Processamento de Linguagem Natural como Filtro para Analisar a Satisfação do Cliente

Cecília Capurucho*
cecibou@sga.pucminas.br
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Minas Gerais

Danielle Dias Vieira*
ddvieira@sga.pucminas.br
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Minas Gerais

Felipe Vilas Boas*
fboas@sga.pucminas.br
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Minas Gerais

João Augusto dos Santos Silva*
joao.silva.452811@sga.pucminas.br
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Minas Gerais

Thiago de Campos Ribeiro Nolasco*
tcrnolasco@sga.pucminas.br
Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte, Minas Gerais

ABSTRACT

Neste trabalho será abordado o Processamento de Linguagem Natural (PLN) com os algoritmos *RoBERTa, Naive Bayes Multinomial* e *Random Forest* para que seja feita uma análise se os comentários feitos pelos clientes da Amazon[1] condiz com as notas dadas ao produto pelo mesmo cliente. Assim é possível avaliar esta funcionalidade de nota e comentário, podendo utilizá-la como embasamento para melhorar os seus serviços. Os resultados mostraram que o PLN possui um excelente desempenho, mas ainda tem muito a melhorar, tendo em vista a complexidade das linguagens naturais. Além disso, elas têm grande variação do sentido das frases de acordo com o posicionamento das palavras, onde palavras que são consideradas negativas para os algoritmos estão em um contexto positivo e vice-versa.

KEYWORDS

machine learning, datasets, data science, tree, random forest, natural language processing, naive bayes multinomial, nltk, pre-processing

ACM Reference Format:
Cecília Capurucho, Danielle Dias Vieira, Felipe Vilas Boas, João Augusto dos Santos Silva, and Thiago de Campos Ribeiro Nolasco. 2022. Processamento de Linguagem Natural como Filtro para Analisar a Satisfação do Cliente. In *Trabalho Prático, Dezembro 04, 2022.* ACM, Belo Horizonte, MG, BR, 7 pages. https://doi.org/0

[1]https://amazon.com/

, ,

ACM ISBN 0…$00.00
https://doi.org/0

1 INTRODUÇÃO

O mercado de pedidos de alimentos *online* vem crescendo mais a cada dia, principalmente após a pandemia do covid-19. Com o fechamento de bares, restaurantes e outros estabelecimentos, para evitar a aglomeração, pedir comida *online* tornou-se a saída para muitas pessoas que antes compravam presencialmente e uma alternativa para continuação da operação das pequenas empresas deste setor. Mas antes da pandemia este mercado já era valorizado pelos clientes por ter uma comodidade de não ter que sair de casa para comprar.

As empresas que fornecem este serviço geralmente tem em seus *softwares* um espaço para o cliente deixar um comentário e nota sobre o pedido e a entrega realizada. Esta é uma das formas onde é possível analisar se o serviço está sendo bem realizado e se os clientes estão satisfeitos.

Uma das empresas de tecnologia que forcene o serviço de entrega, também conhecido como *delivery* é a Amazon. Além de vender produtos eletrônicos, livros, moda, também oferece produtos alimentícios em suas plataformas.

Neste trabalho, nosso objetivo é analisar se o comentário feito pelo cliente condiz com a nota de avaliação que ele forneceu pelo seu pedido de comida na Amazon nas suas plataformas. Com o intuito de ajudar a validar esta funcionalidade nos softwares que mostra se o cliente está satisfeito ou não com o serviço prestado.

Para realizar esta análise é utilizado o Processamento de Linguagem Natural (PLN) nos comentários deixados pelos clientes sobre a entrega de seu pedido. Processamento de Linguagem Natural é uma tecnologia que permite que computadores entendam, interpretem e manipulem a linguagem humana. O PLN é importante porque ajuda a resolver a ambiguidade na linguagem e adiciona uma estrutura numérica útil aos dados para muitas aplicações *downstream*, como reconhecimento de fala ou análise de texto. Neste trabalho é

utilizado nos algoritmos para classificar o comentário como negativo, positivo ou neutro. Com esta classificação é possível realizar uma comparação com as notas que os clientes também forneceram sobre o pedido.

Os comentários classificados e as notas que serão analisadas foram retirados da base de dados [2], que contém 568.454 instâncias e 10 colunas com atributos de entrada, sem uma coluna rótulo.

Neste contexto, os algoritmos utilizados para a classificação dos textos nos comentários dos clientes foram o *RoBERTa*, *Naive Bayes Multinomial* e *Random Forest*. Para uma melhor compreensão da importância do pré-processamento antes da execução destes algoritmos, e como isso pode prejudicar os resultados e diminuir a confiança deles, primeiro é realizado uma análise da classificação dos textos sem o balanceamento das instâncias da base de dados e depois é realizado já com o balanceamento feito, com isso é possível comparar os dois resultados.

Este texto está organizado da seguinte forma: na segunda seção tem a descrição da base de dados selecionada; as etapas com toda a metologia de aplicação dos algoritmos e pré-processamentos aplicados está na terceira seção; na quarta seção é realizada a análise das notas de avaliação juntamente com a classificação dos comentários fornecidos pelos clientes; na quinta seção é exposto os resultados obtidos e as conclusões finais sobre este trabalho. Na sexta seção há o link para ter acesso aos códigos utilizados neste trabalho na linguagem de programação Python. Por último há as referências utilizadas neste trabalho.

2 BASE DE DADOS

Para a execução do trabalho, foi escolhida uma base de dados que apresenta comentários e avaliações realizadas por clientes da Amazon. A base foi montada a partir do trabalho [2], que busca enviar recomendações baseadas no gosto atual do usuário. Os dados abrangem o período de outubro de 1999 a outubro de 2012. A base de dados apresenta 568.454 instâncias com 10 colunas, mas apenas 2 colunas serão utilizadas, a *Score* e *Text*, que contém respectivamente a nota com a avaliação e o comentário do cliente, sobre o serviço prestado pela Amazon. Os dados do *dataset* incluem:

- 256.059 usuários
- 74.258 produtos
- 260 usuários com mais de 50 avaliações

A base de dados não tem uma coluna rótulo com as classificações de cada comentário como positivo, negativo ou neutro. A descrição de todos os atributos da base estão na Tabela 1.

RoBERTa

Assim como mencionado na introdução, o presente trabalho tem como objetivo principal realizar o Processamento de Linguagem Natural (PLN) na base de dados detalhada na subsessão anterior a fim de investigarmos se as notas apresentadas aos produtos na coluna *Score* é condizente com a classificação realizada pelos algoritmos utilizados ao decorrer do trabalho.

O algoritmo escolhido para a classificação inicial do texto é baseado na arquitetura de Redes Neurais *Transformers*, que inicialmente foi criada para PLN e visão computacional. O algoritmo utilizado para a classificação inicial da base de dados foi o *A Robustly Optimized BERT Pretraining Approach (RoBERTa)* [1], modelo baseado na otimização do algoritmo *BERT*, que foi lançado pelo Google em 2018. Ambos algoritmos utilizam modelos pré-treinados a fim de serem capazes de aplicar o PLN considerando o contexto das palavras em sua classificação, pois muitos modelos voltados para PLN apenas consideram as palavras, fazem a contagem de palavras positivas, negativas e neutras, a partir dessa contagem a frase é classificada, mas sabemos que dependendo do contexto, a frase pode ser negativa mesmo apresentando muitas palavras positivas.

Para executar o algoritmo *RoBERTa*, utilizamos um modelo pré-treinado com aproximadamente 58 milhões de Tweets, o que nos garante um modelo treinado com um número de instâncias muito maior do que conseguiríamos utilizar para treinar o modelo. Não foi necessário realizar nenhuma etapa de pré-processamento para a execução do algoritmo, foi feita apenas uma iteração por todas as linhas da base de dados e selecionado o texto presente nessa linha, depois dessa seleção, utilizamos a função *tokenizer* da biblioteca *transformers*. Após a execução dessas etapas, obtemos uma lista com 3 elementos com valores de 0 até 1, o elemento na posição 0 corresponde ao *score* da classe negativo, o elemento na posição 1 corresponde ao *score* da classe neutro, por fim o elemento na posição 3 corresponde ao *score* da classe positivo. Para retornar a classificação do texto, fazemos a comparação entre os valores obtidos no *score*, onde o maior *score* é a classificação do texto de entrada.

Com a execução do algoritmo foram obtidas as seguintes classificações:

- Positivo: 3.254 instâncias
- Neutro: 368 instâncias
- Negativo: 807 instâncias

Os resultados obtidos retornaram uma base visivelmente desbalanceada e o impacto desse desbalanceamento na execução dos demais algoritmos a serem executados no presente trabalho.

Table 1: Descrição dos atributos da base de dados escolhida

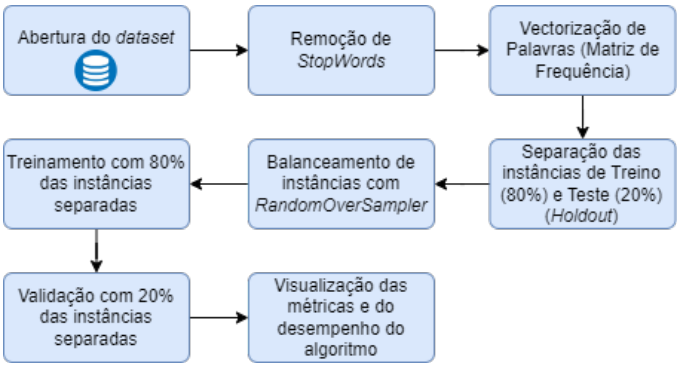
Atributo	Descrição do Atributo	Valores
Id	ID da instância na base de dados	Numérico: Min=1 Máx=568.454
ProductId	Identificador único do produto	Alfanumérico
UserId	Identificador único do usuário	Alfanumérico
ProfileName	Nome do perfil do usuário	Textual
HelpfulnessNumerator	Número de usuários que acharam a avaliação útil	Numérico: Min=0 Máx=866
HelpfulnessDenominator	Número de usuários que indicaram se acharam a avaliação útil ou não	Numérico: Min=0 Máx=923
Score	Nota do serviço	Numérico: Min=1 Máx=5
Time	Timestamp de quando foi feito o comentário	Categórico
Summary	Resumo do comentário sobre o serviço	Textual
Text	Comentário do serviço	Textual

3 ETAPAS

Nesta seção é exposta a metologia de como cada algoritmo foi aplicado e os pré-processamentos feitos para classificar os comentários dos clientes utilizando a Linguagem de Programação Python.

Na etapa 1 a classificação é feita com o algoritmo *Naive Bayes Multinomial*. Na etapa 2 é informado quais pré- processamentos foram aplicados e é utilizado novamente o algoritmo de *Naive Bayes Multinomial* com a base de dados balanceada, assim é possível comparar o desempenho da aplicação do algoritmo com a base balanceada e desbalanceada. Na etapa 3 a classificação é feita com o algoritmo *Random Forest* com a base de dados desbalanceada e balanceada. Na Figura 1 é possível visualizar a ordem dos pré-processamentos aplicados nos algoritmos *Naive Bayes Multinomial* e *Random Forest*.

Figure 1: Passo a passo das etapas aplicadas nos algoritmos *Naive Bayes Multinomial* e *Random Forest*



Naive Bayes Multinomial

O algoritmo *Naive Bayes Multinomial* é adequado para classificação com características discretas (por exemplo, contagem de palavras para classificação de texto). A distribuição multinomial normalmente requer contagens de recursos inteiros. No entanto, na prática, contagens fracionárias como tf-idf também podem funcionar.

A escolha desse algoritmo é uma boa escolha em base no artigo [4] que diz que com um pré-processamento adequado, o qual, consegue chegar a resultados iguais ou parecidos a de algoritmos mais complexos.

O que foi feito antes da parte de treinamento do *Naive Bayes Multinomial* foi a captura de palavras mais frequentes na classificação e a transformação de dados. Para isso, vale ressaltar que o Python aceita somente dados numéricos, como se está trabalhando com o tipo texto, nominal, foi-se utilizado a função *CountVectorizer* que de acordo com o site [3] ela converte uma coleção de documentos de texto em uma matriz de contagens de token. O funcionameto é que se tiver mais palavras que ele classifica como positivo, ele já deixa a entrada como positivo e assim em diante com o neutro e negativo.

As Figuras 2, 3 e 4 mostram os gráficos que comparam os resultados antes e depois do balanceamento das três classes, sendo a primeira a classe *positive*, a segunda *negative* e a última a classe *neutral*. Na Figura 2 quando a classe foi balanceada 2 métricas diminuíram, em relação a antes do balanceamento e *Precison* manteve o mesmo valor. Já na Figura 3 quando a classe foi balanceada as 3 métricas diminuíram, em relação a antes do balanceamento. Já na Figura 4 quando a classe foi balanceada a métrica *Precison* diminui, e as outras aumentaram em relação a antes do balanceamento.

Pré-processamento

Para os processos de pré-processamento na base de dados foi feita uma análise para verificar qual etapa de pré-processamento deveria ser realizada. A base não possui nenhum tipo de atributo ausente, mas está desbalanceada com as três possíveis

Figure 2: Métricas em relação a classe *positive*

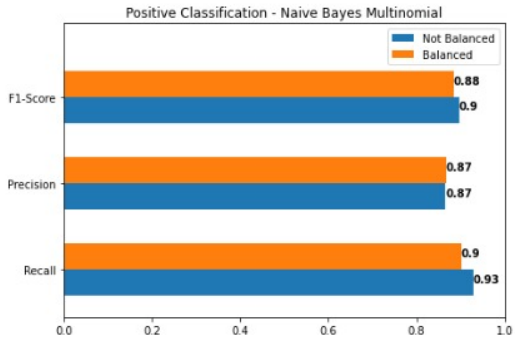


Figure 3: Métricas em relação a classe *negative*

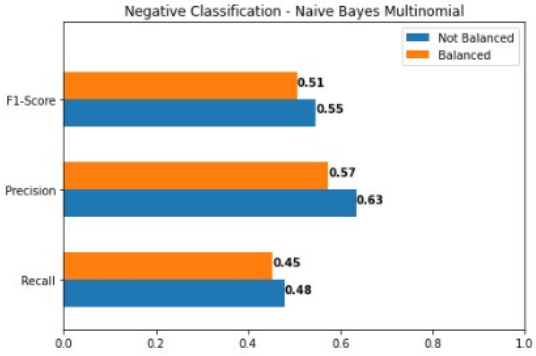
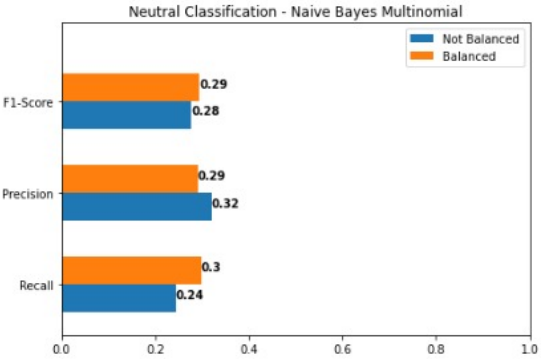


Figure 4: Métricas em relação a classe *neutral*



classes de classificação, o que causa problemas com a classificação das classes minoritárias, já que suas métricas estão baixas em relação a classe majoritária. Conforme explicado na seção *Naive Bayes Multinomial*, foi criada a matriz de frequência das palavras com a função *CountVectorizer*.

Assim, é possível resolver esse problema com o método de *oversampling*. Esse método é quando uma ou mais classes estão com menos instâncias e para balancear é gerado novas instâncias aleatórias. O *Random Over Sampler* da biblioteca *imbalanced-learn* é uma das maneiras de alcançar esse resultado para ele gerar aleatoriamente novas instâncias baseado nas já existentes. É importante ressaltar a ordem do balanceamento, a qual foi feita depois da separação de treino e teste com *Holdout*, na função *train_test_split*, e depois o balanceamento.

Portanto, com a análise dos resultados obtidos e comparando-os, é perceptível a diferença causada pelo balanceamento da base de dados para todas as classes conforme Figuras 2, 3 e 4. As métricas das classes negative e neutral apresentaram um aumento significativo, já que no primeiro teste elas eram as classes minoritárias do modelo.

Random Forest

Para complementar a análise e termos mais dados para comparação, foi utilizado o algoritmo *Random Forest*. O motivo da sua utilização se deve pelo fato de que, geralmente, o algoritmo apresenta um ótimo resultado para classificação da base de dados - dependendo da quantidade de árvores e do critério utilizado. Para uma análise mais fidedigna e correta, trabalharemos com os dados desbalanceados e balanceados, para então comparar os resultados encontrados.

Inicialmente, como em todos os algoritmos anteriores, foi selecionado a base de dados, retirada todas as *stopwords* das frases e, utilizando o *CountVectorizer* da biblioteca *feature extration* encontrada no *sklearn*, contou-se a frequência das palavras na frase para conseguir fazer a classificação, gerando assim uma matriz de frequências.

Após, foi utilizado o *Random Forest* - sem balancear - nas matrizes de frequências para classificar. Foram utilizadas 400 estimadores e o critério de cálculo sendo a Entropia das classes. Para o treino do algoritmo foi separado 80% das instâncias, e para o teste, 20%, utilizando assim o método de amostragem *Holdout*.

Ao analisar a quantidade total de instâncias de cada classe foi verificado uma discrepância na quantidade de cada. Para realizar a classificação com este algoritmo, foi se utilizado da base de dados 5.000 instâncias, sendo classificadas em 3857 como positivas, 778 como negativas e 365 como neutras. Para uma análise mais correta, foi feito um balanceamento destas instâncias utilizando a técnica de *oversampling*, fazendo com que todas as classes minoritárias se igualassem à quantidade de instância da classe majoritária. O *oversampling* foi aplicado nas instâncias de treinamento após divisão - a divisão seguiu a ordem de 80% para treinamento e 20% para teste, também com o método de amostragem *Holdout*.

A comparação dos resultados encontrados podem ser vistos nas Figuras 5, 6 e 7 e é perceptível que não houve muitas alterações nas métricas das classes. Nas classes classificadas como positivas - as majoritárias - os valores não alteraram, já na classificada como negativas, houve uma redução na

proporção de classes classificadas corretamente como negativas, e nas classificadas como neutras, houve uma redução na precisão da classificação. É visível que, para essa base de dados, a abordagem de balanceamento por *oversampling* não é a mais aconselhada por não haver melhora das métricas, no entanto, pode-se concluir que o algoritmo de *Random Forest* foi efetivo, tendo em vista que teve uma precisão alta na classificação das instâncias.

Figure 5: Métricas encontradas na classe *Positive*

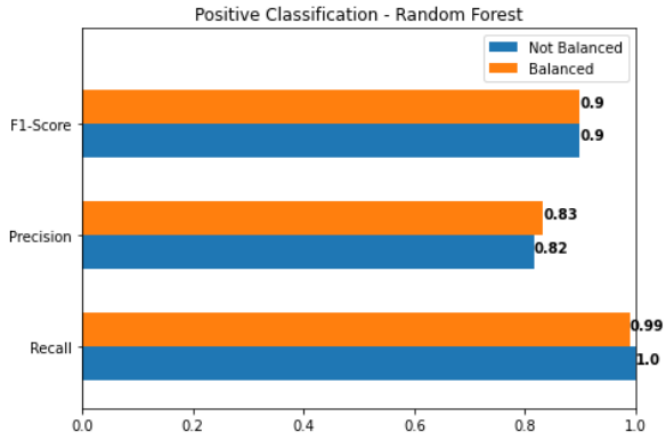
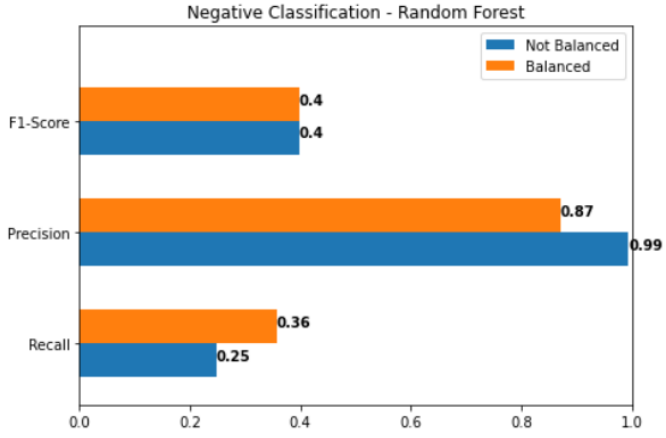


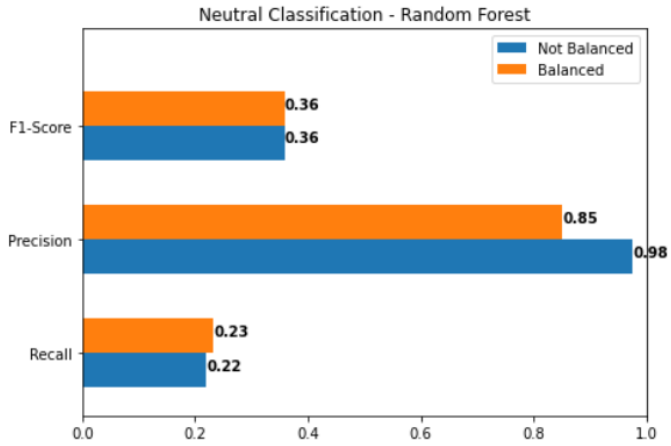
Figure 6: Métricas encontradas na classe *Negative*



4 ANÁLISE DAS NOTAS DE AVALIAÇÃO COM AS CLASSIFICAÇÕES DOS COMENTÁRIOS

Por meio dos gráficos nas Figuras 9, 10 e 11, é possível verificar a relação das notas de avaliação com as classificações dos comentários realizadas pelo algoritmo *Random Forest*, sobre os serviços e produtos vendidos na Amazon. Este algoritmo foi escolhido porque de acordo com a métrica *Precision*, ele foi o que obteve o melhor resultado nas 3 classes,

Figure 7: Métricas encontradas na classe *Neutral*



conforme Figura 8. *Precision* foi escolhido para selecionar o melhor algoritmo neste trabalho, pois os Falsos Positivos são considerados mais prejudiciais que os Falsos Negativos, ou seja, o modelo precisa ser mais preciso para a empresa poder analisar melhor os resultados e com base neles aplicar ações de melhorias nos serviços ou nas vendas.

As hipóteses antes da avaliação sobre as notas, foram que para a classificação positiva a nota estaria com valor acima de 4. Para a classificação negativa abaixo de 2 e para a neutra, seria entre 2 e 3. Porém, após visualizar os gráficos nas Figuras 9, 10 e 11, é possível perceber que na classe *Negative* há notas com valor 3, 4 e 5, na classe *Neutral* há notas 1, 4 e 5, e que na classe *Positive* há notas com valor 1, 2 e 3.

Analisando estes comentários que foram classificados como negativo e têm uma nota 4 ou 5, é possível perceber que alguns elogiam o produto utilizando palavras que são interpretadas pelo algoritmo como negativas e ruins, como por exemplo, "Cuidado, este produto é viciante de tão gostoso". Ou seja, apesar de ser um comentário positivo, as palavras "Cuidado" e "viciante" levam o algoritmo a errar a classificação por não compreender a frase como um todo.

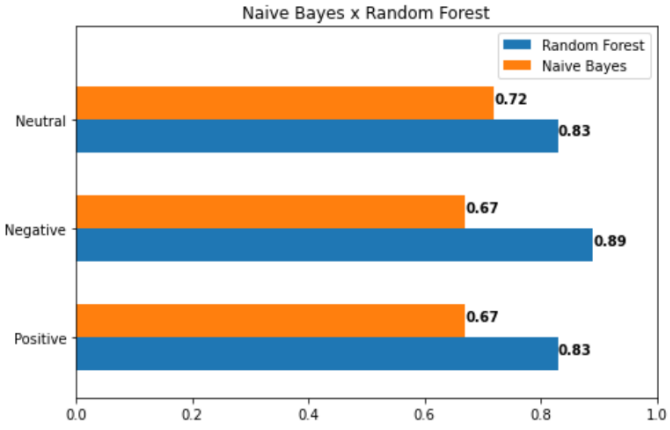
Muitos comentários classificados como negativo foram classificados corretamente, apesar de terem notas 4 ou 5. Motivos que os próprios clientes disseram sobre a nota ser alta, foi que gostaram da qualidade do produto, porém foi classificado como negativo porque davam sugestões sobre a entrega ser mais sustentável, que a taxa de entrega foi alta, ou que veio pouco produto no recipiente, ou que a embalagem não foi fácil de abrir.

Analisando os comentários classificados como *Positive* os clientes elogiaram a qualidade do produto, porém alguns deram notas baixas pelo motivo do preço ser alto, ou comparavam com um mesmo produto de outra marca.

Alguns comentários classificados como *Neutral* são clientes desabafando sobre o seu dia a dia, outros apenas comparando marcas de produto, ou com textos muito grandes onde a frequência de palavras positivas e negativas podem ser muito próximas, classificando assim como *Neutral*.

Portanto, como o algoritmo ainda não é perfeito, é possível analisar melhor sobre a satisfação do cliente utilizando a nota de avaliação juntamente com a classificação, ou seja, ao analisar as notas 4 e 5 juntamente com a classificação *Positive*, estas instâncias podem ser utilizadas pelo setor de compras como informação para quais produtos os clientes mais gostaram, para assim deixar um estoque maior. Ou estes produtos podem ser utilizados em campanhas de *marketing* para atrair mais vendas. Já nas notas 1 e 2 com a classificação *Negative* é possível avaliar o que pode melhorar em preços, embalagens, no serviço de entrega, ou até mesmo deixar de trabalhar com certa marca que não esteja agradando os clientes e trocar por outra.

Figure 8: Métrica *Precision* por classe nos algoritmos *Random Forest* e *Naive Bayes Multinomial*



5 RESULTADOS E CONCLUSÕES

Com a execução dos códigos e análise dos resultados obtidos, é possível concluir que os algoritmos existentes de PLN ainda devem melhorar para que se possa confiar em seus resultados.

Foi observado que a variação da posição das palavras e consequentemente a mudança do significado das mesmas dentro de um determinado contexto não causa nenhuma diferença no resultado retornado pelos algoritmos *Naive Bayes Multinomial e Random Forest*, que não são capazes de identificar expressões, ao contrário do que é proposto com o algoritmo *RoBERTa*, que por apresentar um conjunto pré-treinado de 58 milhões de instâncias, consegue fazer classificações mais precisas, mas ainda distantes do que era esperado pela dificuldade de se analizar um texto informal disponibilizado em plataformas *on-line*.

Figure 9: Relação das notas de avaliação com classe *Positive* pelo *Random Forest*

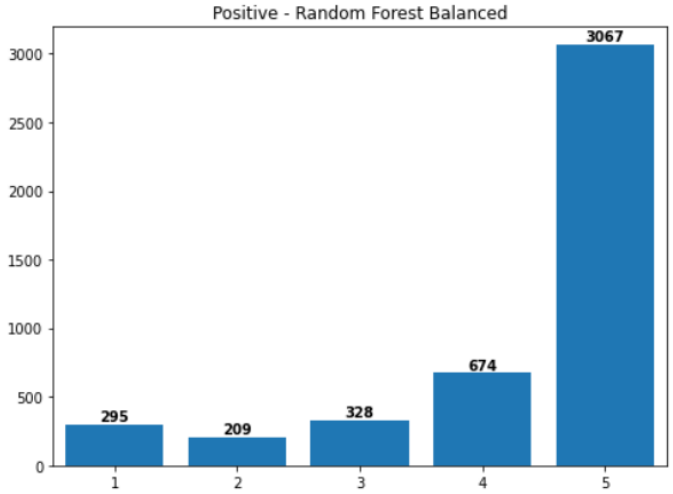
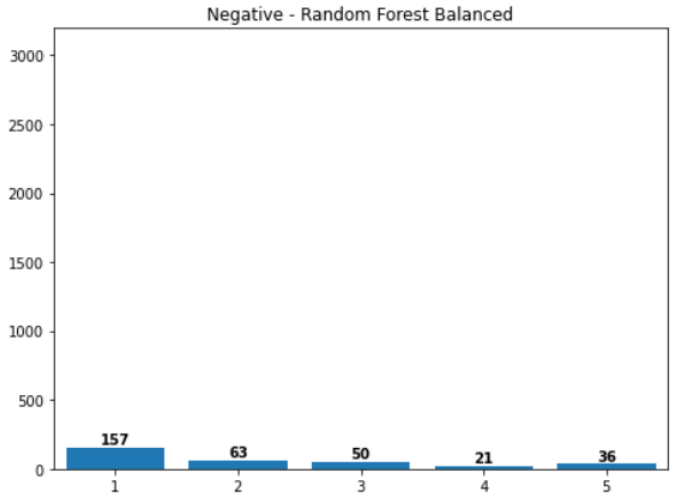


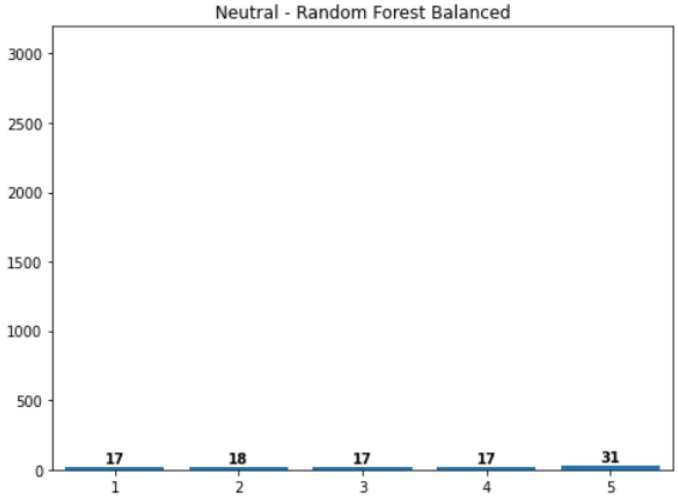
Figure 10: Relação das notas de avaliação com classe *Negative* pelo *Random Forest*



Porém as classificações juntamente com as notas de avaliação podem ser utilizadas como filtros para ajudar a empresa a melhorar os seus serviços. Utilizando, por exemplo, a classe *Positive* com as notas 4 e 5, a classe *Negative* com as notas 1 e 2.

Nos gráficos presentes nas imagens (9, 6,7), é apresentada a contagem de avaliações de acordo com a classe atribuída a cada uma das instâncias pelo modelo treinado com o *Random Forest*, que segundo as métricas analisadas, obteve um desempenho mais satisfatório em relação aos demais analisados. É possível observar que apesar do desbalanceamento presente na base de dados, a classificação positiva de comentários é

Figure 11: Relação das notas de avaliação com classe *Neutral* pelo *Random Forest*



representada por sua grande maioria por avaliações nota 4 e 5 (Figura 5), o que é esperado de um texto positivo. Os textos classificados como negativos (Figura 6), apresentam uma quantidade maior de avaliações nota 1, o que também é satisfatório, tendo em vista que é a nota mais baixa disponível para avaliação. Por fim, os comentários neutros (Figura 7) nos mostram que os textos classificados como neutro estão bem distribuídos entre todas as notas e é justamente onde o algoritmo pode ter ficado em dúvida por não compreender o contexto no qual as palavras foram empregadas.

Para trabalhos futuros, é desejável que novos algoritmos sejam explorados, assim como diferentes bases de dados de diferentes domínios, para que desta forma, possa ser encontrado um algoritmo de PLN que seja de fato eficiente e mais condizente com a realidade.

6 CÓDIGOS DESENVOLVIDOS

Todos os códigos e bases de dados utilizados no trabalho estão disponíveis no GitHub pelo seguinte link:

https://github.com/joaoaugustoss/TrabalhoPratico-IA

REFERENCES

[1] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[2] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*. 897–908.

[3] scikit-learn developers. 2007-2022. sklearn.CountVectorizer. Retrieved November 9, 2022 from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[4] Yan Li Shuo Xu and Wang Zheng. 2017. Bayesian Multinomial Naïve Bayes Classifier to Text Classification. *Lecture Notes in Electrical Engineering* (2017).