective* foi usado o *binary:logistic* para o *dataset* binário e o *multi:softmax* para o multiclasse.

Para evitar um consumo alto de memória na utilização do *XGBoost*, optou-se pela quantidade de estimadores máxima de 1100 e pela profundidade padrão de 6 para cada árvore. Além disso, aumentar o valor de profundidade de árvores em mais de 6 níveis pode levar a sobreajuste (*overfitting)* do modelo de acordo com a documentação do *XGBoost* [14]. Da mesma forma, para evitar *overfitting*, a taxa de aprendizagem foi usada com três valores contidos no intervalo de 0 a 1, sendo dois deles valores menores que o padrão do algoritmo (padrão = 0.3).

Para também evitar o sobreajuste do *XGBoost*, o hiperparâmetro *subsample* foi o utilizado com o valor de 0.7. Conforme documentação [14], o *XGBoost* utiliza um *subsample* padrão de valor 1, mas segundo a própria documentação, definilo com o valor 0.5 faz com que seja evitado o excesso de ajuste. Com isso, foi escolhido um valor próximo do intermediário entre 0.5 e 1. Já para o *reg_alpha*, optou-se pelo valor 0.05, que é maior que o valor padrão (padrão = 0) usado pelo algoritmo e que tem por objetivo um modelo mais conservador para também evitar sobreajuste conforme descrito na documentação [14].

Após o treinamento, cada modelo com seus melhores hiperparâmetros selecionados foram utilizados para a predição com os dados de teste, conforme descrito a seguir. Os resultados de acurácia de treino estão apresentados na Tabela V.

5) Predição : Nesta etapa foram realizadas as previsões utilizando-se os dados de teste com os modelos treinados.

Conforme treinamento, cada modelo foi ajustado através da busca em *grid* e o modelo de melhor desempenho de cada técnica de aprendizagem aplicada foi utilizado para o teste. A Tabela VI apresenta os resultados da acurácia de teste dos classificadores aplicados. Já os resultados individuais de cada modelo e seus melhores hiperparâmetros obtidos a partir do *grid* estão apresentados nas Tabelas de VII a XXII em anexo.

C. Tecnologias

Neste estudo, o *script* foi feito em um computador pessoal Samsung com processador intel (R) Core (TM) i7-5500U, 2 núcleos, CPU 2.40GHz, RAM instalada de 8,00 GB e sistema operacional Windows 10 Home. Também foram utilizadas as seguintes ferramentas:

- Linguagem de programação Python, versão 3.9.13;
- Jupyter Notebook, versão 6.4.5;
- Módulo google-play-scraper, versão 1.2.2;
- Biblioteca pandas, versão 1.4.4;
- Biblioteca numpy, versão 1.21.5;
- Biblioteca matplotlib, versão 3.5.2;
- Módulo re , versão 2.1.1;
- Biblioteca NLTK, versão 3.7;
- Biblioteca scikit-learn, versão 0.24.2;
- Biblioteca xgboost, versão 1.6.2;
- Biblioteca joblib, versão 1.1.0;
- Biblioteca wordcloud, versão 1.8.2.2.

II. RESULTADOS E DISCUSSÕES

Nesta seção estão demonstrados os resultados obtidos neste estudo. Os resultados alcançados na etapa de predição dos modelos com os dados de teste estão apresentados nas próximas subseções.

A subseção a seguir apresenta as medidas das técnicas aplicadas e, a subseção seguinte mostra o desempenho dos modelos com os resultados de acurácia na etapa de treino e teste. Mais detalhes de resultados dos modelos na etapa de teste podem ser vistos nas Tabelas de VII a XXII em anexo a este documento. Este trabalho tem por finalidade comparar os resultados dos modelos aplicados a previsão binária e a multiclasse utilizando-se das vetorizações *CV* e *TF-IDF*. Para comparação das acurácias, a Tabela VI apresenta resumidamente os resultados obtidos por cada modelo em cada tipo de classificação na etapa de teste.

A. Medida de Qualidade dos Modelos

As medidas de desempenho utilizadas neste trabalho foram a acurácia, sensibilidade, precisão e *f1-score*. Para definição dos melhores modelos, utilizou-se a acurácia como a principal métrica. Os melhores resultados de métricas foram obtidos pelo *Extreme Gradient Boosting* nos dois tipos de classificação conforme pode ser observado na Tabela VI. As Tabelas de XIX a XXII em anexo apresentam os resultados mais detalhados do *XGBoost* como suas matrizes de confusão e métricas como *recall*, precisão e *f1-score*. O algoritmo *Naive Bayes* também se destacou e teve métricas próximas as do *XGBoost.* O menor desempenho de modelo quanto as métricas, no geral, foi o *K-NN*, conforme apresentado na Tabela VI e nas Tabelas em anexo de XI a XIV.

B. Desempenho dos Modelos

Nesta subseção estão apresentados nas Tabelas V e VI os resultados das acurácias dos modelos na etapa de treino e teste respectivamente. Os detalhes individuais de cada modelo na etapa de teste foram detalhados nas Tabelas de VII a XXII, que estão em anexo. Esses detalhes individuais possuem, além das matrizes de confusão e relatórios, os hiperparâmetros utilizados neste experimento.

TABELA V

RESULTADOS DE ACURÁCIA NO TREINAMENTO DOS MODELOS

Técnicas	CV-binário	TFIDF-binário	CV-ternário	TFIDF-ternário
	acurácia	acurácia	acurácia	acurácia
Decision Tree	0,8987	0,9026	0,6452	0,6476
K-NN	0,8939	0,7417	0,6519	0,5621
Nayve Bayes	0,9302	0,9244	0,7075	0,7073
XGBoost	**0,9370**	**0,9362**	**0,7262**	**0,7263**

TABELA VI

RESULTADOS DE ACURÁCIA NO TESTE DOS MODELOS

Técnicas	CV-binário	TFIDF-binário	CV-ternário	TFIDF-ternário
	acurácia	acurácia	acurácia	acurácia
Decision Tree	0,9023	0,9050	0,6460	0,6491
K-NN	0,8971	0,7441	0,6513	0,5672
Nayve Bayes	0,9312	0,9264	0,7081	0,7078
XGBoost	**0,9380**	**0,9382**	**0,7281**	**0,7280**

Na Tabela VI, percebe-se que o modelo de método *ensemble Extreme Gradient Boosting* se destacou quanto aos outros. Seus resultados foram os melhores, tanto para a classificação binária quanto a multiclasse. Esse algoritmo obteve acurácias de 0,9380 e 0,9382 para classificação binária com os tipos de vetorização *CV* e *TF-IDF* e 0,7281 e de 0,728 para ternária também com as vetorizações *CV* e *TF-IDF* respectivamente.

Quanto aos outros modelos, destacou-se na previsão binária o *Naive Bayes* com acurácias de 0,9312 e 0,9264 com as vetorizações *CV* e *TF-IDF* respectivamente. O modelo *Decision Tree* obteve acurácias de 0,9023 e 0,905 nas previsões de duas classes. Já o *K-NN*, em comparação com os resultados de previsão binária, teve um bom desempenho apenas com a vetorização *CV*, em que obteve acurácia de 0,8971. Na classificação binária com utilização da vetorização *TF-IDF*, o *K-NN* obteve o resultado de acurácia mais baixo entre os outros algoritmos com 0,7441.

Tendo em vista que o maior resultado de acurácia para a classificação multiclasse foi do *Extreme Gradient Boosting* com 0,7281, percebe-se que no geral o desempenho para esse tipo classificação foi baixo. A classe de neutro dificultou o processo de previsão fazendo com que os algoritmos tivessem um menor desempenho, conforme apresentado nas matrizes de confusão das Tabelas de VII a XXII em anexo.

C. Discussões

Este estudo teve como objetivo avaliar o desempenho de técnicas de aprendizado de máquina para previsão de sentimentos de textos de avaliações de usuários de aplicativos. As técnicas aplicadas aos dois *datasets*, um de saída binária e outro multiclasse utilizaram-se das vetorizações *CV* e *TF-IDF*.

Conforme desenvolvimento deste trabalho, percebeu-se como algumas etapas foram fundamentais para que se chegasse a um bom modelo preditivo. Dentre as etapas, a de pré-processamento dos dados com as atividades de tokenização e lematização foram fundamentais. Essas atividades contribuíram bastante para redução de atributos, o que foi muito importante para a etapa de treinamento dos modelos.

A partir das etapas de treino e teste, pode-se concluir que os melhores resultados de acurácia e outras métricas foram obtidos na classificação binária. Com exceção do modelo *K-NN* na vetorização *TF-IDF*, os modelos tiveram boas performances para o *dataset* binário. O método *ensemble Extreme Gradient Boosting* teve destaque neste trabalho e atingiu desempenho satisfatório no *dataset* binário obtendo a melhor acurácia. Embora seja um algoritmo que exige um tempo considerável de treino para *dataset* com alta dimensionalidade, o *XGBoost* pode alcançar bons resultados em atividades de classificação de sentimentos e classificação no geral. Em trabalhos como [15][16] por exemplo, percebe-se como o *XBoost* se destaca em problemas de classificação comparado a outros algoritmos. Outra técnica que atingiu um bom resultado de classificação para duas classes foi a *Multinomial Naive Bayes* que chegou bem próximo do modelo de método *ensemble* citado. Uma grande vantagem da técnica *Naive Bayes* neste trabalho foi a sua velocidade de treino e predição. Para trabalhos de classificação como este em que se tem muitos atributos ou alta dimensionalidade, um modelo que consuma pouco recurso computacional torna-se bastante apropriado.

Na classificação multiclasse, os modelos no geral não tiveram um bom desempenho. Os erros de classificação cresceram substancialmente devido à dificuldade de discriminar a classe neutro. Muitos rótulos reais de neutros foram classificados como positivos e principalmente negativos. Na classe neutro, por exemplo, é possível encontrar textos que poderiam ser mais adequados como sentenças negativas ou positivas.

Para possíveis melhores resultados neste tipo de atividade, pode-se também recorrer a outras técnicas de aprendizado de máquina. Algumas técnicas como, por exemplo, a utilização do algoritmo *Random Forest* pode ser utilizada para classificação de textos binária ou multiclasse em trabalhos futuros.

Outra forma que possibilita alcançar possíveis melhores resultados é a utilização de outras abordagens para análise de sentimentos. Métodos como de aprendizagem profunda mais o uso de outros tipos de vetorização como os que utilizam *word embedding vectors* (*word2vec)* ou suas variantes, são bastante utilizados e podem alcançar bons resultados conforme pode ser visto em [5][17]. Também pode-se optar por métodos não supervisionados com abordagens léxicas. Conforme [4], a abordagem léxica possui grande vantagem, pois permite que a aplicação não fique limitada a um contexto de treino como na classificação por aprendizagem supervisionada utilizada neste trabalho. No estudo de Reis, Gonçalves et al. [18], por exemplo, pode ser vista a aplicação de 13 métodos de análise de sentimentos em *tweets* de diferentes idiomas.

REFERÊNCIAS

[1] E. Cambria, B. Schuller, Y. Xia and C. Havasi, "New Avenues in Opinion Mining and Sentiment Analysis," in IEEE Intelligent Systems, vol. 28, no. 2, pp. 15-21, March-April 2013, doi: 10.1109/MIS.2013.30.

[2] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*, Cambridge Univ. Press, 2015.

[3] T. Tasukawa and J. Yi, “Sentiment Analysis: Capturing Favorability Using Natural Language Processing,” in Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003). ACM, 2003, pp. 70–77.

[4] F. Benevenuto, F. Ribeiro, and M. Araújo, “Métodos para análise desentimentos em mídias sociais,” inProc. of the Brazilian Symposium on Multimedia and the Web (Webmedia), 2015.

[5] M. Crescencio, A. L. . Gonçalves, e J. L. Todesco, “UM PROCESSO DE CLASSIFICAÇÃO DE TEXTO: ANÁLISE DE SENTIMENTO DAS OPINIÕES NO TRIPADVISOR SOBRE A ATRAÇÃO OKTOBERFEST BLUMENAU”, ciKi, vol. 1, nº 1, nov. 2020.

[6] R. V. O. Araújo. “A Criação de um modelo Natural Language Processing para extração de habilidades técnicas na área de Ciência de Dados”. Dissertação Mestrado, UNL, Lisboa, PT, 2022.

[7] J. L. De Lucca, M. D. G. V Nunes. “Lematização versus Stemming”. Relatórios Técnicos do ICMC-USP, 14 (NILC-TR-02-22), 2002.

[8] Contains information from https://github.com/michmech/lemmatization-lists, which is made available here under the Open Database License (ODbL). Disponível em: https://github.com/michmech/lemmatization-lists. Acesso 2 set. 2022.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, ‘‘Scikit-learn: Machine learning in Python,’’ J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2011.

[10] G. Paltoglou and M. Thelwall, “A study of information retrieval weighting schemes for sentiment analysis,” in Proc. 48th Annu. Meeting ACL, Uppsala, Sweden, 2010, pp. 1386–1395.

[11] M. C. Monard and A. J. Baranauskas. “Conceitos sobre aprendizado de máquina. Sistemas Inteligentes-Fundamentos e Aplicações”, v.1, nº.1, Barueri, SP, Manole, 2003, p.1.

[12] AWS, Amazon Web Services, “Documentação do Amazon Machine Learning – Guia do desenvolvedor”, 2016. Disponível em: https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/what-is-amazon-machine-learning.html. Acesso em 25 set. 2022.

[13] K. Faceli, A. C. Lorena, J. Gama, and A. Carvalho, “Inteligencia Artificial: Uma abordagem de aprendizado de maquina,” ´ Rio de Janeiro: LTC. 2011.

[14] DMLC XGBoost, “XGBoost Documentation”, 2016. Disponível em: https://xgboost.readthedocs.io/en/stable/#. Acesso em 27 set. 2022.

[15] A. Cavalcanti, R. Mello, P. Miranda, and F. Freitas. "Análise Automática de Feedback em Ambientes de Aprendizagem Online", in Anais do XXXI Simpósio Brasileiro de Informática na Educação, Online, 2020, pp. 892-901, doi: https://doi.org/10.5753/cbie.sbie.2020.892.

[16] D. S. C. Rocha. “Aprendizado de Máquina Aplicado ao Reconhecimento Automático de Falhas em Máquinas Rotativas”. Dissertação de Mestrado, UFMG, Belo Horizonte, BR, 2018.

[17] M. Y. Noguti. “Aplicação de Técnicas de Classificação Textual na Predição de Áreas de Atuação do Ministério Público”. Monografia de Especialização, UFPR, Curitiba, BR, 2019.

[18] J. Reis, P. Gonçalves, M. Araújo et al. "Uma Abordagem Multilíngue para Análise de Sentimentos", in Anais do IV Brazilian Workshop on Social Network Analysis and Mining, Recife, 2015, pp. , doi: https://doi.org/10.5753/brasnam.2015.6767.

TABELA VII

RESULTADOS DECISION TREE CV BINÁRIO

Técnica	**Hiperparâmetros**
Decision Tree	criterion: ‘entropy’

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	28868	2690
	Negativo	3510	28410

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,89	0,91	0,90	0,90
negativo	0,91	0,89	0,90	

TABELA VIII

RESULTADOS DECISION TREE TF-IDF BINÁRIO

Técnica	**Hiperparâmetros**
Decision Tree	criterion: ‘gini’

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	28713	2845
	Negativo	3187	28733

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,90	0,91	0,90	0,90
negativo	0,91	0,90	0,91	

TABELA IX

RESULTADOS DECISION TREE CV MULTICLASSE

Técnica	**Hiperparâmetros**
Decision Tree	criterion: ‘gini’

Matriz de Confusão

		Preditos		
		Positivo	Neutro	Negativo
Conhecidos	Positivo	25821	1491	4203
	Negativo	2030	20576	9274
	Neutro	7321	9336	15022

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,73	0,82	0,77	0,65
negativo	0,66	0,65	0,65	
neutro	0,53	0,47	0,50	

TABELA X

RESULTADOS DECISION TREE TF-IDF MULTICLASSE

Técnica	**Hiperparâmetros**
Decision Tree	criterion: ‘gini’

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	25902	1465	4148
	Negativo	1818	20671	9391
	Neutro	7178	9365	15136

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,74	0,82	0,78	0,65
negativo	0,66	0,65	0,65	
neutro	0,53	0,48	050	

TABELA XI

RESULTADOS K-NN CV BINÁRIO

Técnica	**Hiperparâmetros**
K-NN	algorithm: ‘auto’, leaf_size: 30, metric: ‘minkowski’, n_neighbors: 5, p: 2, weights: ‘uniform’.

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	29547	2011
	Negativo	4518	27402

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,87	0,94	0,90	0,90
negativo	0,93	0,86	0,89	

TABELA XII

RESULTADOS K-NN TF-IDF BINÁRIO

Técnica	**Hiperparâmetros**
K-NN	algorithm: ‘auto’, leaf_size: 30, metric: ‘minkowski’, n_neighbors: 5, p: 2, weights: ‘uniform’.

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	29362	2196
	Negativo	14049	17871

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,68	0,93	0,78	0,74
negativo	0,89	0,56	0,69	

TABELA XIII

RESULTADOS K-NN CV MULTICLASSE

Técnica	**Hiperparâmetros**
K-NN	algorithm: ‘auto’, leaf_size: 30, metric: ‘minkowski’, n_neighbors: 15, p: 2, weights: ‘uniform’.

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	27996	1369	2150
	Negativo	3007	22305	6568
	Neutro	9498	10560	11621

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,69	0,89	0,78	0,65
negativo	0,65	0,70	0,67	
neutro	0,57	0,37	0,45	

TABELA XIV

RESULTADOS K-NN TF-IDF MULTICLASSE

Técnica	**Hiperparâmetros**
K-NN	algorithm: ‘auto’, leaf_size: 30, metric: ‘minkowski’, n_neighbors: 5, p: 2, weights: ‘uniform’.

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	25947	2077	3491
	Negativo	7148	15356	9376
	Neutro	10231	8821	12627

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,60	0,82	0,69	0,57
negativo	0,58	0,48	0,53	
neutro	0,50	0,40	0,44	

TABELA XV

RESULTADOS NAIVE BAYES CV BINÁRIO

Técnica	**Hiperparâmetros**
Naive Bayes	padrão

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	29142	2416
	Negativo	1951	29969

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,94	0,92	0,93	0,93
negativo	0,93	0,94	0,93	

TABELA XVI

RESULTADOS NAIVE BAYES CV MULTICLASSE

Técnica	**Hiperparâmetros**
Naive Bayes	padrão

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	27716	1041	2758
	Negativo	1119	22340	8421
	Neutro	6706	7706	17267

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,78	0,88	0,83	0,71
negativo	0,72	0,70	0,71	
neutro	0,61	0,55	0,57	

TABELA XVII

RESULTADOS NAIVE BAYES TF-IDF BINÁRIO

Técnica	**Hiperparâmetros**
Naive Bayes	padrão

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	28253	3305
	Negativo	1364	30556

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,95	0,90	0,92	0,93
negativo	0,90	0,96	0,93	

TABELA XVIII

RESULTADOS NAIVE BAYES TF-IDF MULTICLASSE

Técnica	**Hiperparâmetros**
Naive Bayes	padrão

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	26567	1400	3548
	Negativo	913	24012	6955
	Neutro	5634	9334	16711

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,80	0,84	0,82	0,71
negativo	0,69	0,75	0,72	
neutro	0,61	0,53	0,57	

TABELA XIX

RESULTADOS XGBOOST CV BINÁRIO

Técnica	**Hiperparâmetros**
XGBoost	colsample_bytree: 0.7, gamma: 0.2, learning_rate: 0.2, max_depth: 6, n_estimators: 1100, objective: 'binary:logistic', reg_alpha: 0.05, subsample: 0.7

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	29675	1883
	Negativo	2053	29867

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,94	0,94	0,94	0,94
negativo	0,94	0,94	0,94	

TABELA XXI

RESULTADOS XGBOOST TF-IDF BINÁRIO

Técnica	**Hiperparâmetros**
XGBoost	colsample_bytree: 0.7, gamma: 0.2, learning_rate: 0.2, max_depth: 6, n_estimators: 1000, objective: 'binary:logistic', reg_alpha: 0.05, subsample: 0.7

Matriz de Confusão

		Preditos	
		Positivo	Negativo
Conhecidos	Positivo	29574	1984
	Negativo	1937	29983

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,94	0,94	0,94	0,94
negativo	0,94	0,94	0,94	

TABELA XX

RESULTADOS XGBOOST CV MULTICLASSE

Técnica	**Hiperparâmetros**
XGBoost	colsample_bytree: 0.7, gamma: 0.2, learning_rate: 0.2, max_depth: 6, n_estimators: 1100, objective: 'multi:softmax', reg_alpha: 0.05, subsample: 0.7

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	27767	695	3053
	Negativo	1079	23670	7131
	Neutro	6182	7710	17787

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,79	0,88	0,83	0,73
negativo	0,74	0,74	0,74	
neutro	0,64	0,56	0,60	

TABELA XXII

RESULTADOS XGBOOST TF-IDF MULTICLASSE

Técnica	**Hiperparâmetros**
XGBoost	colsample_bytree: 0.7, gamma: 0.2, learning_rate: 0.2, max_depth: 6, n_estimators: 1000, objective: 'multi:softmax', reg_alpha: 0.05, subsample: 0.7

Matriz de Confusão

		Preditos		
		Positivo	Negativo	Neutro
Conhecidos	Positivo	27295	724	3496
	Negativo	857	23753	7270
	Neutro	5653	7863	18163

Relatório

Classe	**Precisão**	**Recall**	**F1-score**	**Acurácia**
positivo	0,81	0,87	0,84	0,73
negativo	0,73	0,75	0,74	
neutro	0,63	0,57	0,60	