

## Relatório Trabalho Final de NLP

Professora: Bárbara Silveira

Aluno: Daniel Moysés Marques Dutra de Oliveira

Pós-Graduação: Ciência de Dados e Inteligência Artificial

PUC/MINAS

Neste trabalho foram desenvolvidos dois modelos de classificação de sentimentos usando o Dataset disponibilizado pela plataforma Olist.com na comunidade kaggle.com, com comentários e avaliações feitos na plataforma pelos próprios usuários consumidores.

O Dataset é composto por várias tabelas, mas o foco desse trabalho será na tabela “olist\_order\_reviews” , composta por 100 mil registros em colunas contendo os comentários, os títulos dos comentários, as avaliações, datas de criação, datas de resposta e duas colunas de ID, “review\_id” e “order\_id”.


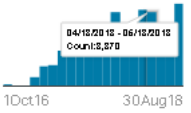
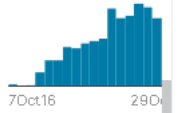
Detail	Compact	Column	7 of 7 columns			
review_id	order_id	review_score	review_comment...	review_comment...	review_creation...	review_answer_ti...
unique review identifier	unique order identifier	Note ranging from 1 to 5 given by the customer on a satisfaction survey.	Comment title from the review left by the customer, in Portuguese.	Comment message from the review left by the customer, in Portuguese.	Shows the date in which the satisfaction survey was sent to the customer.	Shows satisfaction survey answer timestamp.
98410 unique values	98673 unique values		[null] 88% Recomendo 0% Other (11145) 11%	[null] 59% Muito bom 0% Other (40747) 41%		
9a8abbb668bafb95a6d2b85db43284c4	d7bd0e4afdf94846eb73642b4e3e75c3	3			2017-04-30 00:00:00	2017-05-03 00:02
3948b09f7c818e2d86c9a546758b2335	e51478e7e277a83743b6f9991dbfa3fb	5	Super recomendo	Vendedor confiável, produto ok e entrega antes do prazo.	2018-05-23 00:00:00	2018-05-24 03:00
9314d6f9799f5bfba518cc7bcd468c01	8dacf04c5ad59fd5a8cc1faa07c34e39	2		GOSTARIA DE SABER O QUE HOUE, SEMPRE RECEBI E ESSA COMPRA AGORA ME DECPCIONOU	2018-01-18 00:00:00	2018-01-20 21:25
8e15a274d95608fa14f8be64e37a0e67	ff1581e08b3011021e7c7de592ddc81e	5			2018-03-24 00:00:00	2018-03-26 15:58
fdbdb2629a7cde0f66657acc92084e7f	70a752414a13d09cc1f2b437b914b28e	3			2017-09-29 00:00:00	2017-10-02 01:12
373cbeecea8286a2b66c97b1b157ec46	583174fbe37d3d5f0d6661be3aad1786	1	Não chegou meu produto	Péssimo	2018-08-15 00:00:00	2018-08-15 04:10

Figura 1: Tabela olist\_order\_reviews

Link Dataset: <https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce>

## Bibliotecas usadas

As bibliotecas que foram usadas são as seguintes:

- Pandas – Biblioteca para manipulação e tratamento de DataFrames
- Regexp(re) – Biblioteca para tratamento de strings
- String – Biblioteca para manipulação e tratamento de strings
- NLTK(Natural Language Tool Kit) – Biblioteca ou tool kit para processamento de Linguagem Natural
- Matplotlib Pyplot– Plotagem de gráficos
- Train\_Test\_Spli(SKLearn) – Módulo SKLearn para dividir DataFrame em treino e teste
- Logistic\_Regression(SKLearn) – Regressão logística do SKLearn
- XGBClassifier(XGBoost) – Classificador Gradient Boost da biblioteca XGBoost
- Accuracy\_score(SKLearn) – Módulo do SKLearn para cálculo de acurácia de modelos
- Confusion\_matrix(SKLearn) – Módulo do SKLearn para gerar matriz de confusão para avaliação dos modelos
- CountVectorizer(SKLearn) – Módulo para vetorização de textos do SKLearn
- Plot\_confusion\_matrix(SKLearn) – Módulo para visualização de Matriz de Confusão

## Objetivo

O objetivo deste trabalho é usar a coluna “review\_score” e a coluna “review\_comment\_message” para treinar os modelos de classificação e em seguida comparar as classificações feitas com as classificações geradas pelo LeIA(Léxico para Interferência Adaptada) que é uma ferramenta baseada no VADER(Valence Aware Dictionary and sEntiment Reasoner) para análise de sentimentos.

GITHUB LeIA: <https://github.com/rafjaa/LeIA>

## Execução

### 1. Carregamento do Dataset e seleção das Colunas

O Dataset foi carregado usando o método read\_csv() da biblioteca pandas e uma vez que os dados foram carregados no DataFrame pandas, foi criado um novo DataFrame somente com as colunas “review\_score” e “review\_comment\_message”.

### 2. Criação da Coluna “Sentimento”

Foi criada uma nova coluna contendo a classificação de cada comentário baseando-se na nota dada na coluna “review\_score”, onde notas 4 e 5 serão associadas ao sentimento positivo, notas 3 serão associadas ao sentimento neutro e notas 1 e 2 serão associadas ao sentimento negativo. Sendo assim a nova coluna possui os valores “positivo”, “negativo” e “neutro” e os modelos foram treinados usando essa coluna como target.

### 3. Limpeza dos Dados e criação de novo DataFrame

A coluna “review\_comment\_message” apresenta muitos registros faltantes e após a análise dos NaNs foi possível extrair 41753 comentários. Após a limpeza de NaNs foi criado novo DataFrame contendo somente as colunas “review\_comment\_message” e “sentimento”. Esse DataFrame foi usado no treinamento dos modelos.

#### 4. Análise Exploratória dos Dados Limpos e Visualizações

A visualização e exploração dos dados demonstrou que temos um número de sentimentos positivos bem maior que o número de sentimentos negativos e neutros na seguinte proporção:

63,89% positivo

27,32% negativo

8,77% neutro

Cada comentário tem em média 12 palavras por comentário antes da tokenização e remoção de StopWords.

#### 5. Tokenização, Remoção de StopWords e Remoção de Pontuação

Nesta parte do trabalho foi feita a tokenização, remoção de StopWords e Pontuação em todos os comentários da coluna "review\_comment\_message" e inseridos em uma nova lista de comentários já tratados.

Após a tokenização, remoção de StopWords e Pontuação a média de palavras por comentário caiu para 7,17 palavras por comentário.

#### 6. Vetorização dos Comentários

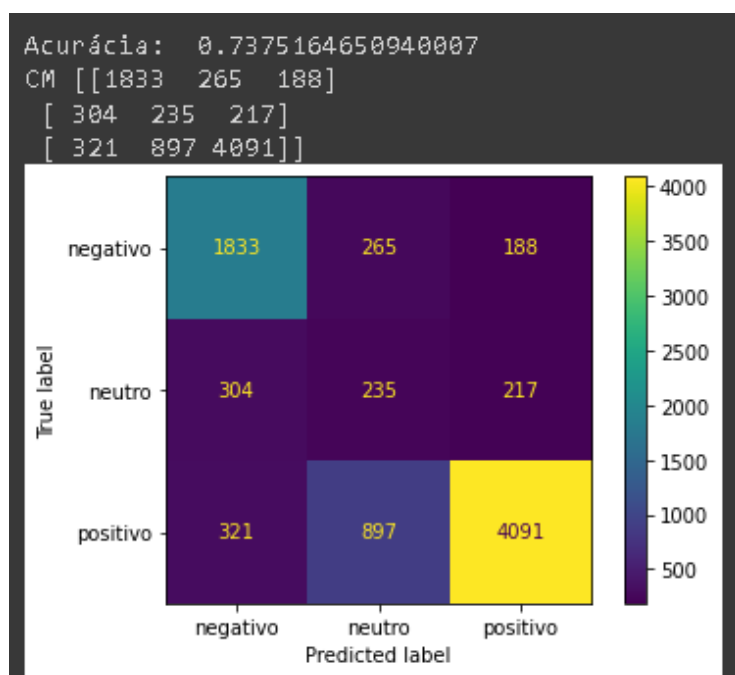
Inicialmente foi tentado usar o CountVectorizer() com o parâmetro n-grams setado para 1, 2 e 3, porém devido a extensão do DataFrame o GoogleColab não teve memória suficiente para executar vetorização com mais de 1 n-gram, então o CountVectorizer() foi usado setado com os parâmetros default.

#### 7. Separação do Dataset em Treino e Teste e Data Augmentation SMOTE

O treinamento dos modelos depende da separação dos dados em dados de treinamento e dados de teste, onde foram separados 20% dos dados para testes após o treinamento dos modelos. Como temos uma desproporção do número de registros para cada classe de predição, foi executado um SMOTE fazendo um Data augmentation e deixando todas as classes com 21371 registros para cada classe.

#### 8. Modelo de Regressão Logística e avaliação do modelo

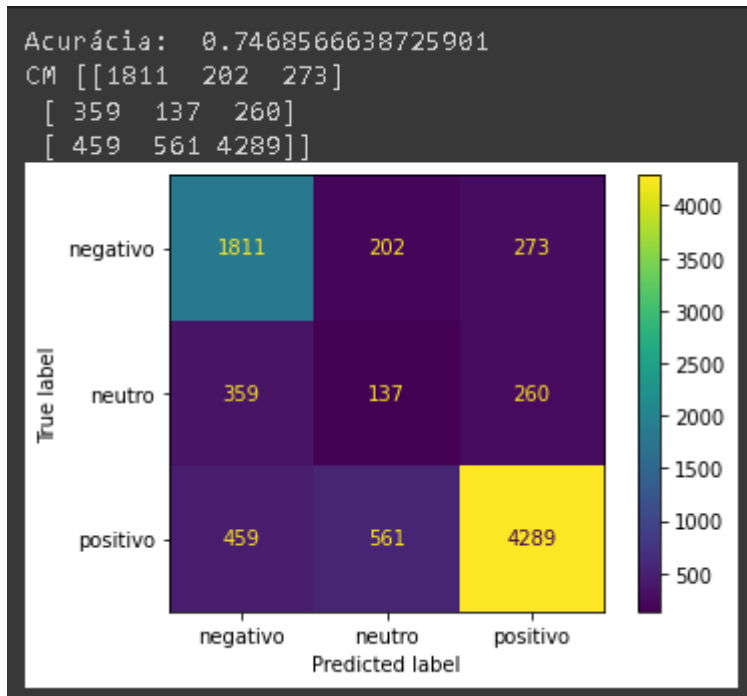
O primeiro modelo a ser treinado é o modelo de regressão logística com os seguintes parâmetros: max\_iter=500, solver='sag', warm\_start=True



## 9. Modelo XGBClassifier e avaliação do modelo

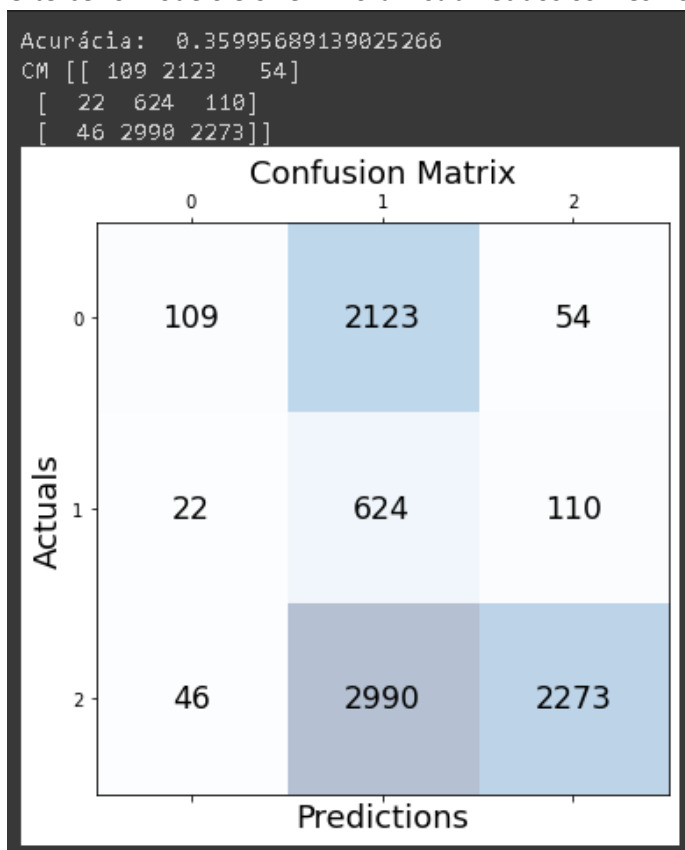
O segundo modelo treinado foi o modelo XGBoost Classifier com os seguintes parâmetros:

n\_estimators = 500, max\_depth = 10, learning\_rate = 0.01, subsample = 1, random\_state=123



## 10. LeIA e avaliação LeIA

O terceiro modelo é o LeIA. Foram submetidos os mesmo 20% dos dados separados ao modelo LeIA.



## 11. RandomSearch com XGBClassifier

Foi executado um RandomSearchCV() sobre o modeloXGBClassifier, afim de melhorar a acurácia do modelo com os seguintes parâmetros:

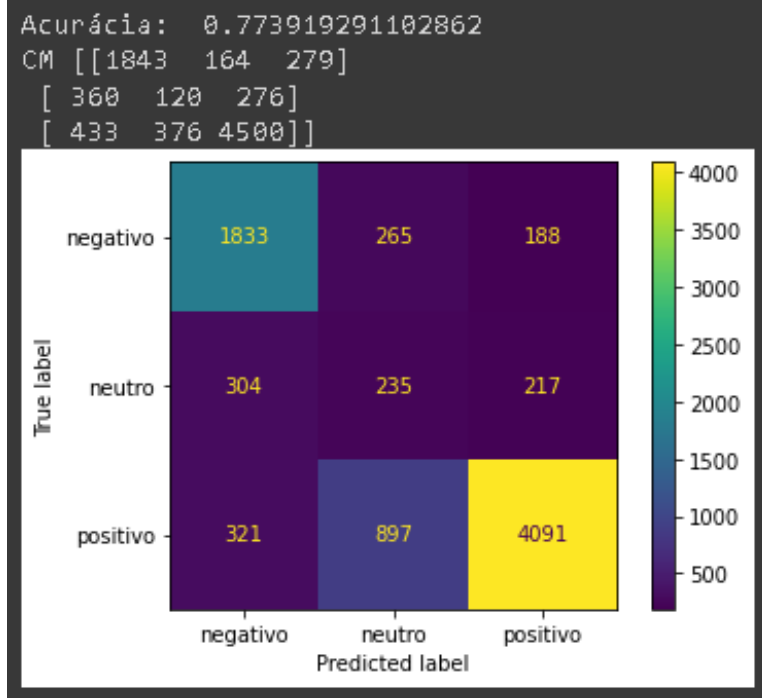
```
params = {
    'min_child_weight': [1, 5, 10],
    'gamma': [0.5, 1, 1.5, 2, 5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'max_depth': [7,8,9,10,11,12,13],
    'n_estimators': [500,550,600,650]
}
folds = 3
param_comb = 5

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb_model, param_distributions=params, n_iter=param_comb, scoring='roc_auc', n_jobs=4, cv=skf.split(vetor,novodf['sentimento']), verbose=3, random_state=1001 )
```

Best estimator:

XGBClassifier(colsample\_bytree=1.0, gamma=1, learning\_rate=0.01, max\_depth=11,  
n\_estimators=650, objective='multi:softprob', random\_state=123,  
subsample=0.6)



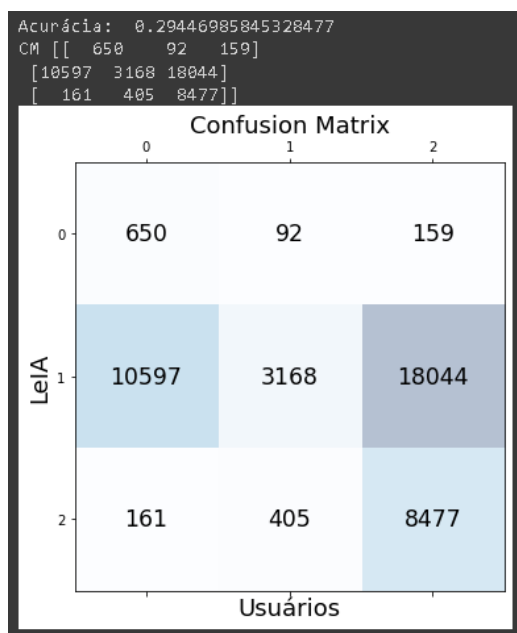
## 12. Comparação das Classificações do LeIA com as classificações postadas pelos Usuários

Afim de determinar as diferenças entre as notas dadas pelos usuários e as classificações do LeIA, todo o Dataset foi submetido ao LeIA e os resultados foram armazenados em nova coluna que foi chamada de “sentimento\_leia”.

	review_comment_message	sentimento	sentimento_leia
3	Recebi bem antes do prazo estipulado.	positivo	neutro
4	Parabéns lojas lannister adorei comprar pela l...	positivo	positivo
9	aparelho eficiente. no site a marca do aparelh...	positivo	neutro
12	Mas um pouco ,travando...pelo valor ta Boa.\n\n	positivo	positivo
15	Vendedor confiável, produto ok e entrega antes...	positivo	neutro
...	...	...	...
99983	Entregou dentro do prazo. O produto chegou em ...	positivo	neutro
99990	O produto não foi enviado com NF, não existe v...	neutro	neutro
99996	Excelente mochila, entrega super rápida. Super...	positivo	positivo
99998	Solicitei a compra de uma capa de retrovisor c...	negativo	neutro
99999	meu produto chegou e ja tenho que devolver, po...	negativo	neutro

41753 rows x 3 columns

Buscando entender melhor essas diferenças foi gerada uma matriz de confusão, comparando a coluna “sentimento\_leia”, que possui as classificações do LeIA, com a coluna “sentimento” que possui as classificações da regra citada acima (notas 1 e 2 = negativo, notas 3 = neutro, notas 4 e 5 = positivo)



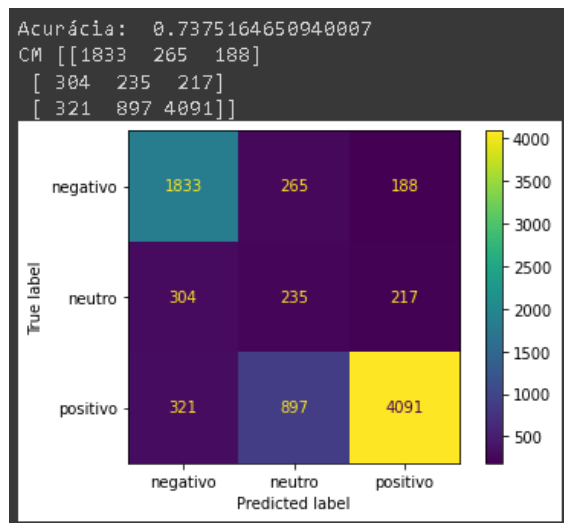
## Conclusões

Após a execução de todos processos, desde a análise dos dados, passando pela limpeza, tokenização, vetorização e treinamento dos modelos, a avaliação dos modelos nos trouxe os seguintes resultados de acurácia de cada modelo:

### Regressão Logística

Acurácia: 73,75%

Analisando a matriz de confusão do modelo de regressão logística é possível notar que o modelo erra principalmente classificando como neutros comentários que originalmente eram marcados como positivos. O modelo também classificou comentários neutros como negativos e positivos, errando a maioria dos neutros. Os negativos e positivos em sua maioria foram classificados corretamente.

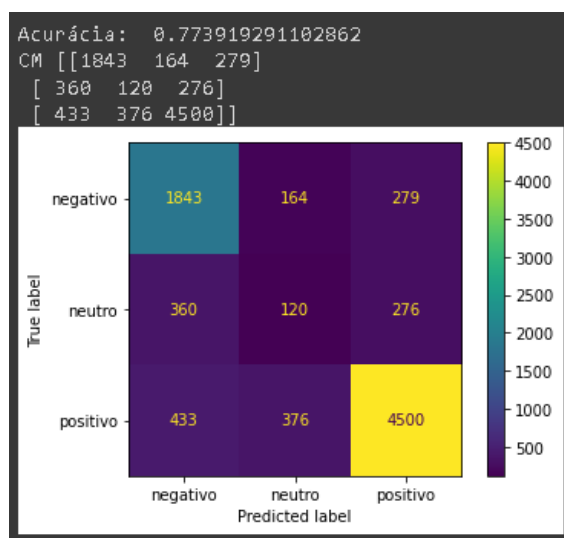
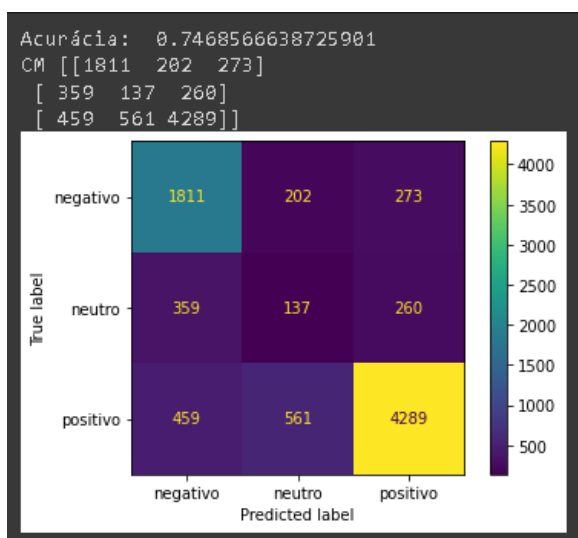


### XGB Classifier

Acurácia pré RandomSearchCV: 74,68%

Acurácia pós RandomSearchCV: 77,39%

A matriz de confusão do XGBClassifier ficou bem próxima da matriz do modelo de regressão logística, porém com mais acertos em positivos e negativos e mais erros na classificação como neutros. Após a execução do RandomSearchCV() e o ajustes dos parâmetros, foi obtido uma melhora de 3% na acurácia, e o modelo acertou mais positivos e negativos mas diminuiu o acerto em neutros.



## LeIA (VADER)

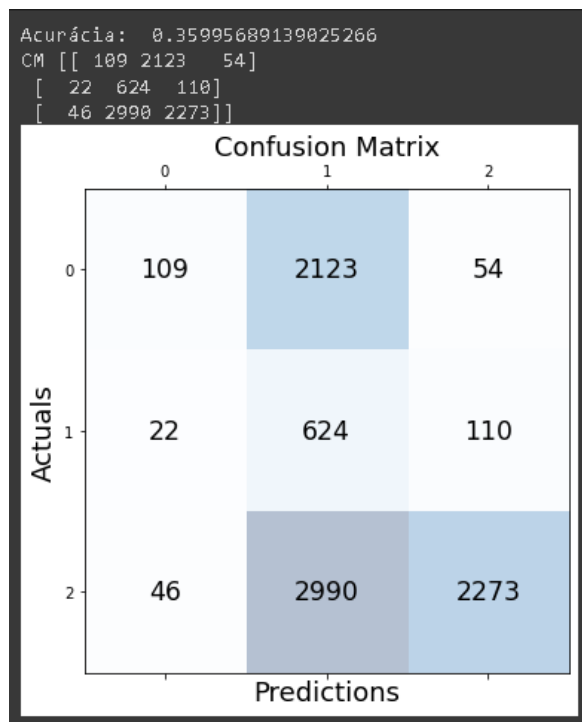
Acurácia: 35,99%

A matriz de confusão do LeIA demonstra que o modelo acerta muito mais os comentários neutros do que os modelos prévios, porém erra consideravelmente os negativos e positivos, associando-os ao neutro.

Analisando algumas sentenças dos comentários que foram classificadas de forma diferente entre os modelos, notou-se que os comentários classificados pelo LeIA como neutros e classificados pelos outros modelos como positivos, poderiam ser classificados por uma pessoa, analisando os comentários por si mesma, como um comentário realmente neutro. O mesmo acontece com comentários classificados negativos pelos outros modelos mas que foram classificados como neutros pelo LeIA.

A diferença na classificação desses modelos com o LeIA pode ser atribuída a alguns fatores, sendo o principal fator o fato dos próprios usuários terem dado as notas junto com as avaliações e essas notas terem sido usadas como referência de positivo, negativo ou neutro no início deste trabalho. Desta forma é possível que um usuário tenha postado um comentário com um texto neutro porém tenha dado uma nota alta ou baixa, fazendo com que os modelos de regressão logística e o XGBClassifier aprendessem que comentários que na verdade são neutros pudessem ser classificados como positivos ou negativos.

O LeIA nos traz seu próprio algoritmo já treinado, por assim dizer, capaz de fazer essas classificações de forma mais generalizada e eficaz. Levando isso em consideração é provável que o LeIA tenha na verdade nos trazido as reais classificações desses comentários, mesmo apresentando uma acurácia de 35,99% , pois como citado anteriormente os comentários tiveram suas classificações geradas pelas notas que os acompanhavam na regra já apresentada acima onde notas 1 e 2 são negativos, 3 é neutro e notas 4 e 5 são positivos . Na realidade o Dataset traz comentários neutros, ou que poderiam ser facilmente classificados como neutros, com boas notas de avaliação.

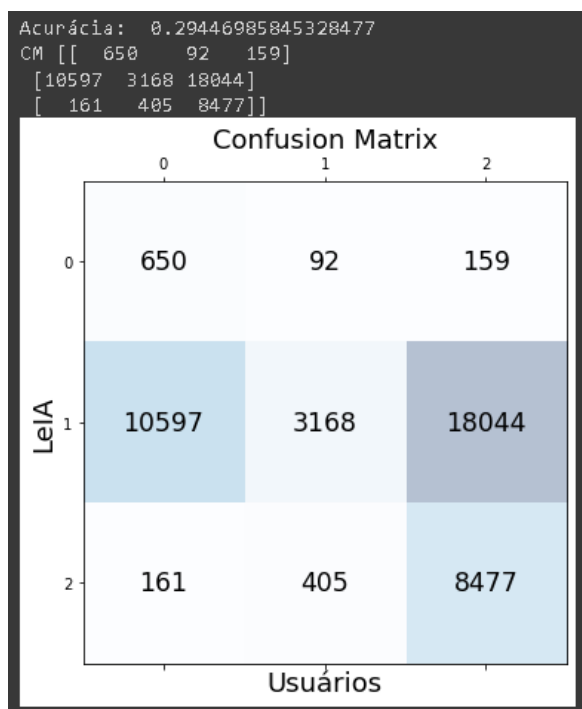


## LeIA vs Usuários

A matriz de confusão apresentando as classificações do LeIA em relação as classificações dos usuários mostra uma diferença no próprio conceito do que é um comentário positivo, negativo ou neutro, uma vez que diversos usuários escreveram comentários que transmitem neutralidade mas deram nota 4 ou 5, ou deram nota 1 ou 2. As classificações do LeIA e dos usuários são iguais em somente 29,44% das vezes.

Seria possível alterar as regras iniciais de classificação e considerar notas 4 e notas 2 como neutros, tornando comentários positivos somente aqueles com nota 5 e comentários negativos somente aqueles com nota 1, o que poderia levar a uma maior concordância entre os modelos.

Sendo assim, pode-se argumentar que os modelos de regressão logística e XGBClassifier estão na verdade classificandoos comentários no contexto e critério dos usuários e não nos critérios mais gerais de neutralidade, positividade e negatividade trazidos pelo LeIA(VADER)





Levando em consideração as acurácias de cada modelo comparando-os com o LeIA(VADER) e as diferenças nas classificações originais comparadas com as classificações do LeIA, e fazendo uma análise visual dos comentários que foram classificados de forma diferente entre os modelos, é mais provável que o LeIA(VADER) seja o mais indicado na resolução do problema de análise de sentimento, uma vez que o LeIA se mostrou mais realista em suas classificações.

A alteração da regra de que notas 1 e 2 são associadas a sentimentos negativos, notas 3 a neutros e notas 4 e 5 a sentimentos positivos, para uma regra onde somente notas 1 fossem associadas a negativo e somente notas 5 fossem associadas a positivo, provavelmente resultaria em maior concordância de classificação entre o LeIA e os outros dois modelos de classificação treinados.

## Tarefas futuras

Podemos incluir como tarefas futuras a execução de um GridSearch() no modelo XGBClassifier abrangendo um intervalo maior no parâmetro n\_estimator e max\_depth e também refazer todos treinamento aplicando a nova regra onde somente notas 1 serão associadas a sentimentos negativos e somente notas 5 serão associadas a sentimentos positivos.

- GridSearch() – setar parâmetros “n\_estimator” e “max\_depth” para intervalos maiores, mais abrangentes
- Aplicar nova regra (notas 1 serão associadas a sentimentos negativos e notas 5 serão associadas a sentimentos positivos)
- Refazer todo o treinamento incluindo RandomSearchCV() e GridSearchCV()

---

## Instruções de Instalação do LeIA(VADER)

Quando o LeIA foi instalado no Google Colab o arquivo LeIA.py e os lexicons vieram faltantes ou incorretos. Por isso estou enviando os arquivos baixados diretamente do GITHUB do LeIA.

Após o Comando “!pip install leia” for executado, a lib do LeIA apresentará um erro, não encontrando o módulo necessário para análise (SentimentIntensityAnalyzer) e quando chequei o arquivo leia.py no Google Colab ele estava sendo instalado vazio.

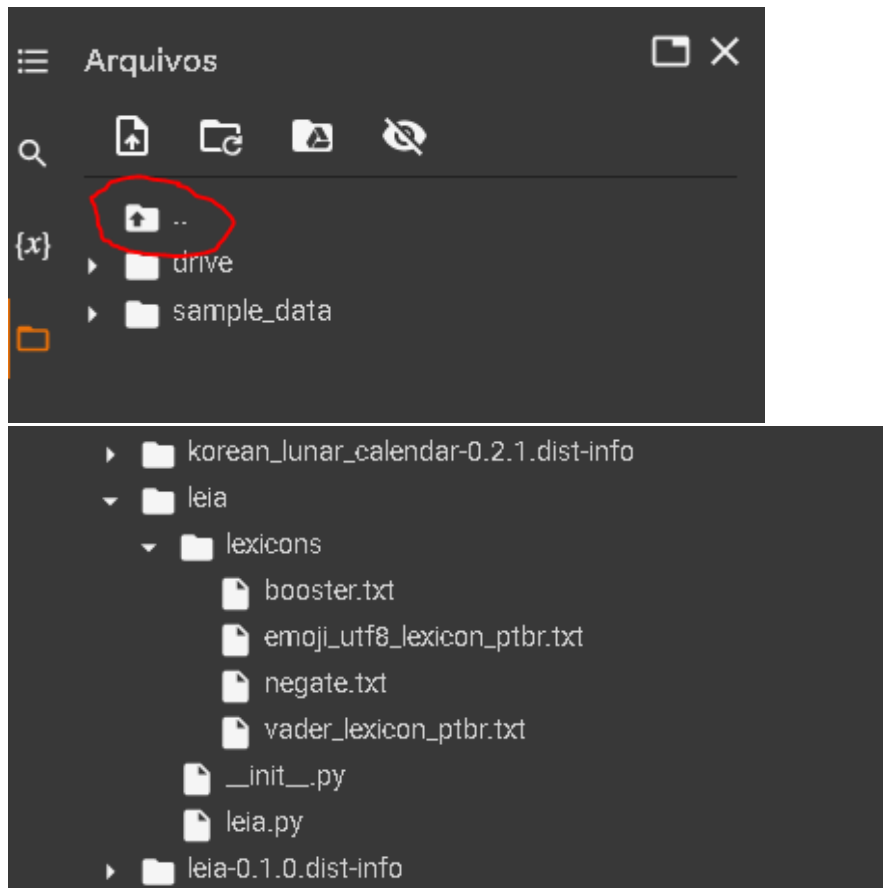
Para o método de análise de sentimento do LeIA (SentimentIntensityAnalyzer) ser carregado foi necessário substituir o arquivo leia.py pelo arquivo leia.py do GIT original.

GITHUB LeIA: <https://github.com/rafjaa/LeIA>

Para isso acesse a pasta do colab:

`/usr/local/lib/python3.7/dist-packages/leia`

1. Substitua o arquivo leia.py (está vazio) pelo arquivo leia.py enviado (original git)
2. Crie uma pasta chamada "lexicons"
3. Faça o upload dos arquivos léxicos em português para dentro da pasta "lexicons"
4. Reinicie o ambiente de execução



Qualquer dúvida na instalação do LeIA é só enviar e-mail para [danielsolo@gmail.com](mailto:danielsolo@gmail.com)

P.S.: O Colab tem dado erro de vez em quando na instalação do LeIA