

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Thiago Berberich Cabral

**ANÁLISE DE SENTIMENTO DE COMENTÁRIOS SOBRE MARCAS DE
CONSUMO COM MACHINE LEARNING**

Belo Horizonte

2023

Thiago Berberich Cabral

**ANÁLISE DE SENTIMENTO DE COMENTÁRIOS SOBRE MARCAS DE
CONSUMO COM MACHINE LEARNING**

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte

2023

SUMÁRIO

1. Introdução.....	4
1.1. Contextualização	4
1.2. O problema proposto.....	5
2. Coleta de dados	10
3. Processamento/Tratamento de Dados	15
4. Análise e Exploração dos Dados	19
5. Criação de Modelos de Machine Learning	24
6. Interpretação e apresentação dos Resultados	38
7. Links.....	42
REFERÊNCIAS.....	43
APÊNDICE.....	44

1. Introdução

1.1. Contextualização

A análise de sentimentos é uma estratégia que tem como base a avaliação e classificação das menções sobre seus temas de interesse. Ao categorizar as menções em positivas, negativas ou neutras é possível identificar pontos fracos, tomar decisões empresariais mais efetivas e solucionar cenários negativos.

Outro motivo para analisar o sentimento das menções é melhorar o relacionamento com o público e entender se a sua organização está atendendo às expectativas dos consumidores.

Elas permitem que as marcas como exemplo (Dell, Apple, LG e etc.) criem relacionamentos com seus consumidores, aumentem a visibilidade de seus produtos e serviços e coletem informações importantes do público. Baseiam-se em regras e usam algoritmos (ou regras) escritos manualmente para avaliar a linguagem. Essas regras usam métodos de linguística computacional como tokenização, lematização, stemização e marcação de partes do discurso. Elas também podem usar léxicos (bancos de palavras).

Essa conectividade global possibilita amizades e colaborações que antes eram impensáveis, promovendo a compreensão intercultural e a solução para análise de sentimentos.

Elas oferecem uma conectividade sem precedentes, amplificam vozes e permitem que pessoas de todo o mundo se engajem em causas comuns.

Resumindo, essa estratégia transforma informações dispersas de canais sociais em dados valiosos. Quando devidamente identificados, avaliados e classificados, comentários e reações refletem a imagem da sua empresa no meio digital.

1.2. O problema proposto

O atual trabalho tem como objetivo extrair dados de sentimento em uma plataforma específica, como um site de avaliação ou suporte ao cliente. Os dados utilizados neste projeto treinam algoritmo de computador para entender o texto de maneiras semelhantes aos humanos e classificá-los de maneiras positivas, negativas ou neutras.

A análise de sentimentos desempenha um grande papel em ajudar as empresas a desenvolver produtos e serviços mais inteligentes que atendam especificamente às necessidades dos clientes.

Os dados utilizados neste projeto foram originalmente disponibilizados pela rede social Twitter, transitado pela plataforma do Kaggle. (Disponível em: <https://www.kaggle.com/code/moisessjr/sentimentos-emo-es/> . Acesso em 22 de outubro de 2023).

Em síntese, pelo método do 5W's, apresento a estratificação do problema da pesquisa.

(Why?) Por que esse problema é importante?

Resposta:

A análise de sentimento fornece um embasamento fundamental para planejar o crescimento e a popularização das redes sociais em uma grande quantidade de dados (insights). Sendo assim, ela ajuda as empresas a entenderem o sentimento social, textual e técnico de sua marca, produto ou serviço.

(Who?) De quem são os dados analisados? De um governo? Um ministério ou secretaria? Dados de clientes?

Resposta:

Neste contexto os dados utilizados foram através de ferramentas de precisão estatísticos identificados através de usuários na web, insights, sistemas de service desk, reviews e avaliações classificadas pelo algoritmo.

(What?): Quais os objetivos com essa análise? O que iremos analisar?

Resposta:

Identificar cada sentimento associada a uma frase, comentário, fragmentos de discurso. Dessa forma, conseguimos identificar e interpretar a análise de sentimentos do usuário. Com isso, analisamos a extração de dados de sentimentos em uma plataforma específica determinando sugestões de melhoria nos produtos e gerenciando crises de imagem, fornecendo um embasamento fundamental para planejar suas ações de mídia sociais.

(Where?): Trata dos aspectos geográficos e logísticos de sua análise.

Resposta:

O dataset não apresenta informações geográficas.

(When?): Qual o período está sendo analisado? A última semana? Os últimos 6 meses? O ano passado?

Resposta:

O dataset apresenta informações do ano de 2022.

Para nosso projeto iremos utilizar os seguintes modelos:

- a) Árvore de Decisão
- b) Random Forest
- c) Regressão Logística

d) Naive Bayes

A ideia é comparar as métricas de cada um dos modelos na etapa de validação afim de verificamos o que apresenta os melhores resultados aplicáveis ao nosso problema.

A Tabela 1, documentada abaixo, resume bem cada um dos modelos utilizados neste trabalho,

Tabela 1 - Modelos De Machine Learning

Modelo	Definição
Árvore de Decisão	<p>É basicamente um mapeamento dos possíveis resultados que podem surgir a partir de uma decisão específica.</p> <p>Uma árvore de decisão é um algoritmo supervisionado que é utilizado para classificação em categorias discretas (sim ou não, por exemplo) e para prever valores numéricos (o valor do lucro em reais).</p>
Random Forest	<p>Significa floresta aleatória. Este nome explica muito bem o funcionamento do algoritmo.</p> <p>Em resumo, o random forest cria muitas árvores de decisão, de maneira aleatória, formando o que podemos entender como regras para tomada de decisão, sendo este conhecimento fundamental para o entendimento do algoritmo.</p>

Regressão Logística	É uma técnica de análises de dados que usa a estatística para encontrar as relações entre dois fatores de dados. Em seguida, essa relação tem como objetivo prever o valor desses fatores com base no outro (variável categórica, frequentemente binária, como sim ou não).
Naive Bayes	É classificada por um algoritmo que se baseia e gera uma tabela de probabilidades a partir de uma técnica de classificação de dados. É usado para machine learning, realizando o aprendizado bayesiano (ou seja, as classes).

Fonte: (Autor)

As etapas a seguir vão adentrar nos procedimentos executados no trabalho. Serão apresentados na tabela 2.

Tabela 2 – Procedimentos utilizados no trabalho

Entendimento do problema	
Etapas 1	<p>A primeira etapa consiste na identificação do problema que você deseja resolver através de dados para definir uma clareza para as pessoas que lerão o seu relatório.</p> <p>Em nosso projeto, vamos abordar análise de sentimento de comentários sobre marcas de consumo com machine learning.</p>

Entendimento dos dados	
Etapa 2	Esta etapa consiste em coletar, organizar e documentar todos os dados que se encontram disponíveis. Integrando-os em uma única fonte de dados. Buscando datasets contendo dados que se identificam com o problema. Esta fonte de dados é utilizada para a resolução do problema, analisar a qualidade desses dados, realizar análise descritiva e identificar padrões.
Preparação dos dados	
Etapa 3	Nesta etapa precisamos tratar os dados para que eles possam ser analisados e que as informações estão de acordo com o que se espera. Consistência de erros e valores ausentes, dados desbalanceados devem ser resolvidos para que possamos selecionar amostras aleatórias e utilizá-las para treino, validação e teste dos algoritmos.
Modelos de Machine Learning	
Etapa 4	Nesta etapa categorizamos a avaliação dos modelos de machine learning que vamos utilizar. Documentamos as técnicas selecionadas, motivo da escolha, avaliação e análise do comportamento das métricas para cada modelo de machine learning escolhido.
Avaliação final do modelo	
Etapa 5	Nesta etapa, serão analisados os modelos desenvolvidos com base nas métricas definidas anteriormente. Será examinada uma análise aprofundada do desempenho com base em métricas como acurácia, precisão, recall e F1-score. A validação cruzada e a avaliação em conjunto com dados de teste serão utilizadas para garantir a robustez do modelo. As definições e cálculos das métricas de avaliação estão detalhados mais adiante em nosso projeto, na descrição das etapas de avaliação dos modelos de machine learning.

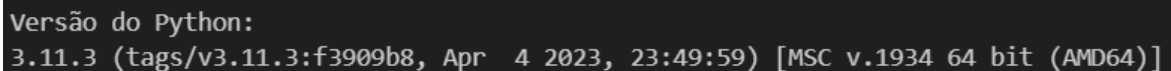
Fonte: (Autor)

2. Coleta de Dados

Para realizarmos nosso projeto utilizamos o Python versão 3.10.2 (Figura 1), que é uma linguagem de programação amplamente usada em aplicações da Web, desenvolvimento de software, ciência de dados e machine learning (ML). Os desenvolvedores usam o Python porque é eficiente e fácil de aprender e pode ser executada em muitas plataformas diferentes.

O Python aumenta a produtividade dos desenvolvedores, porque eles podem escrever um programa Python usando menos linhas de código em comparação a muitas outras linguagens.

Figura 1 - Versão do Python

A screenshot of a terminal window with a dark background. The text displayed is: 'Versão do Python: 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]'.

```
Versão do Python:  
3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)]
```

Fonte: (Autor)

Antes de importarmos nossas bibliotecas Python, executamos o código “pip install –U scikit-learn”, que é usada para instalar o pacote “Scikit-learn”, que é uma biblioteca para machine learning que fornece uma seleção de recursos eficazes para modelagem estatística, análise e mineração de dados, além de suporte supervisionado e não supervisionado. (Disponível em https://scikit-learn.org/stable/getting_started.html . Acesso em 20 de outubro de 2023).

Após a instalação das bibliotecas, além da Scikit-learn Importamos algumas outras bibliotecas Python (Figura 2), e utilizamos também bibliotecas nativas da linguagem, para realizarmos as etapas de análise, processamento, tratamento dos dados e criação dos modelos de machine learning. Importamos também “Pandas” e “Numpy” para manipulação, análise de dados e para realizar operações matemáticas e numéricas; “Matplotlib” para criar gráficos e visualizações. (Disponível em <https://pypi.org/project/matplotlib/> .Acesso em 20 de outubro de 2023); “BeautifulSoup” para extrair dados da web, executar raspagem de dados (web scraping). (Disponível

<https://pypi.org/project/beautifulsoup4/> . Acesso em 20 de outubro de 2023); “requests” para fazer requisição HTTP (Disponível <https://pypi.org/project/requests/> . Acesso 20 de outubro de 2023); “re” para manipulação de textos e padronização.

Figura 2 – Importando bibliotecas Python

```
import pandas as pd
import numpy as np
import sys
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import KFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix
import re
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import requests
```

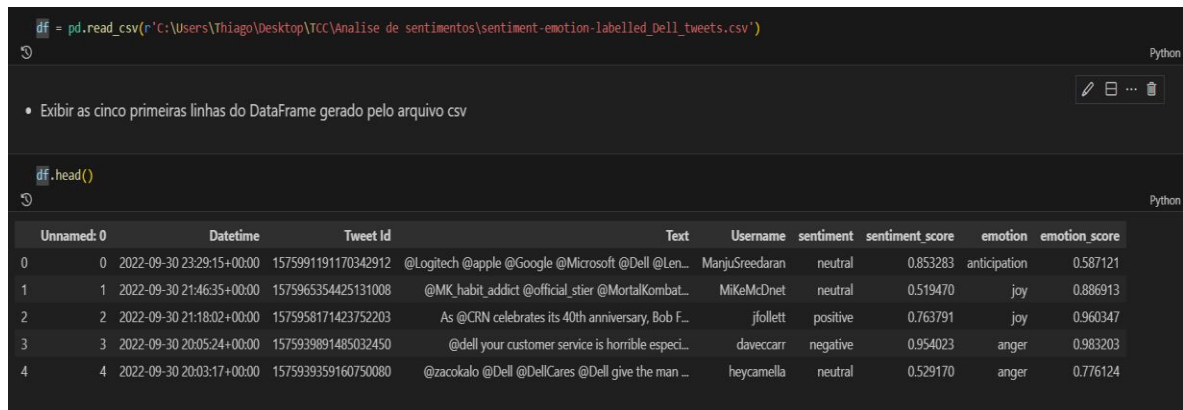
Fonte: (Autor)

A base de dados empregada está em formato csv (Comma Separated Values), que é o formato de importação e exportação mais prevalente para planilhas e bancos de dados. Fundamentalmente, trata-se de um tipo de arquivo de texto simples que armazena informações, onde cada linha denota um registro e os valores são separados por vírgulas (ou outro caractere delimitador).

Os dados utilizados neste projeto foram originalmente disponibilizados na API (Application Programming Interface) do Twitter, mas foram acessados através do Kaggle. (Disponível em: <https://www.kaggle.com/code/moisesjr/sentimentos-emo-es/input> . Acesso em 20 de outubro de 2023). De acordo com o apresentado na Figura 3, empregamos o comando "pd.read_csv" para ler e carregar um arquivo no formato CSV (Comma-

Separated Values) por meio da biblioteca "Pandas". O comando "df.head()" em Python é utilizado para mostrar as cinco primeiras linhas de um DataFrame. Isso é bastante proveitoso para visualizar de maneira rápida a estrutura e o conteúdo do conjunto de dados carregado.

Figura 3 - Realizando a leitura do DataFrame importado



The screenshot shows a Jupyter Notebook interface. The top cell contains the code `df = pd.read_csv(r'C:\Users\Thiago\Desktop\TCC\Analise de sentimentos\sentiment-emotion-labelled_Dell_tweets.csv')`. Below it, a message indicates that the first five lines of the DataFrame are being displayed. The second cell contains the code `df.head()`. Below this, a preview of the DataFrame is shown with the following columns: Unnamed: 0, Datetime, Tweet Id, Text, Username, sentiment, sentiment_score, emotion, and emotion_score. The first five rows of data are displayed.

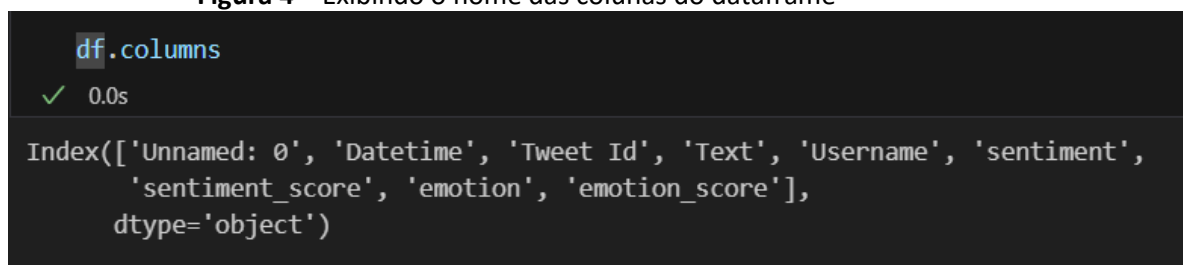
	Unnamed: 0	Datetime	Tweet Id	Text	Username	sentiment	sentiment_score	emotion	emotion_score
0	0	2022-09-30 23:29:15+00:00	1575991191170342912	@Logitech @apple @Google @Microsoft @Dell @Len...	ManjuSreedaran	neutral	0.853283	anticipation	0.587121
1	1	2022-09-30 21:46:35+00:00	1575965354425131008	@MK_habit addict @official_stier @MortalKombat...	MiKeMcDnet	neutral	0.519470	joy	0.886913
2	2	2022-09-30 21:18:02+00:00	1575958171423752203	As @CRN celebrates its 40th anniversary, Bob F...	jfollett	positive	0.763791	joy	0.960347
3	3	2022-09-30 20:05:24+00:00	1575939891485032450	@dell your customer service is horrible espec...	daveccarr	negative	0.954023	anger	0.983203
4	4	2022-09-30 20:03:17+00:00	1575939359160750080	@zacokalo @Dell @DellCares @Dell give the man ...	heycamella	neutral	0.529170	anger	0.776124

Fonte: (Autor)

Após importar o nosso conjunto de dados, conseguimos obter uma breve visualização das informações contidas na nossa coleção.

Posteriormente, solicitamos a exibição dos nomes das colunas do DataFrame utilizando o seguinte código (`df.columns`). Isso é feito para proporcionar uma visão inicial da estrutura do conjunto de dados, permitindo-nos compreender melhor as variáveis disponíveis e facilitando a análise exploratória. Conhecer os nomes das colunas é fundamental para direcionar futuras manipulações e extrações de informações específicas durante a análise de dados, como foi apresentado na figura 4. Utilizamos também o código (`df.info()`), para verificamos o tipo de cada variável, como foi apresentado na figura 5.

Figura 4 – Exibindo o nome das colunas do dataframe



The screenshot shows a Jupyter Notebook interface. The top cell contains the code `df.columns`. Below it, a green checkmark and the text "0.0s" indicate successful execution. The output is displayed as an Index of column names: Index(['Unnamed: 0', 'Datetime', 'Tweet Id', 'Text', 'Username', 'sentiment', 'sentiment_score', 'emotion', 'emotion_score'], dtype='object').

Fonte: (Autor)

Figura 5 – Verificando o tipo das variáveis

```
df.info()
✓ 0.1s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24970 entries, 0 to 24969
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Unnamed: 0      24970 non-null  int64
1   Datetime        24970 non-null  object
2   Tweet Id       24970 non-null  int64
3   Text            24970 non-null  object
4   Username        24970 non-null  object
5   sentiment       24970 non-null  object
6   sentiment_score 24970 non-null  float64
7   emotion         24970 non-null  object
8   emotion_score   24970 non-null  float64
dtypes: float64(2), int64(2), object(5)
memory usage: 1.7+ MB
```

Fonte: (Autor)

Após a conclusão dessas etapas, torna-se viável realizar uma análise mais aprofundada das variáveis presentes no nosso conjunto de dados. Registramos na Tabela 2 o nome, descrição de cada campo/coluna e o tipo de dado associado a cada uma das variáveis que estamos utilizando. Esta documentação fornece um guia valioso para compreender a natureza e o propósito de cada componente do dataset, facilitando assim a interpretação e manipulação adequada durante o processo de análise de dados.

Em relação aos tipos identificados em nosso conjunto de dados, observamos três categorias distintas:

- *Object*: O tipo "object" no Pandas é empregado para representar colunas de dados heterogêneos, ou seja, que podem conter diversos tipos como strings, números e objetos Python

- *Int64*: O tipo "int64" no Pandas é utilizado para armazenar colunas de dados numéricos inteiros de 64 bits com sinal.
- *Float64*: O tipo "float64" no Pandas é empregado para representar colunas de dados numéricos de ponto flutuante de 64 bits.

Tabela 2 - Descrição das colunas e tipos de dados do dataset

Nome da coluna/campo	Descrição	Tipo
Unnamed: 0	Código de identificação	Int64
Datetime	Data e horário do tweet	object
Tweet Id	Código de identificação do Tweet	Int64
Text	Texto do tweet	object
Username	Nome do usuário que twittou	object
Sentiment	Sentimento do tweet (neutro, positivo e negativo)	object
Sentimento_score	Pontuação de sentimento do tweet	float64
Emotion	Emoji/figurinha que ilustra o sentimento do tweet	object
Emotion_score	Pontuação de emoção	float64

Fonte: (Autor)

Com os dados importados para um DataFrame e após familiarizarmo-nos com as variáveis do conjunto de dados, podemos avançar para a próxima fase: o processamento e tratamento. Essa etapa tem como objetivo extrair insights ou informações relevantes que possam influenciar nos nossos modelos de Machine Learning.

3. Processamento/Tratamento de Dados

Após a importação dos dados para o nosso DataFrame, procedemos para a fase de processamento e tratamento. Essa fase engloba a coleta, organização e documentação abrangente de todos os dados disponíveis. O objetivo é identificar os elementos cruciais para a resolução do problema, analisar a qualidade dos dados, realizar uma análise descritiva e buscar padrões nos procedimentos adotados. Iniciamos essa fase verificando quantas linhas e quantas colunas o dataframe possui, utilizando o código (`df.shape`), como é mostrado na figura 6. Notamos que o dataframe possui 24970 linhas e 9 colunas. Logo em seguida, verificamos se no dataframe possui valores nulos nas colunas para garantir a integridade dos dados utilizando o código (`df.isna().sum()`), como foi apresentado na Figura 7. Essa etapa é crucial para identificar e lidar com possíveis lacunas ou ausências de informação que possam impactar as análises subsequentes. Em seguida utilizando o código (`df.drop_duplicates()`) verificando e excluindo algum dado duplicado. Essa operação é relevante para garantir a consistência e a precisão do conjunto de dados, eliminando entradas repetidas que podem distorcer as análises subsequentes, como apresentado na figura 8. Logo em seguida, como apresentado na figura 9, utilizamos o código (`df.rename()`) para renomear o nome das colunas do dataframe. Este procedimento permite personalizar os rótulos das colunas, tornando-os mais descritivos e alinhados com a interpretação e análise específica que estamos buscando realizar. Renomear as colunas pode facilitar a compreensão do conjunto de dados e melhorar a clareza durante as fases subsequentes de manipulação e análise.

Figura 6 – Verificando quantas linhas e quantas colunas




```
df.shape
✓ 0.0s
(24970, 9)
```

Fonte: (Autor)

Figura 7 – Verificando dados nulos no dataframe

```
df.isna().sum()
```




Unnamed: 0	0
Datetime	0
Tweet Id	0
Text	0
Username	0
sentiment	0
sentiment_score	0
emotion	0
emotion_score	0
dtype: int64	

Fonte: (Autor)

Figura 8 - Verificando e excluindo dados duplicados

```
df = df.drop_duplicates()
```



Fonte: (Autor)

Figura 9 – Renomeando as colunas

```
df.rename(columns ={
    'Unnamed: 0': 'id',
    'Datetime': 'data_horario',
    'Text': 'texto',
    'Username': 'usuario',
    'sentiment': 'sentimento',
    'sentiment_score': 'sentimento_score',
    'emotion': 'emoji',
    'emotion_score': 'emoji_score'
}, inplace=True )
```

Fonte: (Autor)

Em seguida, procedemos com o uso do código (`df.drop()`) para eliminar as colunas que não serão úteis na análise dos dados, como foi mostrado na figura 10. Essa etapa de remoção de colunas desnecessárias contribui para simplificar o conjunto de dados, focando apenas nas variáveis relevantes para os objetivos específicos da análise.

Figura 10 – Deletando colunas desnecessárias

```
df = df.drop(['data_horario', 'Tweet Id', 'usuario', 'sentimento_score', 'emoji_score'], axis=1)
```

Fonte: (Autor)

Logo após, realizamos a substituição dos valores de uma coluna visando aprimorar os resultados da nossa análise. Isso é alcançado por meio do código (`df['sentimento'].replace()`), assim como mostramos na figura 11. Essa ação é particularmente útil quando desejamos padronizar ou categorizar certos valores para facilitar a interpretação ou a aplicação de algoritmos de análise de dados. Imediatamente após, por meio do código (`df.sample()`), efetuamos um sorteio aleatório de todos os dados do DataFrame, como mostrado na figura 12. Essa abordagem é empregada para evitar que os dados permaneçam agrupados por categorias de sentimento, promovendo uma distribuição mais uniforme. Essa aleatorização contribui para uma aplicação mais eficaz do algoritmo, uma vez que impede que padrões específicos relacionados à ordenação inicial dos dados influenciem de maneira indesejada o desempenho do modelo.

Figura 11 – Substituindo valores da coluna

```
df['sentimento'] = df['sentimento'].replace({'positive': 0, 'negative': 1, 'neutral': 2})
```

Fonte: (Autor)

Figura 12 – Sorteio aleatório dos dados do dataframe

```
df = df.sample(frac = 1)
df
```

✓ 0.0s

	id	texto	sentimento	emoji
8101	8101	@mrlarryridley @Compete4ever @esportsXTRAsHow ...	2	anticipation
9051	9051	@RajendraKapase @DellCares @Dell don't trusts ...	1	anger
12167	12167	@Dell expects you to include your laptop passw...	1	disgust
10062	10062	Charles Kaufman and Radia Perlman giving their...	2	anticipation
20120	20120	@kfranasz @Dell That's what I thought. Hopeful...	2	anger
...
10036	10036	@RobertGauss5 @Dell They have me, as well. Was...	1	disgust
7784	7784	@EricMarsi @Dell That is awesome level of hack...	0	joy
12427	12427	@Dell atleast if it the things are not in your...	1	anger
23754	23754	Teen hacker finds bug that lets him control 25...	2	anticipation
13643	13643	@seankinneyRCR @matteastwood @united @Dell @De...	2	anticipation

24970 rows × 4 columns

Fonte: (Autor)

Com o intuito de padronizar os textos, implementei a função (def conversão_de_texto(texto)) a qual converte o texto para minúsculo e remove espaços extras, caracteres especiais, URLs, entre outros, conforme demonstrado na Figura 13. Essa abordagem de pré-processamento é valiosa para normalizar o formato dos textos, tornando-os mais homogêneos e facilitando a análise subsequente, especialmente quando se lida com dados textuais em algoritmos de machine learning. Posteriormente, procedemos à aplicação da função na coluna específica conforme necessário tal como evidenciado na Figura 14.

Figura 13 – Criação da função de conversão de texto

```
def conversao_de_texto(texto):
    texto = texto.lower()
    texto = re.sub('[\.\*\?\']', '', texto)
    texto = re.sub("\W", " ", texto)
    texto = re.sub('https?://\S+|www\.\S+', '', texto)
    texto = re.sub('<.*?>+', '', texto)
    texto = re.sub('[%s]' % re.escape(string.punctuation), '', texto)
    texto = re.sub('\n', '', texto)
    texto = re.sub('\w*\d\w*', '', texto)

    return texto
```

Fonte: (Autor)

Figura 14 – Aplicação da função de conversão de texto

```
df['texto'] = df['texto'].apply(conversao_de_texto)
df.head()
```

✓ 2.6s

	id	texto	sentimento	emoji
8101	8101	mrlarryridley esportsxtrashow bncnews bn...	2	anticipation
9051	9051	rajendrakupase dellcares dell don t trusts ...	1	anger
12167	12167	dell expects you to include your laptop passw...	1	disgust
10062	10062	charles kaufman and radia perlman giving their...	2	anticipation
20120	20120	kfranasz dell that s what i thought hopeful...	2	anger

Fonte: (Autor)

Depois de realizar a verificação e exclusão dos dados ausentes em nossa base, juntamente com outras etapas de preparação e adequação, progredimos para uma análise mais aprofundada das características presentes e sua relação com o objetivo do nosso trabalho. Essa fase envolve uma exploração mais detalhada das informações contidas no conjunto de dados, buscando insights relevantes e compreensão mais aprofundada das relações entre as variáveis em estudo. Essa análise mais profunda é crucial para embasar e informar as decisões durante as etapas subsequentes do processo de análise de dados.

4. Análise e Exploração dos Dados

Com o objetivo de simplificar nossa análise exploratória de dados e assegurar uma compreensão mais abrangente das informações contidas, buscaremos explorar de maneira mais precisa a diversidade e a natureza das características presentes. Identificar possíveis padrões, tendências ou peculiaridades é crucial, pois tais elementos podem contribuir

significativamente para o entendimento e a análise subsequente, proporcionando insights valiosos que fundamentam decisões e interpretações mais informadas.

A análise exploratória em Machine Learning representa uma etapa crucial no processo de exploração e compreensão inicial dos dados. Essa fase envolve a aplicação de técnicas estatísticas e visualizações com o propósito de identificar padrões, tendências, anomalias e relacionamentos nos dados. O objetivo é extrair insights valiosos que possam orientar as decisões durante as fases subsequentes do desenvolvimento do modelo e proporcionar uma compreensão mais aprofundada do conjunto de dados em questão.

Uma significação de análise exploratória em Machine Learning:

“Exploratory data analysis (EDA) comprehends interactive presentation, exploration, and discovery of data, their trends, behaviors, and relationships with the visualization as one of the key tools in this process. Additionally, the EDA is often also used as a support in variable (or feature) selection for data analytics models and is an integral part of systems where a human in the loop (and machine-to-human interaction) is foreseen (for instance, in digital twins).

An initial step of the exploratory data analysis is the presentation, and it aims to provide a swift and cursory familiarity with the dataset. It involves computing and interactively visualizing descriptive statistics based on the data type of the variable, by utilizing a wide spectrum of visualization techniques (i.e., histograms, scatter plots bubble charts, matrix plots, box-and-whisker plots, etc.).”

(Ivana Cavar Semanjski, in [Smart Urban Mobility](#), 2023).

Nesta descrição, destaca-se a utilização de abordagens visuais para resumir as características fundamentais dos conjuntos de dados, visando descobrir padrões, relações e insights ocultos. Essa prática contribui significativamente para a compreensão e a identificação das variáveis essenciais nas análises e na modelagem subsequente. Utilizando uma ampla gama de técnicas de visualização, como histogramas, gráficos de dispersão, bolhas, matriz, caixa, entre outros. O uso de métodos visuais proporciona uma representação mais clara e intuitiva das nuances presentes nos dados, facilitando a tomada de decisões informadas durante o desenvolvimento de modelos em Machine Learning.

Iniciando a nossa fase de análise exploratória, vamos reexaminar os dados únicos para cada coluna, buscando uma compreensão mais aprofundada da situação, como é mostrado na Figura 15.

Figura 15 - Verificando os dados únicos por coluna para checar distribuição

```
for column in df.columns:
    print(f'Coluna {column}: {df[column].unique()}')
    print('-----' * 10)
✓ 0.0s
```

Coluna id: [12821 10903 4692 ... 14993 7480 9901]

Coluna texto: [' dellcares dell please suggested what should i do michael dell dellcares delltech'
'congrats palmoze cohort for landing a role with dell technologies as an associate systems engineer have a great
'also i saw there are multiple brands repaired at this center no sure dell dellindia claiming as authorised service
...
'am i the only tech guy who can t figure out why dell laptops are so expensive i used to recommend them exclusively no
' best dell laptops top dell laptops you can buy thread '
' roxannehoge dell ugh tight schedule ']

Coluna sentimento: [2 0 1]

Coluna emoji: ['anticipation' 'joy' 'anger' 'disgust' 'sadness' 'fear' 'optimism'
'surprise']

Fonte: (Autor)

Ao examinarmos as distribuições de dados em cada coluna, acompanhadas das análises dos tipos de dados correspondentes, torna-se possível estruturar uma abordagem mais eficaz para analisar cada uma das características (features). Essa análise combinada proporciona uma visão mais completa e contextualizada, permitindo uma compreensão mais aprofundada das particularidades de cada variável no conjunto de dados.

Nossas descobertas revelaram que o conjunto de dados apresenta informações que podem ser categorizadas, fornecendo uma orientação mais específica para nossas análises. A categoria identificada é a seguinte:

- **Informações sobre avaliação do comentário:** Representado pelas colunas texto e sentimento. Essas colunas refletem o texto do comentário, que dependendo das palavras que estão escritas, ela terá um valor no sentimento sendo 0, 1 ou 2 (positivo, negativo e neutro, respectivamente).

Ao classificarmos nossa característica dessa maneira, torna-se mais praticável conduzir uma análise mais aprofundada em cada uma delas. Essa abordagem possibilita uma análise exploratória mais detalhada, proporcionando uma consciência situacional mais robusta e resultando em um entendimento mais claro do problema em questão. Categorizar as características permite uma exploração mais focalizada, facilitando a identificação de padrões específicos, tendências e relações relevantes dentro de cada categoria, contribuindo para uma análise mais abrangente e informada.

Para as análises da categoria Informações sobre avaliação do comentário, inicialmente, verificamos a divisão de nosso dataset e balanceamento das classes considerando a coluna “sentimento”, como é mostrado na Figura 17.

O código exibido na Figura 16 realiza a verificação da divisão do dataset, primeiro estamos fazendo a contagem dos sentimentos, criamos a variável ‘contagem_sentimento_0’ para fazer a contagem do sentimento: positivo, utilizamos o código `(df['sentimento'] == 0).sum())` e gravamos na variável criada. Também criamos a variável ‘contagem_sentimento_1’ para fazer a contagem do sentimento: negativo, utilizamos o código `(df['sentimento'] == 1).sum())` e gravamos na variável criada. E criamos a variável ‘contagem_sentimento_2’ para fazer a contagem do sentimento: neutro e utilizamos o código `(df['sentimento'] == 2).sum())` e gravamos na variável criada. Definimos uma variável com o nome ‘sentimentos’ com uma lista de strings, sendo Positivo, Negativo e Neutro. Também, definimos uma variável com o nome ‘contagens’, com uma lista obtendo as variáveis criada nos passos anteriores, sendo elas, contagem_sentimento_0, contagem_sentimento_1 e contagem_sentimento_2, para armazenar as contagens correspondentes a cada sentimento. Após esses passos, chegou a hora de plotar o gráfico, primeiro definimos o tamanho do gráfico, utilizamos o código `(plt.figure(figsize = (8,6)))`, sendo que ele terá 8 unidades de largura e 6 unidades de altura. Definido o tamanho do gráfico, vamos gerar as barras, utilizando `(plt.bar(sentimentos, contagens, color=['green', 'red', 'blue']))`, criamos um gráfico de barras usando os rótulos e contagens, com cores distintas para cada sentimento sendo (verde para positivo, vermelho para negativo e azul para neutro). Definido as barras, colocamos um rótulo em nosso gráfico, com o código `(plt.ylabel('Quantidade'))` adicionamos um rótulo ao eixo y indicando a quantidade. Logo depois, demos um título ao nosso gráfico, com o seguinte código `(plt.title('Contagem de Sentimentos'))`. Finalizado essas etapas, finalmente podemos exibir nosso gráfico com o código `(plt.show())`.

Figura 17 - Verificando a divisão de nosso dataset e balanceamento das classes

```
# Calculando as contagens de sentimentos
contagem_sentimento_0 = (df['sentimento'] == 0).sum()
contagem_sentimento_1 = (df['sentimento'] == 1).sum()
contagem_sentimento_2 = (df['sentimento'] == 2).sum()

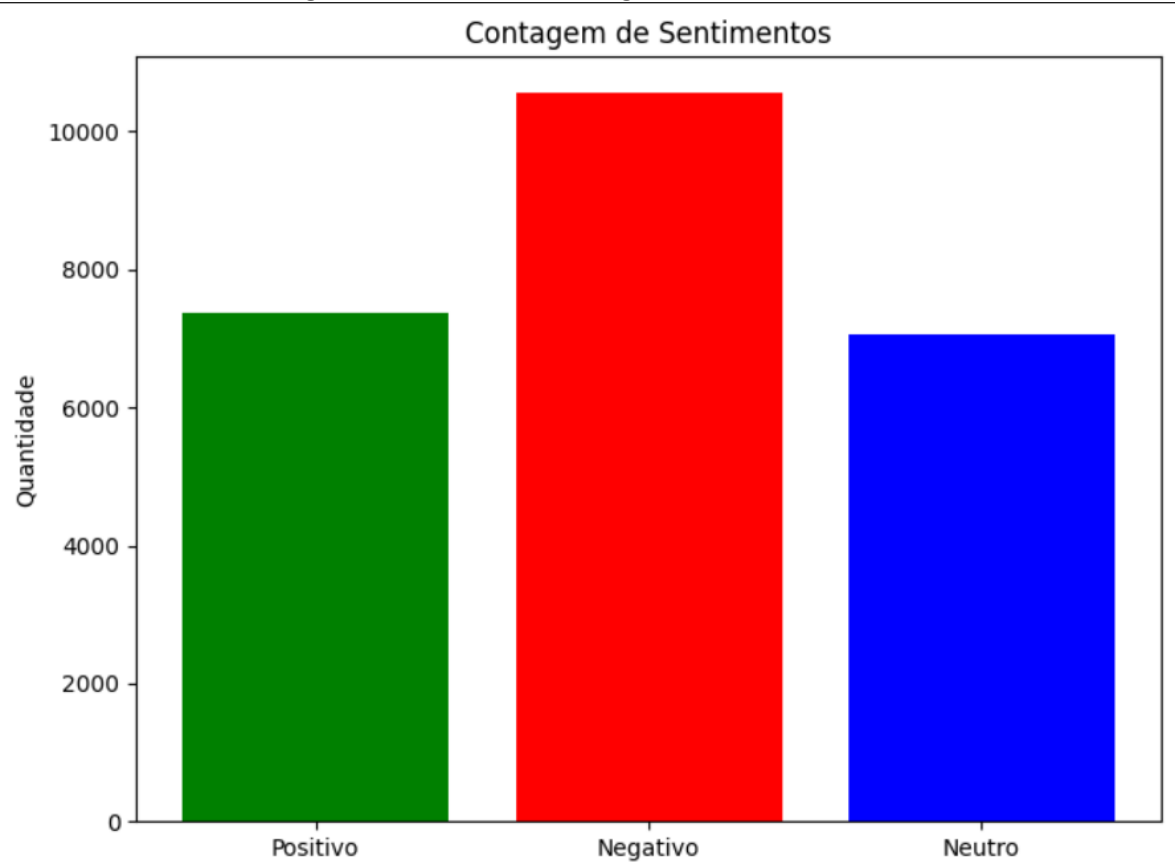
# Criando os rótulos das colunas
sentimentos = ['Positivo', 'Negativo', 'Neutro']

# Criando as contagens
contagens = [contagem_sentimento_0, contagem_sentimento_1, contagem_sentimento_2]

# Criando o gráfico de barras
plt.figure(figsize=(8, 6))
plt.bar(sentimentos, contagens, color=['green', 'red', 'blue'])
plt.ylabel('Quantidade')
plt.title('Contagem de Sentimentos')
plt.show()
```

Fonte: (Autor)

Figura 16 – Gráfico de contagem de sentimentos



Fonte: (Autor)

Na categoria “negativo” da coluna analisada, é notável uma predominância em termos de quantidade de dados. Entretanto, é relevante destacar que essa disparidade não configura um cenário desfavorável, visto que, em comparação com os sentimentos “positivo” e “neutro”, a discrepância é compensada. Essa distribuição equilibrada entre as categorias sugere que a variação na quantidade de dados não compromete a integridade da análise, proporcionando uma perspectiva mais equitativa e representativa dos diferentes sentimentos presentes no conjunto de dados em avaliação. Essa compreensão detalhada da distribuição contribui para uma interpretação mais precisa e contextualizada dos resultados obtidos durante a análise exploratória. Na coluna negativo, observa-se uma maior quantidade de dados, mas não é uma diferença que possa trazer resultados ruins, pois a diferença em comparado com o sentimento positivo e neutro se balanceia.

5. Criação de Modelos de Machine Learning

Machine Learning (ML), ou Aprendizado de Máquina em português, representa um subconjunto da inteligência artificial (IA) dedicado à construção de sistemas capazes de aprender e aprimorar seu desempenho a partir dos dados que são expostos. Essa abordagem permite que os sistemas automaticamente identifiquem padrões, façam ajustes e melhorem suas capacidades ao longo do tempo, sem uma programação explícita para cada tarefa. Em essência, o Machine Learning capacita as máquinas a adquirir conhecimento e tomar decisões de maneira autônoma com base nas informações disponíveis.

O aprendizado de máquina, também conhecido como machine learning, pode ser categorizado em três tipos principais:

a) **Aprendizado Supervisionado:** Este tipo busca encontrar uma função a partir de dados que possuem rótulos de entrada e saída. O objetivo principal é prever uma variável desejada com base nos padrões identificados nos dados rotulados.

b) **Aprendizado Não Supervisionado:** Nesse caso, o foco é explorar ou descrever um conjunto de dados sem a presença de atributos de saída rotulados. O intuito principal é identificar padrões desconhecidos ou estruturas intrínsecas no conjunto de dados.

c) **Aprendizado por Reforço:** No aprendizado por reforço, a interação e o feedback desempenham um papel crucial no desenvolvimento do processo de aprendizado. Os modelos aprendem por meio de tentativa e erro, recebendo recompensas ou penalidades com base em suas ações, o que é especialmente útil em ambientes dinâmicos.

No trabalho atual, estamos focando no aprendizado supervisionado, mais precisamente na classificação de avaliação de usuários sobre determinado produto. Onde os dados rotulados podem consistir em amostras de texto (como avaliações de produtos ou postagens em redes sociais) associadas a rótulos que indicam se o sentimento é positivo, negativo ou neutro. O algoritmo é treinado para aprender a relação entre as características do texto e as categorias de sentimento correspondentes.

Nesta fase do trabalho, estabelecemos e avaliamos os modelos de Aprendizado de Máquina que serão testados. Registrando as técnicas escolhidas, explicamos a lógica por trás da seleção, e conduzimos a avaliação e análise do desempenho das métricas para cada modelo de Aprendizado de Máquina escolhido. Após o processamento realizado na etapa anterior, a maioria das características (features) já foi preparada para os modelos. Agora, a próxima etapa envolve a padronização das características numéricas e a determinação dos modelos específicos que serão utilizados na análise e no processo de tomada de decisão. Este passo é fundamental para garantir a consistência e a eficácia dos modelos ao serem aplicados aos dados preparados.

Com o objetivo de comparar o desempenho e as melhorias do modelo, criamos bases de referência (baselines) para cada um dos modelos a serem avaliados. Vamos proceder com a separação dos dados destinados ao treinamento e aos testes. É importante observar que os dados de teste serão reservados exclusivamente para a fase final do projeto, garantindo assim uma avaliação mais realista do desempenho dos modelos. Essa prática é essencial para verificar como os modelos generalizam para novos dados, proporcionando uma validação robusta de sua eficácia e capacidade de lidar com casos não vistos durante o treinamento.

Em nosso trabalho serão utilizados 4 modelos de Machine Learning, são eles: *Logistic Regression* (Regressão Logística), *Decision Tree* (Árvore de decisão), *Random Forest* (Floresta Aleatória) e *Naive Bayes*, ambos são modelos de classificação. O objetivo é comparar as métricas de avaliação de cada modelo a fim de chegarmos a uma conclusão de qual modelo terá um melhor resultado em uma aplicação aos nossos problemas.

Antes vamos conhecer sobre as métricas de avaliação que será analisadas para sabermos a eficácia dos nossos modelos. Com esse intuito, vamos utilizar duas funções da biblioteca Scikit-learn: “accuracy_score” e “classification_report”. Essas funções fornecem informações importantes na hora de avaliarmos o nosso modelo, que são elas: acurácia, precisão, recall e F1 score.

A acurácia indica a porcentagem de previsões corretas feitas pelo nosso modelo em relação ao total de observações.

$$\text{Acurácia} = (\text{Verdadeiros Positivos} + \text{Verdadeiros Negativos}) / \text{Total de amostras}$$

Precisão refere-se à quantidade proporcional de identificações positivas feitas corretamente pelo modelo. Essa métrica fornece informações sobre a precisão das identificações positivas, indicando qual proporção dessas identificações estava correta.

$$\text{Precisão} = \text{Verdadeiros Positivos} / (\text{Verdadeiros Positivos} + \text{Falsos Positivos})$$

O recall, também conhecido como sensibilidade ou taxa de verdadeiros positivos, representa a proporção de dados corretamente classificados como verdadeiros em relação ao total de resultados verdadeiros presentes na amostra.

$$\text{Recall} = \text{Verdadeiros Positivos} / (\text{Verdadeiros Positivos} + \text{Falsos Negativos})$$

A F1 score traz a média ponderada da precisão e do recall, oferecendo um número único que indica a qualidade geral do modelo. Essa métrica é uma média harmônica entre a precisão e o recall, sendo especialmente útil quando há um desejo de equilibrar essas duas métricas, considerando ambas igualmente importantes. A pontuação ideal para a F1 score é 1, indicando um equilíbrio perfeito entre precisão e recall. Por outro lado, o pior valor possível é 0, sugerindo uma situação em que a precisão e o recall são ambos nulos.

$$\text{F1 score} = (2 \times \text{Precisão} \times \text{Recall}) / (\text{Precisão} + \text{Recall})$$

Dando sequência ao nosso trabalho, iremos dividir o nosso conjunto de dados em conjuntos de treinamento e teste, visando testar o modelo com uma base à qual ele não teve contato prévio. A divisão será feita de maneira estratificada, levando em conta a classe, e seguirá a seguinte proporção: 75% para o conjunto de treinamento e 25% para o conjunto de teste, como mostrado na figura 18.

Figura 18 – Separando a base de dados em treino e teste

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, stratify=Y, random_state=50)
print("Treinamos com {} elementos e testaremos com {} elementos".format(len(x_train), len(x_test)))
```

✓ 0.0s

Treinamos com 18727 elementos e testaremos com 6243 elementos

Fonte: (Autor)

Em seguida, como vamos trabalhar com texto, teremos que converter os textos para números, pois como os algoritmos de machine learning trabalham com dados numéricos como entrada. Essa técnica é conhecida como “Vetorização de texto”. E com isso utilizamos a classe *TfidfVectorizer* do scikit-learn que converte uma coleção de documentos de texto em uma matriz de recursos TF-IDF, como é mostrado na figura 19. Ela leva em consideração a importância de cada palavra, considerando a frequência em um documento específico em relação à frequência em todo o conjunto de dados.

Figura 19 – Conversão de texto para número

```
vectorizer = TfidfVectorizer()
xv_train = vectorizer.fit_transform(x_train)
xv_test = vectorizer.transform(x_test)
```

Fonte: (Autor)

Depois da preparação dos nossos dados, podemos enfim começar a treinar nossos modelos. Vamos começar pelo modelo *Logistic Regression* (LR), que é um modelo estatístico eficaz quando estamos lidando com problemas de classificação binária. Ela é particularmente útil para prever a probabilidade de ocorrência de um evento específico,

como "sim" ou "não", "positivo" ou "negativo". Como mostrado na figura 20, primeiro instanciamos o modelo com o código (`lr = LogisticRegression()`), para depois treinarmos nosso modelo utilizando os dados determinados como dados de treinamento, sendo “`xv_train`” é o conjunto de dados de treino e “`y_train`” são os rótulos correspondentes, como diz o código (`lr.fit(xv_train, y_train)`). Com nosso modelo treinado podemos fazer nossa primeira previsão utilizando o modelo de LR, como é mostrado na figura 21.

Figura 20 – Instanciando e treinando o modelo

```
lr = LogisticRegression()
lr.fit(xv_train, y_train)
```

Fonte: (Autor)

Figura 21 – Visualização da acurácia da previsão

```
previsao_lr = lr.predict(xv_test)
acuracia = accuracy_score(y_test, previsao_lr)
f'Acurácia: {acuracia * 100: .2f} %'
```

✓ 0.0s

```
'Acurácia: 77.43 %'
```

Fonte: (Autor)

Utilizamos o método “Predict” do modelo LR para fazer previsões no conjunto de teste “`xv_test`”. Em seguida, calculamos a acurácia comparando as previsões “`previsao_lr`” com os rótulos reais do conjunto de teste “`y_test`”, usando a função “`accuracy_score`” do scikit-learn e imprimimos o resultado da acurácia, onde com nossos dados de teste, teve uma percentagem de 77.43% de acertos. Agora vamos verificar os resultados das nossas métricas, utilizando o código (`print(classification_report(y_test, previsao_lr))`), como é mostrado na figura 22.

Figura 22 – Resultados das métricas

```
print(classification_report(y_test, previsao_lr))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.79	0.77	0.78	1842
1	0.81	0.87	0.84	2639
2	0.69	0.63	0.66	1762
accuracy			0.77	6243
macro avg	0.76	0.76	0.76	6243
weighted avg	0.77	0.77	0.77	6243

Fonte: (Autor)

O (`classification_report`) fornece informações detalhadas para cada classe individualmente, permitindo uma avaliação mais específica do desempenho do modelo em cada categoria. No geral, a acurácia do modelo é de aproximadamente 77%, e as métricas de precisão, recall e F1-score variam para cada classe, como podemos ver a classe 0 possui uma precisão de acertos de 79%, um recall de 78% e um f1 score de 78%, já a classe 1 possui uma precisão de acertos de 81%, recall de 87% e f1 score de 84%, a classe 2 possui uma precisão de acertos de 69%, um recall de 63% e f1 score de 66%.

Para verificarmos se nosso modelo teve um ótimo resultado de treinamento, vamos utilizar o Dummy Classifier (DC) como referência, que é um modelo simples e básico que utilizamos como linha de base (baseline) para avaliarmos o desempenho dos nossos modelos mais sofisticados. Primeiro iremos instanciar o Dummy Classifier, com o código (`dummy = DummyClassifier()`), depois fizemos o treinamento do DC, com o código (`dummy.fit(xv_train, y_train)`), após treinar, podemos fazer a previsão, com o código (`previsões_dummy = dummy.predict(x_test)`) e geramos a acurácia com o (`acurácia_dummy = accuracy_score(y_test, previsões_dummy)*100`), como é mostrado na figura 23.

Figura 23 – Modelo de referência

```
# Para testar se as acurácias estão boas ou ruins --- Usando como baseline
dummy = DummyClassifier()
dummy.fit(xv_train, y_train)
previsoes_dummy = dummy.predict(x_test)

acuracia_dummy = accuracy_score(y_test, previsoes_dummy) * 100
print("A acurácia do dummy foi %.2f%%" % acuracia_dummy)
✓ 0.0s

A acurácia do dummy foi 42.27%
```

Fonte: (Autor)

Como a acurácia de modelo que estamos usando como referência foi de 42.27%, podemos concluir que o nosso modelo LR que possuiu um ótimo resultado tendo uma acurácia de 77.43%.

Após treinarmos o modelo LR, treinamos o modelo de Decision Tree (DT), que é um algoritmo de aprendizado de máquina utilizado para tarefas de classificação e regressão. Ela é uma estrutura de árvore onde cada nó interno representa uma decisão baseada em um atributo, cada ramificação representa o resultado de uma decisão e cada folha representa a classe ou valor predito. Benefícios da árvore de decisão incluem a interpretabilidade, facilidade de compreensão e a habilidade de lidar com dados numéricos e categóricos. No entanto, árvores de decisão podem ser vulneráveis a sobreajuste (overfitting), que se refere a uma condição em que o modelo se ajusta muito bem aos dados de treinamento, mas não se sai bem com dados novos. Como mostrado na figura 24, primeiro instanciamos o modelo com o código (`dt = DecisionTreeClassifier()`), para depois treinarmos nosso modelo utilizando os dados determinados como dados de treinamento, sendo “`xv_train`” é o conjunto de dados de treino e “`y_train`” são os rótulos correspondentes, como diz o código (`dt.fit(xv_train, y_train)`). Com nosso modelo treinado podemos fazer nossa previsão utilizando o modelo de DT, como é mostrado na figura 25.

Figura 24 - Instanciando e treinando o modelo

```
dt = DecisionTreeClassifier()
dt.fit(xv_train, y_train)
✓ 34.8s
```

▼ DecisionTreeClassifier
DecisionTreeClassifier()

Fonte: (Autor)

Figura 25 - Visualização da acurácia da previsão

```
previsao_dt = dt.predict(xv_test)
```

✓ 0.2s

- Calculando a acurácia do modelo de árvore de decisão

```
acuracia = accuracy_score(y_test, previsao_dt)
f'Acurácia: {acuracia * 100: .2f} %'
```

✓ 0.0s

'Acurácia: 63.83 %'

Fonte: (Autor)

Utilizamos o método “Predict” do modelo DT para fazer previsões no conjunto de teste “xv_test”. Em seguida, calculamos a acurácia comparando as previsões “previsão_dt” com os rótulos reais do conjunto de teste “y_test”, usando a função “accuracy_score” do scikit-learn e imprimimos o resultado da acurácia, onde com nossos dados de teste, teve uma percentagem de 63.83% de acertos. Agora vamos verificar os resultados das nossas métricas, utilizando o código (print(classification_report(y_test, previsao_dt))), como é mostrado na figura 26.

Figura 26 - Resultados das métricas

```
print(classification_report(y_test, previsao_dt))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.64	0.63	0.63	1842
1	0.71	0.71	0.71	2639
2	0.53	0.54	0.53	1762
accuracy			0.64	6243
macro avg	0.63	0.63	0.63	6243
weighted avg	0.64	0.64	0.64	6243

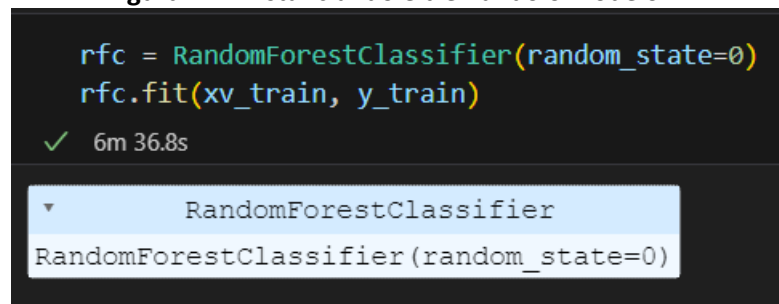
Fonte: (Autor)

No geral, a acurácia do modelo é de aproximadamente 63,83%, e as métricas de precisão, recall e F1-score variam para cada classe, como podemos ver a classe 0 possui uma precisão de acertos de 64%, um recall de 63% e um f1 score de 63%, já a classe 1 possui uma precisão de acertos de 71%, recall de 71% e f1 score de 71%, a classe 2 possui uma precisão de acertos de 53%, um recall de 54% e f1 score de 53%.

Como a acurácia de modelo que estamos usando como referência foi de 42.27%, podemos concluir que o nosso modelo DT que possuiu um ótimo resultado tendo uma acurácia de 63,83%, mas ele é inferior a acurácia do modelo LR que foi de 77,43%.

Após treinarmos os modelos de LR e DT, treinamos o modelo de Random Forest Classifier(RFC), que é um algoritmo de aprendizado de máquina que utiliza a técnica de ensemble, combinando várias árvores de decisão para melhorar o desempenho e a generalização do modelo. Como mostrado na figura 27, primeiro instanciamos o modelo com o código (`rfc = RandomForestClassifier(random_state=0)`), o parâmetro “`random_state`” é utilizado para inicializar o gerador de números aleatórios interno do RFC, garantindo que a geração de números aleatórios sejam reproduzível. Depois treinamos nosso modelo utilizando os dados determinados como dados de treinamento, sendo “`xv_train`” é o conjunto de dados de treino e “`y_train`” são os rótulos correspondentes, como diz o código (`rfc.fit(xv_train, y_train)`). Com nosso modelo treinado podemos fazer nossa previsão utilizando o modelo de RFC, como é mostrado na figura 28.

Figura 27 - Instanciando e treinando o modelo



```
rfc = RandomForestClassifier(random_state=0)
rfc.fit(xv_train, y_train)
```

✓ 6m 36.8s

▼ RandomForestClassifier

```
RandomForestClassifier(random_state=0)
```

Fonte: (Autor)

Figura 28 - Visualização da acurácia da previsão

```
# Fazendo previsões com base nos dados treinados
# Usando as características dos dados de teste
previsao_rfc = rfc.predict(xv_test)

acuracia = accuracy_score(y_test, previsao_rfc)
f'Acurácia: {acuracia * 100: .2f} %'

# O algoritmo classificador de floresta aleatória acerta 99% das previsões
✓ 0.6s

'Acurácia: 73.81 %'
```

Fonte: (Autor)

Utilizamos o método “Predict” do modelo RFC para fazer previsões no conjunto de teste “xv_test”. Em seguida, calculamos a acurácia comparando as previsões “previsão_rfc” com os rótulos reais do conjunto de teste “y_test”, usando a função “accuracy_score” do scikit-learn e imprimimos o resultado da acurácia, onde com nossos dados de teste, teve uma percentagem de 73,81% de acertos. Agora vamos verificar os resultados das nossas métricas, utilizando o código (print(classification_report(y_test, previsao_rfc))), como é mostrado na figura 29.

Figura 29 – Resultados das métricas

```
print(classification_report(y_test, previsao_rfc))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.80	0.67	0.73	1842
1	0.73	0.91	0.81	2639
2	0.69	0.55	0.61	1762
accuracy			0.74	6243
macro avg	0.74	0.71	0.72	6243
weighted avg	0.74	0.74	0.73	6243

Fonte: (Autor)

No geral, a acurácia do modelo é de aproximadamente 73,81%, e as métricas de precisão, recall e F1-score variam para cada classe, como podemos ver a classe 0 possui uma precisão de acertos de 80%, um recall de 67% e um f1 score de 73%, já a classe 1 possui uma precisão de acertos de 73%, recall de 91% e f1 score de 81%, a classe 2 possui uma precisão de acertos de 69%, um recall de 55% e f1 score de 61%.

Como a acurácia de modelo que estamos usando como referência foi de 42.27%, podemos concluir que o nosso modelo RFC que possuiu um ótimo resultado tendo uma acurácia de 73,81%, mas ele ainda é inferior a acurácia do modelo LR que foi de 77,43%.

Após treinarmos os modelos de LR, DT e RFC, treinamos o modelo de Naive Bayes(NB), que é um algoritmo de classificação probabilístico fundamentado no Teorema de Bayes. A sua abordagem "ingênua" pressupõe independência condicional entre cada par de recursos, embora essa suposição não seja realista para muitos conjuntos de dados do mundo real. Apesar dessa simplificação, ele é especialmente útil em problemas de classificação de texto, como a categorização de e-mails como spam ou não spam. Como mostrado na figura 30, primeiro instanciamos o modelo com o código (`nb = MultinomialNB()`), assim treinamos nosso modelo utilizando os dados determinados como dados de treinamento, sendo “`xv_train`” é o conjunto de dados de treino e “`y_train`” são os rótulos correspondentes, como diz o código (`nb.fit(xv_train, y_train)`). Com nosso modelo treinado podemos fazer nossa previsão utilizando o modelo de RFC, como é mostrado na figura 31.

Figura 30 - Instanciando e treinando o modelo

```
# Crie uma instância do modelo Naive Bayes
nb = MultinomialNB()

✓ 0.0s

# Supondo que 'X_train' são os recursos e 'y_train' são os rótulos
nb.fit(xv_train, y_train)

✓ 0.0s

▼ MultinomialNB
MultinomialNB()
```

Fonte: (Autor)

Figura 31 - Visualização da acurácia da previsão

```
# Faça previsões no conjunto de teste
previsao_nb = nb.predict(xv_test)

# Avalie o desempenho do modelo
acuracia = accuracy_score(y_test, previsao_nb)
f'Acurácia: {acuracia * 100: .2f} %'
```

✓ 0.0s

'Acurácia: 69.41 %'

Fonte: (Autor)

Utilizamos o método “Predict” do modelo NB para fazer previsões no conjunto de teste “xv_test”. Em seguida, calculamos a acurácia comparando as previsões “previsão_nb” com os rótulos reais do conjunto de teste “y_test”, usando a função “accuracy_score” do scikit-learn e imprimimos o resultado da acurácia, onde com nossos dados de teste, teve uma percentagem de 69,41% de acertos. Agora vamos verificar os resultados das nossas métricas, utilizando o código (print(classification_report(y_test, previsão_nb))), como é mostrado na figura 32.

Figura 32 – Resultados das métricas

```
print(classification_report(y_test, previsao_nb))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.78	0.66	0.71	1842
1	0.64	0.98	0.78	2639
2	0.81	0.30	0.44	1762
accuracy			0.69	6243
macro avg	0.74	0.65	0.64	6243
weighted avg	0.73	0.69	0.66	6243

Fonte: (Autor)

No geral, a acurácia do modelo é de aproximadamente 69,41%, e as métricas de precisão, recall e F1-score variam para cada classe, como podemos ver a classe 0 possui uma precisão de acertos de 78%, um recall de 66% e um f1 score de 71%, já a classe 1 possui uma precisão de acertos de 64%, recall de 98% e f1 score de 78%, a classe 2 possui uma precisão de acertos de 81%, um recall de 30% e f1 score de 44%.

Como a acurácia de modelo que estamos usando como referência foi de 42.27%, podemos concluir que o nosso modelo NB que possuiu um ótimo resultado tendo uma acurácia de 69,41%, mas ele ainda é inferior a acurácia do modelo LR que foi de 77,43%.

Após o treinamento dos nossos modelos, vamos fazer usar uma técnica para avaliar o desempenho dos nossos modelos, que é nomeado por *Cross-Validation* (Validação Cruzada). Essa abordagem ajuda a fornecer uma estimativa mais robusta do desempenho dos modelos, evitando problemas como *overfitting* ou *underfitting*. Como é ilustrado nas figuras 34 e 35. Primeiro criamos uma função para imprimir os resultados do método, como é mostrado na figura 33.

Figura 33 – Criação da função para imprimir resultados do método de Cross-Validation

```
def imprimir_resultados(results):
    media = results['test_score'].mean()
    print(f"Acurácia média na validação cruzada: {media * 100:.2f}%")
    desvio_padrao = results['test_score'].std()
    print("Accuracy com cross validation = [%.2f, %.2f]" % ((media - 2 * desvio_padrao)*100, (media + 2 * desvio_padrao) * 100))
```

Fonte: (Autor)

Primeiro estamos calculando a média da acurácia a partir dos resultados da validação cruzada. Os resultados são acessados usando a chave `"test_score"` e a média é calculada usando o método `"mean()"`. Após o calculo da média e sua impressão, estamos calculando o desvio padrão da acurácia dos resultados da validação cruzada, utilizando o método `"std()"`. E logo depois estamos imprimindo a faixa de confiança de 95% para a acurácia, caculada como média mais ou menos 2 vezes o desvio padrão, formatada com duas casas decimais.

Figura 34 – Aplicando a função no método Cross-Validation

```
# Aplicando o metodo de validação cruzada - Cross Validation

modelo_lr = LogisticRegression()
cv = KFold(n_splits = 10, shuffle= True)
results = cross_validate(modelo_lr, xv_train, y_train, cv = cv, return_train_score=False)
imprimir_resultados(results)

✓ 24.6s

C:\Users\Thiago\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
C:\Users\Thiago\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(

C:\Users\Thiago\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
```

Fonte: (Autor)

Figura 35 – Continuação dos dos resultados da aplicação da função no método Cross-Validation

```
C:\Users\Thiago\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
...
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings.
Acurácia média na validação cruzada: 77.41%
Accuracy com cross validation = [75.22, 79.61]
C:\Users\Thiago\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(
```

Fonte: (Autor)

Nele estamos primeiro criando um modelo de regressão logística, após a criação da instância do modelo, criamos um objeto “KFold” com 10 splits para a validação cruzada, o parâmetro “shuffle=True” indica que os dados serão embaralhados antes de cada divisão, para não viciar os modelos com um padrão de dados. Depois aplicamos a validação cruzada e imprimimos os resultados, utilizando a função “cross_validate” para realizar a validação cruzada do modelo de regressão logística. Os parâmetros incluem o modelo “modelo_lr”, os dados de treinamento “xv_train, y_train”, o objeto KFold para controle da validação cruzada “cv=cv”, e “return_train_score=False” indica que não estamos interessados nos resultados de treinamento. Logo depois chamamos a nossa função de impressão dos resultados para exibir as métricas calculadas a partir dos resultados da validação cruzada.

6. Interpretação e apresentação dos Resultados

Agora vamos apresentar os resultados obtidos com base na aplicação dos modelos de machine learning desenvolvidos para as nossas classificações. Primeiro fazendo um web scrapping, que é uma prática para extrair informações ou dados de sites da web, vamos extrair da web comentários/avaliações de usuários sobre um determinado produto e fazer os tratamentos dos textos para aplicarmos em nosso modelo de machine learning para uma avaliação final do modelo. Fazemos essas extração usando a biblioteca “BeautifulSoup()” e “requests”, sendo que uma é biblioteca projetada para facilitar a extração de dados de documentos HTML e XML e outra é para fazer requisições HTTP, simplificando a tarefa de enviar requisições para APIs, páginas da web e outros serviços de Internet, respectivamente. Como é mostrado na figura 36, primeiro definimos uma url, que é a url do site que estamos querendo fazer a requisição para termos acesso aos dados. Logo depois, estamos fazendo a requisição a url que foi definida anteriormente, utilizando o código “requests.get(url)”, que está sendo enviado uma solicitação HTTP do tipo “GET”, como é mostrado na figura 37.

Figura 36 – Definição da url do site que é acessado

```
url = 'https://www.producthunt.com/products/chatgpt-1/reviews?rating=0&order=LATEST'
```

Fonte: (Autor)

Figura 37 – Requisição à URL

```
page = requests.get(url)
```

Fonte: (Autor)

Agora, vamos utilizar a biblioteca “Beautiful Soap” para criar um objeto de análise a partir do conteúdo HTML obtido na resposta da requisição HTTP., como é mostrado na figura 38.

Figura 38 – Criando um objeto de análise pelo conteúdo HTML

```
soap = BeautifulSoup(page.text, 'html')
```

Fonte:(Autor)

Como é mostrado na figura 39, vamos encontrar as tags que contêm as avaliações dos usuários no site escolhido, as tags serão encontradas no Inspeccionar do Google Chrome, clicando com o botão direito do Mouse em cima da parte de avaliações do site definido. Assim, pedimos para ele localizar todas as tags “div” e a classe “styles_htmlText__iftLe styles_format__k3_8m italic-default color-lighter-grey fontSize-18 fontWeight-400 styles_overallExperience__x7Gqf”, que é onde está as avaliações no site. Após localizamos as avaliações no site, vamos exportar os textos de todas as avaliações, como é mostrado na figura 40. Utilizamos o método “get_text()” para extrair o texto de uma tag HTML, ele retorna todo o texto contido dentro da tag, incluindo o texto de suas tags filhas, e utilizamos uma list comprehension, para iterar sobre cada elemento da lista “comment_tags”, que são as tags “div” com a classe especificada, e para cada uma dessas tags, extraí o texto usando “get_text()”. Depois, criamos um dataframe para colocarmos os textos das avaliações que foram exportadas anteriormente em uma coluna, como é mostrado na figura 41.

Figura 39 – Encontrando o conteúdo do site

```
# Encontrar as tags que contêm os comentários
# Ajuste isso com base na estrutura HTML da página
comment_tags = soap.find_all('div', class_='styles_htmlText__iftLe styles_format__k3_8m italic-default color-lighter-grey fontSize-18 fontWeight-400 styles_overallExperience__x7Gqf')
```

Fonte: (Autor)

Figura 40 – Exportando os textos das avaliações

```
# Extrair o texto dos comentários
comments = [comment.get_text() for comment in comment_tags]
```

Fonte: (Autor)

Figura 41 – Criação do DataFrame com os textos das avaliações

```
# Criar um DataFrame pandas
df_teste = pd.DataFrame({'Comments': comments})
df_teste
```

✓ 0.0s

	Comments
0	Dear J e t h a c k s and Team, My husband Jo...
1	Tool for using every day. Must have.
2	Since its launch, ChatGPT has taken the tech w...
3	I think ChatGPT is a good tools for content wr...
4	Thanks for building such an amazing tool!
5	Chat Gpt really revolutionize the AI industry
6	nice project i like this chatgpt
7	Has been a great help to me as a web developer...
8	Very convenient! So easy for todays different...
9	excellent chat, answers many questions, but ve...

Fonte: (Autor)

Agora, podemos aplicar os modelos treinados e testados no dataframe gerado com os textos das avaliações, antes iremos aplicar a nossa função de “conversao_de_texto” nos textos do dataframe e vamos transforma-los em números vetorizando os textos, para podermos aplicar em nossos modelos. Como é mostrado na figura 42 e 43. Primeiro criamos uma função para prever a polaridade de comentários, esta função recebe um comentário como entrada, realiza a conversão de texto, vetoriza o comentário usando o vetorizador (vectorizer), e então faz previsões usando quatro modelos diferentes (Logistic Regression, Random Forest Classifier, Decision Tree, Naive Bayes). E ainda, retorna os resultados como uma ‘pd.Series’ contendo as previsões para cada modelo. Após a criação da função de prever a popularidade de comentários, vamos aplica-lo em nosso dataframe, resultandi em um novo dataframe chamado “df_resultado”. Depois, vamos concatenar o dataframe resultante ao original e vamos gravar em “df_teste”. Agora, podemos exibir o dataframe resultante, exibindo as 10 primeiras linhas, mostrando os comentários originais e as previsões de popularidade feitas por cada modelo.

Figura 42 – Aplicação dos nossos modelos e criação das colunas no dataframe

```
# Função para converter a polaridade numérica em rótulo
def output_label(n):
    if n == 0:
        return "Positivo"
    elif n == 1:
        return "Negativo"
    elif n == 2:
        return "Neutro"

# Função para prever polaridade
def prever_polaridade(comentario):
    df_teste['Comments'] = df_teste['Comments'].apply(conversao_de_texto)
    comentario_vectorized = vectorizer.transform([comentario])

    # Substitua os próximos três comandos pelos seus modelos específicos
    pred_LR = lr.predict(comentario_vectorized)[0]
    pred_RFC = rfc.predict(comentario_vectorized)[0]
    pred_DT = dt.predict(comentario_vectorized)[0]
    pred_NB = nb.predict(comentario_vectorized)[0]

    return pd.Series({
        'LR_Prediction': output_label(pred_LR),
        'RFC_Prediction': output_label(pred_RFC),
        'DT_Prediction': output_label(pred_DT),
        'NB_Preiction': output_label(pred_NB)
    })

# Aplicar a função de previsão ao longo das linhas do DataFrame
df_resultado = df_teste['Comments'].apply(prever_polaridade)

# Concatenar o DataFrame resultante ao original
df_teste = pd.concat([df_teste, df_resultado], axis=1)

# Exibir o DataFrame resultante
df_teste.head(10)
```

Fonte: (Autor)

Figura 43 – Resultados das aplicações dos modelos

	Comments	LR_Prediction	RFC_Prediction	DT_Prediction	NB_Preiction
0	dear j e t h a c k s and team my husband jo...	Positivo	Positivo	Negativo	Negativo
1	tool for using every day must have	Positivo	Neutro	Neutro	Negativo
2	since its launch chatgpt has taken the tech w...	Positivo	Negativo	Positivo	Positivo
3	i think chatgpt is a good tools for content wr...	Positivo	Negativo	Negativo	Negativo
4	thanks for building such an amazing tool	Positivo	Positivo	Negativo	Positivo
5	chat gpt really revolutionize the ai industry	Positivo	Neutro	Neutro	Positivo
6	nice project i like this chatgpt	Positivo	Positivo	Positivo	Positivo
7	has been a great help to me as a web developer...	Positivo	Positivo	Positivo	Negativo
8	very convenient so easy for todays different...	Positivo	Negativo	Negativo	Negativo
9	excellent chat answers many questions but ve...	Positivo	Neutro	Neutro	Negativo

Fonte: (Autor)

Após a execução dos procedimentos de limpeza de dados, padronização, comparação dos modelos baselines e balanceamento das classes constataram que nosso trabalho obteve um resultado satisfatório.

Em conclusão, ao analisar os modelos, observamos que o modelo que obteve o melhor desempenho na classificação das avaliações dos usuários se foi uma avaliação de forma positiva, negativa ou neutra, foi o modelo de regressão logística, alcançando uma taxa de acerto de 77,43%.

8. Links

Apresentação do trabalho: <https://youtu.be/Nk32Jiv0QfQ>

GitHub: <https://github.com/thiagobc98/Portfolio/tree/main/TCC>

REFERÊNCIAS

MITCHELL, T. Aprendizado de máquina. Nova York: McGrawhill, 1997.

BREIMAN, L. Random Forests, Machine Learning, Vol. 45, pp. 5 – 32, 2001.

GUANGLI, Nie et al. Credit card churn forecasting by logistic regression and decision tree. Expert Systems with Applications, v. 38, n. 12, p. 15273-15285, 2011. Disponível em: . Acesso em: 10 mai. 2023.

Tukey, J. W. (1977). Exploratory Data Analysis. Reading: Addison-Wesley.

VERBRAKEN, Thomas; VERBEKE, Wouter; BAESENS, Bart. Profit optimizing customer churn prediction with Bayesian network classifiers. Intelligent Data Analysis, v. 18, n. 1, p. 3-24, 2014. Disponível em: . Acesso em: 10 mai. 2023.

GUANGLI, Nie et al. Credit card churn forecasting by logistic regression and decision tree. Expert Systems with Applications, v. 38, n. 12, p. 15273-15285, 2011. Disponível em: . Acesso em: 10 mai. 2023.

MÜLLER, A. C.;GUIDO, S. Introduction to Machine Learning with Python. O'Reilly Media, Inc. 2017. https://doi.org/10.1007/978-3-030-36826-5_10. Acesso em: 10 mai. 2023.

Tukey, J. W. (1962). The future of data analysis. The Annals of Mathematical Statistics, 33, 1–67.

MARTÍNEZ, Estela. Machine learning algorithms for the prediction of non-metallic inclusions in steel wires for tire reinforcement. 2019.

APÊNDICE

```

import pandas as pd
import numpy as np
import sys
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import KFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix
import re
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import requests

print("Versão do Python:")
print(sys.version)

df = pd.read_csv(r'/content/sentiment-emotion-
labelled_Dell_tweets.csv')

df.head()

df.columns

Index(['Unnamed: 0', 'Datetime', 'Tweet Id', 'Text', 'Username', 'sentiment',
      'sentiment_score', 'emotion', 'emotion_score'],
      dtype='object')

df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24970 entries, 0 to 24969
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            24970 non-null  int64

```

```

1   Datetime          24970 non-null object
2   Tweet Id          24970 non-null int64
3   Text              24970 non-null object
4   Username          24970 non-null object
5   sentiment          24970 non-null object
6   sentiment_score    24970 non-null float64
7   emotion           24970 non-null object
8   emotion_score      24970 non-null float64
dtypes: float64(2), int64(2), object(5)

```

```
memory usage: 1.7+ MB
```

```
df.shape
```

```
(24970, 9)
```

```
df.isna().sum()
```

```

Unnamed: 0      0
Datetime        0
Tweet Id        0
Text            0
Username        0
sentiment        0
sentiment_score  0
emotion         0
emotion_score    0
dtype: int64

```

```
df = df.drop_duplicates()
```

```

df.rename(columns={
    'Unnamed: 0': 'id',
    'Datetime': 'data_horario',
    'Text': 'texto',
    'Username': 'usuario',
    'sentiment': 'sentimento',
    'sentiment_score': 'sentimento_score',
    'emotion': 'emoji',
    'emotion_score': 'emoji_score'
}, inplace=True)

```

```
df = df.drop(['data_horario', 'Tweet Id', 'usuario', 'sentimen-
to_score', 'emoji_score'], axis=1)
```

```
df['sentimento'] = df['sentimento'].replace({'positive': 0, 'negative':
1, 'neutral': 2})
```

```
df = df.sample(frac = 1)
```

df

	id	texto	sentimento	emoji
11441	11441	@shannonrwatts @HP @Oracle @AmericanAir @South...	1	anger
21389	21389	@ecomericcarlson @Dell @Apple Specs? Good For ...	0	anticipation
13514	13514	@valentyn_bez @Dell oh, nice to know!	0	joy
15075	15075	@GamecockDave69 @Dell Def don't let your kids ...	1	anger
18323	18323	@_ChrisAlbertyn @Dell @DellXPS Dells response ...	1	anger
...
4394	4394	@Dell_IN @DellTechIndia @DellCares @Dell \n\nl...	2	anger
21305	21305	@ballsletics @iota @ClimateCHECK @goldstandard...	1	anger
2107	2107	@cocolowco @Dell I'm going to make sure this t...	1	anger
218	218	Is this a knock off? Or a misprint? I CANNOT f...	1	anger
13253	13253	@intelcanada @Dell The Intel Evo Laptop makes ...	0	joy

24970 rows × 4 columns

```
def conversao_de_texto(texto):
    texto = texto.lower()
    texto = re.sub('\[.*?\]', '', texto)
    texto = re.sub("\W", " ", texto)
    texto = re.sub('https?://\S+|www\.\S+', '', texto)
    texto = re.sub('<.*?>+', '', texto)
    texto = re.sub('[%s]' % re.escape(string.punctuation), '', texto)
    texto = re.sub('\n', '', texto)
    texto = re.sub('\w*\d\w*', '', texto)

    return texto
```

```
df['texto'] = df['texto'].apply(conversao_de_texto)
df.head()
```

	id	texto	sentimento	emoji
11441	11441	shannonrwatts hp oracle americanair south...	1	anger
21389	21389	ecomericcarlson dell apple specs good for ...	0	anticipation
13514	13514	valentynbez dell oh nice to know	0	joy
15075	15075	dell def don t let your kids watch that gar...	1	anger
18323	18323	chrisalbertyn dell dellxps dell response t...	1	anger

```
for column in df.columns:
    print(f'Coluna {column}: {df[column].unique()}')
    print('-----' * 10)
```

```
Coluna id: [11441 21389 13514 ... 2107 218 13253]
```

```
-----
Coluna texto: [' shannonrwatts hp oracle americanair southwestair
wholefoods dell heb the ceos and corporate leaders and politicians
send their kids to private school they don t care '
' ecomericcarlson dell apple specs good for photoshop and all '
' valentynbez dell oh nice to know ' ...
' cocolowco dell i m going to make sure this tweet goes viral so
dell has a bad quarter and loses sales worst computer maker of all ti-
me they take a bunch of third party parts and throw them together and
none of them work '
'is this a knock off or a misprint i cannot find anything but a dell
venue tablet help me out dell dell wtf mislabeled newproduct pc
pcgaming pcrepair windows microsoft microsoftwindows repair ap-
plerepair righttorepair tecktok techtoker sysadmin https t co '
' intelcanada dell the intel evo laptop makes video chatting the best
and easiest way for moms to connect over long distances ']
```

```
-----
Coluna sentimento: [1 0 2]
```

```
-----
Coluna emoji: ['anger' 'anticipation' 'joy' 'sadness' 'disgust' 'opti-
mism' 'fear'
'surprise']
```

```
# Calculando as contagens de sentimentos
contagem_sentimento_0 = (df['sentimento'] == 0).sum()
contagem_sentimento_1 = (df['sentimento'] == 1).sum()
contagem_sentimento_2 = (df['sentimento'] == 2).sum()

# Criando os rótulos das colunas
sentimentos = ['Positivo', 'Negativo', 'Neutro']

# Criando as contagens
```

```
contagens = [contagem_sentimento_0, contagem_sentimento_1, contagem_sentimento_2]
```

```
# Criando o gráfico de barras
plt.figure(figsize=(8, 6))
plt.bar(sentimentos, contagens, color=['green', 'red', 'blue'])
plt.ylabel('Quantidade')
plt.title('Contagem de Sentimentos')
plt.show()
```

```
X = df['texto'].values
Y = df['sentimento'].values
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.25, stratify=Y, random_state=50)
print("Treinamos com {} elementos e testaremos com {} elementos".format(len(x_train), len(x_test)))
```

Treinamos com 18727 elementos e testaremos com 6243 elementos

```
vectorizer = TfidfVectorizer()
xv_train = vectorizer.fit_transform(x_train)
xv_test = vectorizer.transform(x_test)
```

```
lr = LogisticRegression()
lr.fit(xv_train, y_train)
```

```
previsao_lr = lr.predict(xv_test)
acuracia = accuracy_score(y_test, previsao_lr)
f'Acurácia: {acuracia * 100: .2f} %'
```

Acurácia: 77.14 %

```
print(classification_report(y_test, previsao_lr))
```

	precision	recall	f1-score	support	
0		0.78	0.75	0.76	1842
1		0.82	0.87	0.85	2639
2		0.68	0.64	0.66	1762
accuracy				0.77	6243
macro avg		0.76	0.75	0.76	6243
weighted avg		0.77	0.77	0.77	6243

```
# Para testar se as acurácia estão boas ou ruim --- Usando como baseline
```



```
dummy = DummyClassifier()
dummy.fit(xv_train, y_train)
previsoes_dummy = dummy.predict(x_test)

acuracia_dummy = accuracy_score(y_test, previsoes_dummy) * 100
print("A acurácia do dummy foi %.2f%%" % acuracia_dummy)
A acurácia do dummy foi 42.27%
```

```
dt = DecisionTreeClassifier()
dt.fit(xv_train, y_train)
```

```
previsao_dt = dt.predict(xv_test)
```

```
acuracia = accuracy_score(y_test, previsao_dt)
f'Acurácia: {acuracia * 100: .2f} %'
```

Acurácia: 61.99 %

```
print(classification_report(y_test, previsao_dt))
```

precision	recall	f1-score	support		
	0	0.62	0.60	0.61	1842
	1	0.68	0.71	0.70	2639
	2	0.52	0.50	0.51	1762
accuracy				0.62	6243
macro avg		0.61	0.60	0.61	6243
weighted avg		0.62	0.62	0.62	6243

```
# Criando uma instancia do classificador de Floresta Aleatória
rfc = RandomForestClassifier(random_state=0)
```

```
# Treinando o modelo de Randon Forest
rfc.fit(xv_train, y_train)
```

```
# Fazendo previsões com base nos dados treinados
# Usando as características dos dados de teste
previsao_rfc = rfc.predict(xv_test)
```

```
acuracia = accuracy_score(y_test, previsao_rfc)
f'Acurácia: {acuracia * 100: .2f} %'
```

```
# O algoritmo classificador de floresta aleatória acerta 99% das pre-
visões
```

Acurácia: 73.17 %

```
print(classification_report(y_test, previsao_rfc))
```

precision	recall	f1-score	support		
	0	0.79	0.66	0.72	1842
	1	0.72	0.90	0.80	2639
	2	0.69	0.55	0.61	1762
accuracy				0.73	6243
macro avg		0.73	0.70	0.71	6243
weighted avg		0.73	0.73	0.72	6243

```
# Criando uma instância do modelo Naive Bayes
nb = MultinomialNB()
```

```
# 'X_train' são os recursos e 'y_train' são os rótulos
nb.fit(xv_train, y_train)
```

```
# Fazendo previsões no conjunto de teste
previsao_nb = nb.predict(xv_test)

# Avalie o desempenho do modelo
acuracia = accuracy_score(y_test, previsao_nb)
f'Acurácia: {acuracia * 100: .2f} %'
```

Acurácia: 68.88 %

```
print(classification_report(y_test, previsao_nb))
```

precision	recall	f1-score	support		
	0	0.79	0.64	0.71	1842
	1	0.63	0.98	0.77	2639
	2	0.82	0.30	0.44	1762
accuracy				0.69	6243
macro avg		0.75	0.64	0.64	6243
weighted avg		0.73	0.69	0.66	6243

```
def imprimir_resultados(results):
```

```

media = results['test_score'].mean()
print(f"Acurácia média na validação cruzada: {media * 100:.2f}%")
desvio_padrao = results['test_score'].std()
print("Accuracy com cross validation = [%.2f, %.2f]" % ((media - 2
* desvio_padrao)*100, (media + 2 * desvio_padrao) * 100))

# Aplicando o metodo de validação cruzada - Cross Validation

modelo_lr = LogisticRegression()
cv = KFold(n_splits = 10, shuffle= True)
results = cross_validate(modelo_lr, xv_train, y_train, cv = cv, re-
turn_train_score=False)
imprimir_resultados(results)

```

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
Acurácia média na validação cruzada: 77.99%
```

```
Accuracy com cross validation = [75.52, 80.47]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

```
n_iter_i = _check_optimize_result(
```

```
url = 'https://www.producthunt.com/products/chatgpt-1/reviews?rating=0&order=LATEST'
```

```

page = requests.get(url)

soap = BeautifulSoup(page.text, 'html')

# Encontrar as tags que contêm os comentários
# Ajuste isso com base na estrutura HTML da página
comment_tags = soap.find_all('div', class_='styles_htmlText__iftLe
styles_format__k3_8m italic-default color-lighter-grey fontSize-18
fontWeight-400 styles_overallExperience__x7Gqf') # Substitua 'div' e
'comment-class' pelos valores reais

# Extrair o texto dos comentários
comments = [comment.get_text() for comment in comment_tags]

# Criar um DataFrame pandas
df_teste = pd.DataFrame({'Comments': comments})
df_teste

```

Comments

- 0 Dear J e t h a c k s and Team, My husband Jo...
- 1 Tool for using every day. Must have.
- 2 Since its launch, ChatGPT has taken the tech w...
- 3 I think ChatGPT is a good tools for content wr...
- 4 Thanks for building such an amazing tool!
- 5 Chat Gpt really revolutionize the AI industry
- 6 nice project i like this chatgpt
- 7 Has been a great help to me as a web developer...
- 8 Very convenient! So easy for todays different...
- 9 excellent chat, answers many questions, but ve...

```

# Função para converter a polaridade numérica em rótulo
def output_label(n):
    if n == 0:
        return "Positivo"
    elif n == 1:

```

```

        return "Negativo"
    elif n == 2:
        return "Neutro"

# Função para prever polaridade
def prever_polaridade(comentario):
    df_teste['Comments'] =
df_teste['Comments'].apply(conversao_de_texto)
    comentario_vectorized = vectorizer.transform([comentario])

    # Substitua os próximos três comandos pelos seus modelos específicos
    pred_LR = lr.predict(comentario_vectorized)[0]
    pred_RFC = rfc.predict(comentario_vectorized)[0]
    pred_DT = dt.predict(comentario_vectorized)[0]
    pred_NB = nb.predict(comentario_vectorized)[0]

    return pd.Series({
        'LR_Prediction': output_label(pred_LR),
        'RFC_Prediction': output_label(pred_RFC),
        'DT_Prediction': output_label(pred_DT),
        'NB_Prediction': output_label(pred_NB)
    })

# Aplicar a função de previsão ao longo das linhas do DataFrame
df_resultado = df_teste['Comments'].apply(prever_polaridade)

# Concatenar o DataFrame resultante ao original
df_teste = pd.concat([df_teste, df_resultado], axis=1)

# Exibir o DataFrame resultante
df_teste.head(10)

```

Comments	LR_Prediction	RFC_Prediction	DT_Prediction	NB_Preiction	
0	it s one of the best innovations of this decade	Positivo	Positivo	Positivo	Positivo
1	fantastic thing boost my productivity	Positivo	Neutro	Neutro	Positivo
2	significantly accelera-tes the development proc...	Neutro	Neutro	Neutro	Neutro
3	dear j e t h a c k s and team my hus-band jo...	Positivo	Negativo	Positivo	Negativo

Comments	LR_Prediction	RFC_Prediction	DT_Prediction	NB_Preiction	
4	tool for using every day must have	Positivo	Neutro	Neutro	Negativo
5	since its launch chatgpt has taken the tech w...	Positivo	Positivo	Positivo	Positivo
6	i think chatgpt is a good tools for content wr...	Positivo	Negativo	Negativo	Negativo
7	thanks for building such an amazing tool	Positivo	Positivo	Positivo	Positivo
8	chat gpt really revolutionize the ai industry	Positivo	Neutro	Neutro	Positivo
9	nice project i like this chatgpt	Positivo	Positivo	Positivo	Positivo