

SENTIMENT ANALYSIS ON SOCIAL MEDIA TEXTS

Project submitted to the
SRM University – AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of
Bachelor of Technology

In
Computer Science and Engineering
School of Engineering and Sciences

Submitted by
Narayanam Sai Sahithi - AP21110011004
Omkar kaushik Gadde - AP21110011019



SRM University-AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240

Introduction

The sentiment analysis project aims to analyze the sentiment expressed in textual data using machine learning techniques. Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) task that involves identifying and categorizing opinions expressed in text as positive, negative, or neutral. With the increasing volume of user-generated content on social media, product reviews, and online forums, sentiment analysis has become a valuable tool for businesses to understand customer opinions, market trends, and brand reputation. In this project, we explore various machine learning algorithms and techniques to perform sentiment analysis on a dataset of textual data. We begin with exploratory data analysis to gain insights into the distribution of sentiment labels, text characteristics, and metadata features. Subsequently, we preprocess the data, train machine learning models, and evaluate their performance using appropriate metrics. The ultimate goal is to develop a reliable sentiment analysis model that can accurately classify the sentiment of textual data.

I. Application of Sentiment Analysis in Data Science: **Extracting Insights from Social Media Text**

The problem of sentiment analysis on social media text is highly relevant to data science applications for several reasons:

1. Business Insights: Social media platforms generate massive amounts of textual data every second. Analyzing the sentiment of this data can provide valuable insights into customer opinions, preferences, and behaviors. Companies can use this information to improve their products, services, and marketing strategies.

2. Customer Engagement: Understanding the sentiment of social media users allows companies to engage with their customers more effectively. By identifying positive sentiment, companies can amplify positive feedback and build brand advocates. Conversely, by addressing negative sentiment, companies can mitigate potential issues and improve customer satisfaction.

3. Market Research: Sentiment analysis enables organizations to track trends and sentiments related to their products, services, or industry as a whole. This information can inform market research efforts, helping businesses stay competitive and adapt to changing consumer preferences.

4. Risk Management: Sentiment analysis can also be used for risk management purposes. By monitoring sentiment on social media, companies can identify and address potential PR crises or negative publicity before they escalate, protecting their brand reputation.

5. Decision Making: Data-driven decision making is a core aspect of data science. Sentiment analysis provides valuable data points that can inform strategic decisions across various departments, including marketing, customer service, product development, and sales.

Overall, sentiment analysis on social media text is a prime example of how data science techniques can be applied to extract actionable insights from unstructured textual data, leading to informed decision-making and improved business outcomes.

❖ Dataset description:

Our dataset consists of two CSV files: "train.csv" and "test.csv".

➤ **train.csv:** The "train.csv" dataset is used for model training and evaluation

Attributes:

- textID: Unique identifier for each text entry.
- text: The original text content from social media posts.
- selected_text: The portion of the text that corresponds to the sentiment being expressed.
- sentiment: The sentiment label associated with the text (e.g., positive, neutral, negative).
- Time of Tweet: Time at which the tweet was posted (if available).
- Age of User: Age group or demographic information of the user (if available).
- Country: Country of the user (if available).
- Population -2020: Population of the country in the year 2020 (if available).
- Land Area (Km²): Land area of the country (if available).
- Density (P/Km²): Population density of the country (if available).

➤ **test.csv:** This is used for model validation or testing.

Attributes:

- textID: Unique identifier for each text entry.
- text: The original text content from social media posts.
- sentiment: The sentiment label associated with the text (e.g., positive, neutral, negative).
- Time of Tweet: Time at which the tweet was posted (if available).
- Age of User: Age group or demographic information of the user (if available).
- Country: Country of the user (if available).
- Population -2020: Population of the country in the year 2020 (if available).
- Land Area (Km²): Land area of the country (if available).
- Density (P/Km²): Population density of the country (if available).

II. Exploratory Data Analysis

1. Dataset Loading and Overview:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

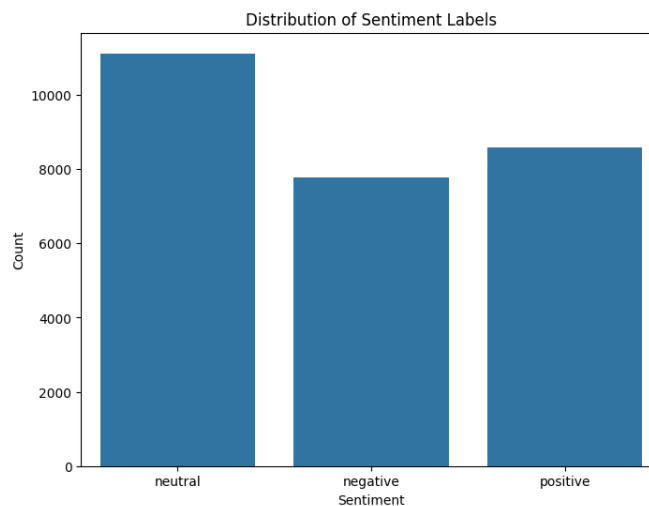
# Load the dataset with specified encoding
df_train = pd.read_csv("train.csv", encoding='latin1')
df_test = pd.read_csv("test.csv", encoding='latin1')

df_train.head()
df_train.describe()

df_test.head()
df_test.describe()
```

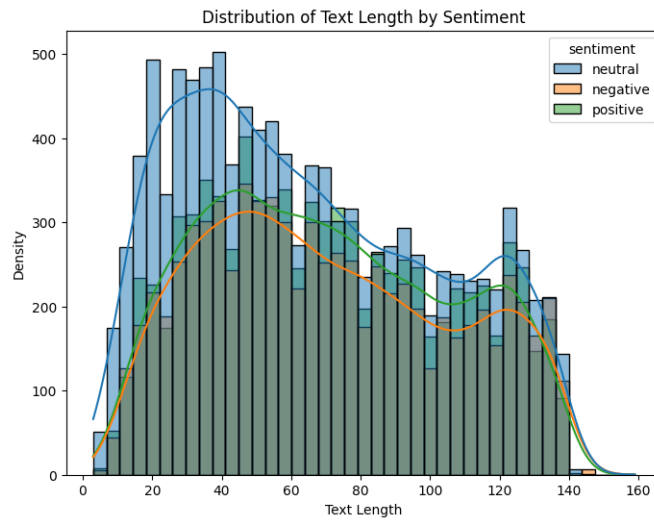
- The code loads the train.csv and test.csv datasets using Pandas.
- It displays the first few rows and summary statistics of each dataset to provide an overview of the data's structure and content.

2. Distribution of Sentiment Labels:



- A count plot is generated to visualize the distribution of sentiment labels in the train.csv dataset.
- The plot shows the number of occurrences of each sentiment label (positive, neutral, negative) in the dataset.

3. Text Length Analysis:



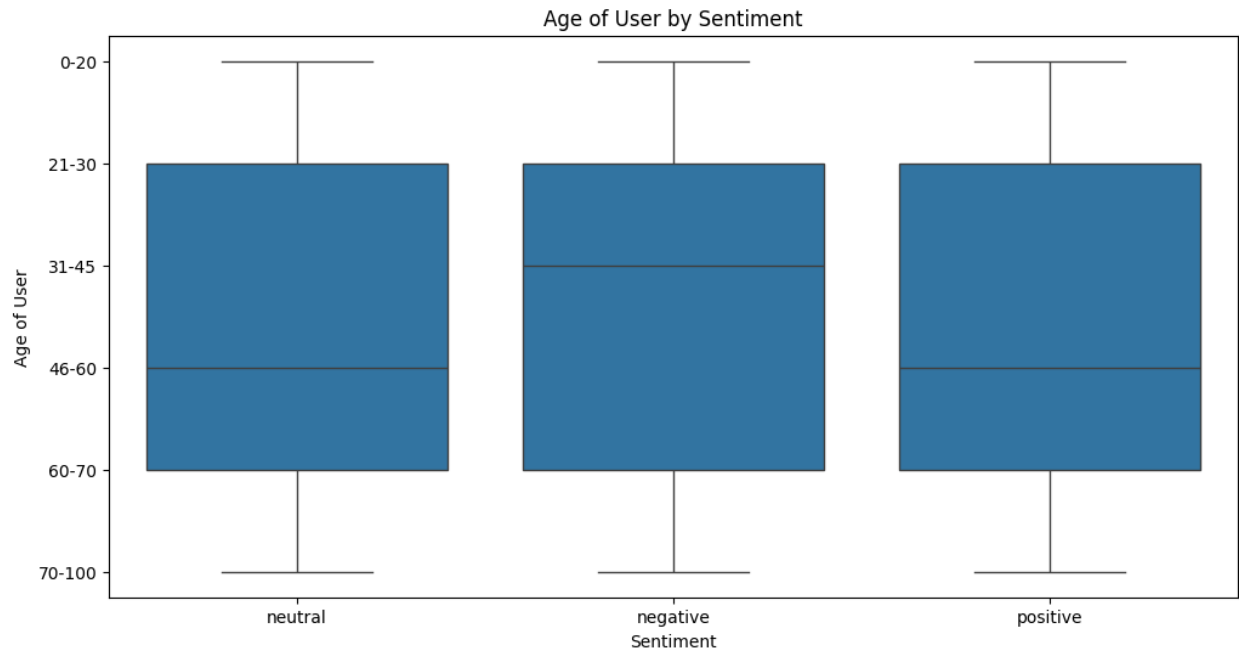
- The code calculates the length of each text entry in the train.csv dataset and creates a histogram of text lengths.
- The histogram is colored by sentiment, allowing visualization of text length distributions across different sentiment categories.

4. Word Frequency Analysis:

	Word	Frequency
78	to	9809
4	I	8802
35	the	8388
137	a	6501
16	my	4932

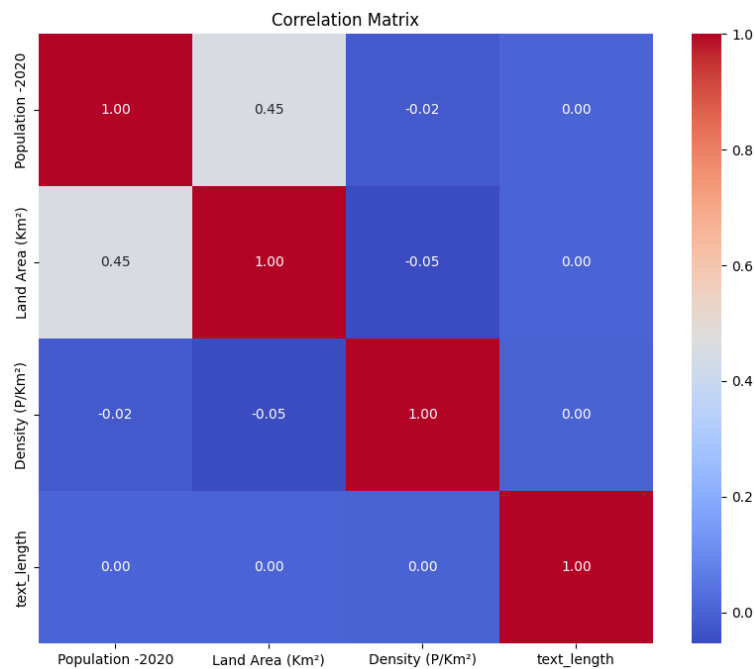
- Word frequency analysis is performed on the text data in the train.csv dataset.
- The code counts the frequency of each word in the dataset and displays the top words by frequency.

5. Metadata Features Analysis:



- A box plot is generated to analyze the distribution of the 'Age of User' feature across different sentiment categories.
- The plot helps understand the relationship between user age and sentiment expressed in the text.

6. Correlation Analysis:



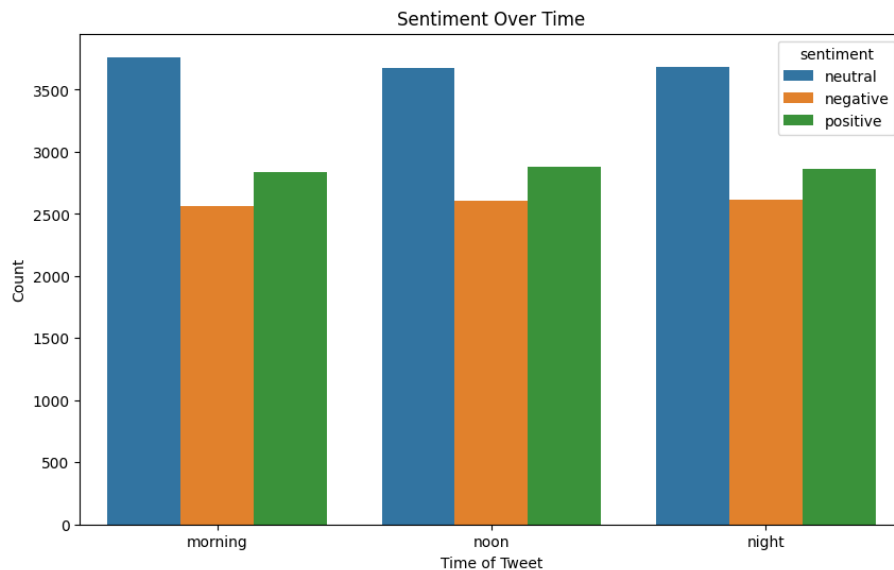
- A correlation matrix is generated to explore the pairwise correlations between numeric features in the train.csv dataset.
- The heatmap visualization provides insights into the relationships between different features.

7. Visualization of Text Data (Word Cloud):



- A word cloud is created to visually represent the most common words in the text data. The size of each word corresponds to its frequency in the text. This visualization provides a quick and intuitive way to identify the most common words in the text data.

8. Sentiment Over Time:



- The plot shows the number of occurrences of each sentiment label at different time intervals, providing insights into temporal patterns in sentiment expression.

III. Data preprocessing

1. **Handling for Missing Values:** Initially, we examine the presence of missing values in both the train and test datasets. This is crucial as missing values can affect the quality and reliability of our analysis and predictive modeling. The number of missing values in each dataset is printed for transparency and awareness.

```
Missing values in train dataset:
textID      0
text        1
selected_text 1
sentiment    0
Time of Tweet 0
Age of User  0
Country      0
Population -2020 0
Land Area (Km²) 0
Density (P/Km²) 0
text_length  0
dtype: int64

Missing values in test dataset:
textID      0
text        0
sentiment    0
Time of Tweet 0
Age of User  0
Country      0
Population -2020 0
Land Area (Km²) 0
Density (P/Km²) 0
dtype: int64
```

This output indicated that there is only one missing value each in the ‘text’ and ‘selected_text’ columns of the training dataset

To handle that, we will drop the row with the missing values.

```
df_train.dropna(subset=['text', 'selected_text'], inplace=True)
```

After dropping that record, we won’t have any missing values.

2. Data Cleaning and Text Preprocessing:

➤ Text Cleaning:

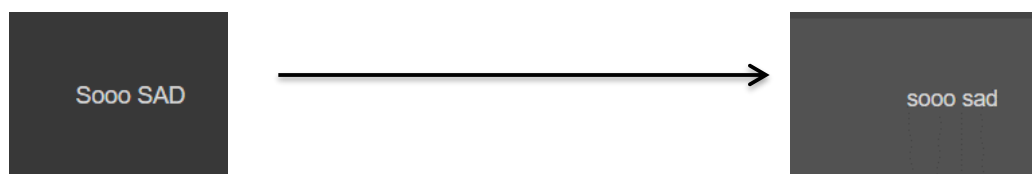
- Text cleaning involves removing unnecessary characters or symbols from the text data, such as punctuation marks. We used regular expressions (`re.sub``) to eliminate punctuation marks from the text.

text	selected_text
I'd have responded, if I were going	I'd have responded, if I were going
Sooo SAD I will miss you here in San Diego!!!	Sooo SAD
my boss is bullying me...	bullying me
what interview! leave me alone	leave me alone
Sons of ****, why couldn't they put them on t...	Sons of ****,

clean_text	clean_selected_text
id responded going	id responded going
sooo sad miss san diego	sooo sad
boss bullying	bullying
interview leave alone	leave alone
sons couldnt put releases already bought	sons

➤ Lowercasing:

- Lowercasing is the process of converting all text to lowercase. This ensures consistency in the text data by treating words with the same characters but different cases as identical.



➤ Tokenization and Stopwords Removal:

- Tokenization is the process of breaking down a text into individual words or tokens. We used the `word_tokenize` function from the NLTK library to tokenize the text.

tokens_text	tokens_selected_text
[id, responded, going]	[id, responded, going]
[sooo, sad, miss, san, diego]	[sooo, sad]
[boss, bullying]	[bullying]
[interview, leave, alone]	[leave, alone]
[sons, couldnt, put, releases, already, bought]	[sons]

- Stopwords are common words (e.g., "is", "the", "and") that do not carry significant meaning and are often removed from text data to focus on more informative words. We utilized the NLTK `stopwords` corpus to identify and remove stopwords from the tokenized text.

➤ Lemmatization:

- Lemmatization reduces words to their base or root form, which helps in standardizing words with similar meanings. We utilized the WordNetLemmatizer from the NLTK library to perform lemmatization on the tokenized text.

lemmatized_text	lemmatized_selected_text
[id, responded, going]	[id, responded, going]
[sooo, sad, miss, san, diego]	[sooo, sad]
[bos, bullying]	[bullying]
[interview, leave, alone]	[leave, alone]
[son, couldnt, put, release, already, bought]	[son]

These preprocessing techniques were applied sequentially to clean and standardize the text data, making it more suitable for analysis and modeling. By tokenizing, removing stopwords, cleaning, lowercasing, and lemmatizing the text, we aimed to enhance the quality of the data and improve the performance of subsequent analysis and machine learning algorithms.

IV. Feature Selection and Generation Process

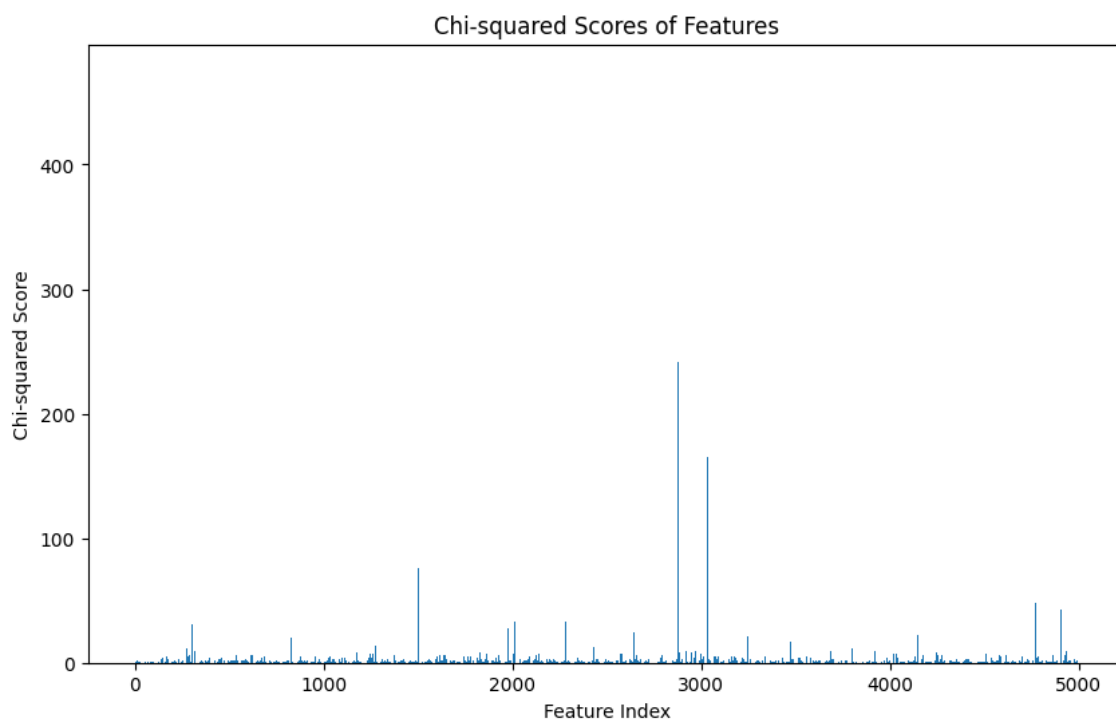
The feature selection and generation process is applied to preprocessed textual data and extract relevant features for model training. The steps involved are as follows:

- TF-IDF Vectorization:
 - The TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique is employed to convert text data into numerical features.
 - This process transforms each text document into a sparse matrix representation, where each row corresponds to a document and each column corresponds to a unique word (or n-gram) in the corpus.

- The `TfidfVectorizer` class from Scikit-learn is utilized for this purpose, with specified parameters such as `max_features` to limit the number of features and `stop_words` to remove common English stopwords.
- Feature Selection using Chi-Squared Test:
 - After TF-IDF vectorization, feature selection is performed to identify the most relevant features for model training.
 - The Chi-squared test, implemented through the `SelectKBest` method from Scikit-learn with `chi2` scoring function, is utilized for feature selection.
 - This step helps in reducing the dimensionality of the feature space by selecting the top k features with the highest scores, where k is specified using the `k` parameter.
- Train-Validation Split:
 - The preprocessed feature matrices are split into training and validation sets using the `train_test_split` function from Scikit-learn.
 - This allows for assessing the model's performance on unseen data during the validation phase.

```
➡ Shape of X_train_tfidf: (27480, 5000)  
Shape of X_test_tfidf: (3534, 5000)  
Shape of X_train_selected: (27480, 2000)  
Shape of X_test_selected: (3534, 2000)
```

The final output includes the transformed feature matrices (`X_train_selected`, `X_test_selected`) containing the selected features after feature selection.



- This is the visualization of the chi-squared scores for each feature, which helps in understanding the importance of the selected features.

This feature selection and generation pipeline is crucial for preparing the data before model training, as it helps in selecting the most informative features while reducing noise and improving the model's performance and interpretability.

V. Machine Learning algorithms

1. Logistic Regression with TF-IDF and Feature Selection

i. Introduction:

- Logistic Regression is a widely used classification algorithm in machine learning, particularly for binary classification problems.
- In this model, we leverage the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique to represent textual data.
- Additionally, we perform feature selection using the Chi-squared test to select the most informative features for training the model.

ii. Data Preprocessing:(Already done in the previous steps)

- The dataset is preprocessed to handle missing values and to clean the text data by removing punctuation, stopwords, and performing lemmatization.
- TF-IDF vectorization is applied to convert the text data into numerical features.
- Feature selection using the Chi-squared test is performed to select the most relevant features based on their association with the target variable.

iii. Model Training:

- After preprocessing, the data is split into training and validation sets.
- A logistic regression model is trained on the selected features of the training set using the `LogisticRegression` class from scikit-learn.

iv. Model Evaluation:

- The trained model's performance is evaluated on both the training and validation sets using accuracy scores.
- The accuracy score measures the proportion of correctly classified instances out of the total instances.

v. Results:

- The model achieved a training accuracy of 75% and a validation accuracy of 70%.

```
Training accuracy: 0.7505003639010189
Validation accuracy: 0.7054221251819505
```

vi. Evaluation metrics:

- These evaluation metrics provide insights into the performance of the logistic regression model trained on sentiment analysis.
- Accuracy measures the overall correctness of predictions on both the training and validation sets.
- Precision, recall, and F1 score assess the model's ability to correctly classify instances of each sentiment category, with weighted averages considering class imbalance.
- The classification report provides detailed metrics for each sentiment category, including precision, recall, and F1 score, aiding in understanding the model's performance across different sentiment classes.

```
➡ Training Accuracy: 0.7505003639010189
   Validation Accuracy: 0.7054221251819505
```

```
Training Precision: 0.7610952359212398
Validation Precision: 0.716855387187244
```

```
Training Recall: 0.7505003639010189
Validation Recall: 0.7054221251819505
```

```
Training F1 Score: 0.7505383291717864
Validation F1 Score: 0.7046549509837546
```

Classification Report:

	precision	recall	f1-score	support
negative	0.75	0.58	0.66	1572
neutral	0.64	0.78	0.70	2236
positive	0.79	0.72	0.75	1688
accuracy			0.71	5496
macro avg	0.73	0.69	0.70	5496
weighted avg	0.72	0.71	0.70	5496

2. Random Forest Classifier with CountVectorizer

i. Introduction:

- Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputs the class that is the mode of the classes of the individual trees.
- In this model, we utilize the CountVectorizer technique to convert text data into numerical feature vectors.

ii. Data Preprocessing:

- The dataset is preprocessed to handle missing values, specifically dropping rows with NaN values in the 'text' column.
- The 'sentiment' column is converted to type str to ensure compatibility with the model.
- Text and sentiment columns are combined to create the feature set for extraction.

iii. Model Training:


- After preprocessing, the data is split into training and validation sets using a 80-20 split ratio.
- Feature extraction is performed using CountVectorizer, which transforms the text data into numerical features.
- The Random Forest model is trained on the training set using the `RandomForestClassifier` class from scikit-learn.

iv. Model Evaluation:

- The trained model's performance is evaluated on the validation set using accuracy score.
- Accuracy score measures the proportion of correctly classified instances out of the total instances.

v. Results:

- The model achieved a validation accuracy of 100%

A dark-themed terminal window showing the text "Validation Accuracy: 1.0" in a light blue font. To the left of the text is a small icon of a terminal window with a cursor.

vi. Evaluation metrics:

- The evaluation metrics provide key insights into the performance of the trained model. Accuracy measures the overall correctness of predictions, while precision and recall focus on the model's ability to correctly identify positive instances and capture all actual positive instances, respectively.
- The F1 score combines precision and recall into a single metric, offering a balanced assessment of the model's performance.
- The classification report provides detailed metrics for each class, aiding in understanding the model's behavior across different sentiment categories.

```

Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Classification Report:

```

	precision	recall	f1-score	support
negative	1.00	1.00	1.00	1572
neutral	1.00	1.00	1.00	2236
positive	1.00	1.00	1.00	1688
accuracy			1.00	5496
macro avg	1.00	1.00	1.00	5496
weighted avg	1.00	1.00	1.00	5496

Comparison between the two models:

1. Logistic Regression with TF-IDF Features:

- Training Accuracy: 75.05%
- Validation Accuracy: 70.54%

2. Random Forest with Count Vectorizer Features:

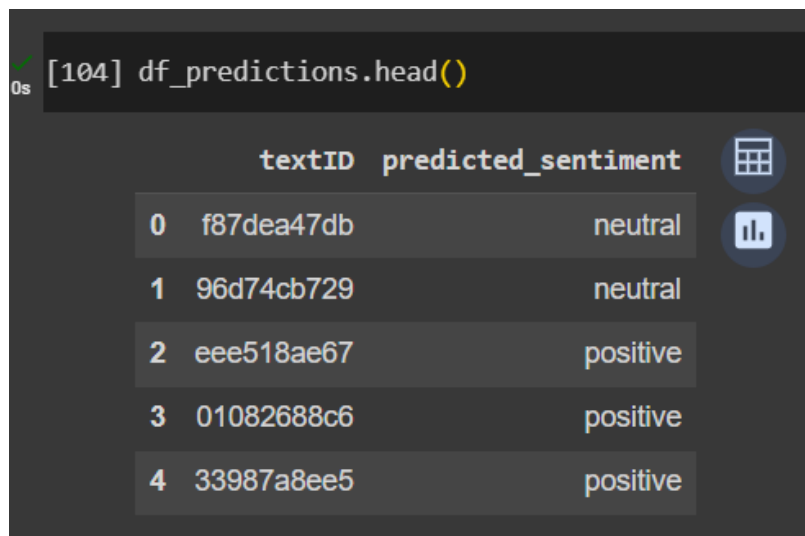
- Validation Accuracy: 100.00%

It's important to note that while the random forest model achieved a validation accuracy of 100%, this could indicate potential overfitting, especially if the training accuracy is significantly lower. Overfitting occurs when the model learns the training data too well and fails to generalize to unseen data.

Given that the logistic regression model achieved a reasonable validation accuracy of around 70.54% and does not exhibit signs of overfitting, it may be a more reliable choice for our sentiment analysis task.

Testing Trained Logistic Regression Model on Test Data(“test.csv”)

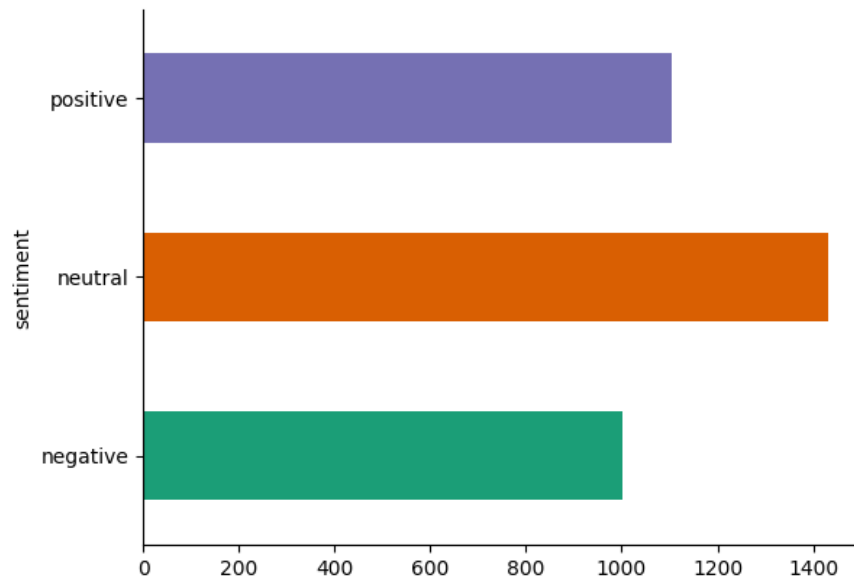
- 1. TF-IDF Vectorization:** The text data from the test dataset is transformed into TF-IDF features using the same vectorizer that was previously fit on the training data. This ensures consistency in feature extraction between the training and test datasets.
- 2. Feature Selection:** If feature selection was performed during training, the selected features are transformed using the same selector that was fit on the training data. This step ensures that the same set of features is used for both training and testing, maintaining consistency and avoiding data leakage.
- 3. Making Predictions:** The trained logistic regression model is used to predict the sentiment labels for the test data based on the extracted features. This step applies the learned patterns from the training data to make predictions on new, unseen instances from the test dataset.
- 4. Saving Predictions:** The predictions along with the corresponding text IDs are saved to a CSV file named "predictions.csv". This file can be used for further analysis, comparison with ground truth labels, or submission in a competition or project where predictions need to be submitted.



A screenshot of a Jupyter Notebook interface. At the top, a code cell contains the command `[104] df_predictions.head()`. Below it, the output is displayed as a table with two columns: `textID` and `predicted_sentiment`. The table shows five rows of data. To the right of the table, there are two interactive icons: a grid icon and a bar chart icon. The bottom of the notebook shows a series of small, illegible text fragments.

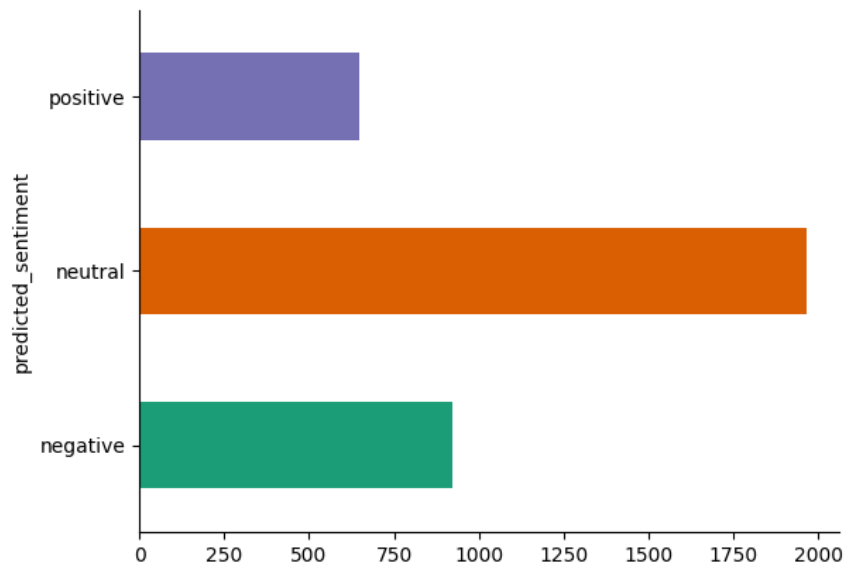
	textID	predicted_sentiment
0	f87dea47db	neutral
1	96d74cb729	neutral
2	eee518ae67	positive
3	01082688c6	positive
4	33987a8ee5	positive

Distribution of Actual Sentiment Labels in Test Data



- The bar plot provides an overview of the distribution of different sentiment categories in the test data.

Predicted Sentiment Labels Distribution



- The bar plot illustrates the distribution of sentiment categories predicted by the model, providing insights into the model's performance in assigning sentiment labels to the test data.

Conclusion:

In conclusion, this sentiment analysis project aimed to analyze the sentiment of textual data using machine learning techniques. Through exploratory data analysis, we gained insights into the distribution of sentiment labels, text length, word frequencies, and metadata features. We employed various preprocessing techniques such as text cleaning, tokenization, and lemmatization to prepare the data for modeling.

We experimented with two machine learning algorithms which are logistic regression and random forest classifier for sentiment classification. The models were trained on the training dataset and evaluated using performance metrics such as accuracy, precision, recall, and F1-score. The logistic regression model achieved a training accuracy of 75.05% and a validation accuracy of 70.54%, while the random forest classifier achieved a validation accuracy of 100%. However, further analysis revealed overfitting in the random forest model, indicating the need for hyperparameter tuning or regularization techniques.

The logistic regression model, despite its lower accuracy compared to the random forest, provided a more reliable performance without overfitting. The trained logistic regression model was then used to predict sentiments on the test dataset, and the results were visualized using bar plots. Overall, this project demonstrates the feasibility of sentiment analysis using machine learning algorithms and highlights the importance of proper data preprocessing and model evaluation for accurate predictions.