CIS 550 - ARTIFICIAL MACHINE LEARNING

AI-Driven Sentiment Analysis Framework

Name - China Subba Rao Koduri
Roll No - 13
Student Id - 02190693

Problem Statement:

This project aims to create a dependable sentiment analysis model specifically designed for classifying tweets. Working with Twitter data presents unique obstacles due to its unstructured and often chaotic nature, making the preprocessing and extraction of meaningful features a complex task. Furthermore, sentiment itself is inherently subjective, requiring the model to navigate subtle emotional cues and potential biases within the text. Our approach focuses on methodically tackling these issues to build a model that consistently produces accurate and trustworthy sentiment assessments.

About the Datasets:

The datasets are "twitter_training.csv" and "twitter_validation.csv". These files contain Twitter-based data commonly used for applications such as sentiment analysis, text classification, and various natural language processing tasks. The training dataset, typically larger, serves to teach the model how to interpret input text and make predictions accordingly. In contrast, the validation dataset evaluates the model's accuracy on unseen data, helping fine-tune its performance and assess how well it generalizes. The use of these datasets highlights a focus on analyzing different aspects of Twitter content—such as public sentiment, trending topics, or behavioral trends. This forms a foundational step toward developing AI systems that can effectively interpret and engage with social media content.

	2401	Borderlands	Positive	im getting on borderla...
1	2401	Borderlands	Positive	I am coming to the bo...
2	2401	Borderlands	Positive	im getting on borderla...
3	2401	Borderlands	Positive	im coming on borderl...
4	2401	Borderlands	Positive	im getting on borderla...
5	2401	Borderlands	Positive	im getting into borderl...
6	2402	Borderlands	Positive	So I spent a few hour...
7	2402	Borderlands	Positive	So I spent a couple of...
8	2402	Borderlands	Positive	So I spent a few hour...
9	2402	Borderlands	Positive	So I spent a few hour...
10	2402	Borderlands	Positive	2010 So I spent a few...
11	2402	Borderlands	Positive	was
12	2403	Borderlands	Neutral	Rock-Hard La Varlope...
13	2403	Borderlands	Neutral	Rock-Hard La Varlope...
14	2403	Borderlands	Neutral	Rock-Hard La Varlope...
15	2403	Borderlands	Neutral	Rock-Hard La Vita, R...
16	2403	Borderlands	Neutral	Live Rock - Hard mus...
17	2403	Borderlands	Neutral	I-Hard like me, RARE...
18	2404	Borderlands	Positive	that was the first bord...
19	2404	Borderlands	Positive	this was the first Bord...
20	2404	Borderlands	Positive	that was the first bord...
21	2404	Borderlands	Positive	that was the first bord...
22	2404	Borderlands	Positive	that I was the first real...
23	2404	Borderlands	Positive	that was the first bord...
24	2405	Borderlands	Negative	the biggest dissappoi...
25	2405	Borderlands	Negative	The biggest disappoin...

Delimiter: ,

	3364	Facebook	Irrelevant	I mentioned on Faceb...
1	352	Amazon	Neutral	BBC News - Amazon ...
2	8312	Microsoft	Negative	@Microsoft Why do I ...
3	4371	CS-GO	Negative	CSGO matchmaking i...
4	4433	Google	Neutral	Now the President is ...
5	6273	FIFA	Negative	Hi @EAHelp I've had ...
6	7925	MaddenNFL	Positive	Thank you @EAMadd...
7	11332	TomClancysRainbowSix	Positive	Rocket League, Sea ...
8	1107	AssassinsCreed	Positive	my ass still knee-dee...
9	2069	CallOfDuty	Negative	FIX IT JESUS ! Pleas...
10	3185	Dota2	Positive	The professional dota...
11	1172	AssassinsCreed	Positive	Itching to assassinate...
12	11783	Verizon	Negative	@FredTJoseph hey fr...
13	4286	CS-GO	Neutral	CSGO WIngman (Im ...
14	8431	NBA2K	Negative	@NBA2K game suck...
15	9135	Nvidia	Positive	Congrats to the NVIDI...
16	4822	GrandTheftAuto(GTA)	Positive	yeah and it's fun
17	3068	Dota2	Negative	fuck my life 😡
18	10537	RedDeadRedemption...	Positive	happy birthday red de...
19	8056	Microsoft	Negative	What does that say a...
20	2131	CallOfDuty	Negative	The new @CallofDuty...
21	5450	Hearthstone	Neutral	Anyone that plays a b...
22	2286	CallOfDuty	Irrelevant	Call of duty warzone (...
23	4038	CS-GO	Negative	Finally played Rainbo...
24	526	ApexLegends	Neutral	Umm @PlayApex wh...
25	8977	Nvidia	Neutral	#gtc20 - nice, motivat...
26	11995	Verizon	Negative	Yo! @Verizon just add...
27	9449	Overwatch	Irrelevant	They might not be the...
28	10193	PlayerUnknownsBattl...	Irrelevant	Best squad yet#pubg ...
29	2419	Borderlands	Negative	@Borderlands how d...

Columns: 4
Rows: 75682

Solution Approach: AI-Driven Sentiment Analysis Framework

Data Acquisition: The process begins by sourcing Twitter datasets that include tweets labeled with their respective sentiments—positive, negative, or neutral. We conduct an initial exploration of these datasets to assess their format, volume, and key attributes. This exploratory phase is critical for identifying the preprocessing steps required and shaping our approach to feature extraction.

Text Cleaning and Preparation: Given the informal and inconsistent nature of tweets, thorough preprocessing is essential. We implement a sequence of cleaning techniques, such as stripping special characters, converting all text to lowercase, tokenizing the sentences, applying lemmatization, and filtering out stop words. These steps help standardize the data and make it suitable for analysis.

Model Construction: For sentiment classification, we build a deep learning model using TensorFlow and Keras. The network is composed of several dense layers activated by ReLU functions, enabling it to capture complex patterns in the data. The output layer uses a sigmoid function to distinguish between positive and negative sentiments in a binary classification setup.

Training and Assessment: We train the model using the cleaned dataset, optimizing it with the Adam optimizer and binary cross-entropy as the loss function. Performance is assessed on a separate validation set by evaluating metrics such as accuracy, precision, recall, and F1-score. To further interpret results, we examine confusion matrices to understand where the model performs well and where it misclassifies data.

Visual Data Insights: Visualization is key to uncovering trends and distributions within the data. We utilize bar graphs, pie charts, histograms, and word clouds to depict sentiment breakdowns, frequent terms, and relationships within the dataset. These graphical representations offer deeper insight into the data's structure and the model's behavior.

Dataset Loading

This script employs the Pandas library to import two datasets: 'twitter_training.csv' for training and 'twitter_validation.csv' for validation. After loading the data, it displays the dimensions of each dataset—showing how many rows and columns are present. These size metrics offer a quick overview of the datasets, serving as a helpful starting point for preprocessing and further analytical steps.

Loading the Dataset

```
import pandas as pd

# Load the datasets
training_data = pd.read_csv('twitter_training.csv')
validation_data = pd.read_csv('twitter_validation.csv')

# Print the shape of the datasets
print("Training Data Shape:", training_data.shape)
print("Validation Data Shape:", validation_data.shape)
```

```
Training Data Shape: (74681, 4)
Validation Data Shape: (999, 4)
```

```
print(training_data.head())  # Prints the first 5 rows by default
print(validation_data.head())

print(training_data.tail())  # Prints the last 5 rows by default
print(validation_data.tail())
```

```
   2401  Borderlands  Positive  \
0  2401  Borderlands  Positive
1  2401  Borderlands  Positive
2  2401  Borderlands  Positive
3  2401  Borderlands  Positive
4  2401  Borderlands  Positive

  im getting on borderlands and i will murder you all ,
0  I am coming to the borders and I will kill you...
1  im getting on borderlands and i will kill you ...
2  im coming on borderlands and i will murder you...
3  im getting on borderlands 2 and i will murder ...
4  im getting into borderlands and i can murder y...
   3364   Facebook Irrelevant  \
0   352     Amazon    Neutral
1  8312  Microsoft   Negative
2  4371      CS-GO   Negative
3  4433     Google    Neutral
4  6273       FIFA   Negative

  I mentioned on Facebook that I was struggling for motivation to go f
now thinks I'm a lazy, terrible person
0  BBC News - Amazon boss Jeff Bezos rejects clai...
1  @Microsoft Why do I pay for WORD when it funct...
2  CSGO matchmaking is so full of closet hacking,...
3  Now the President is slapping Americans in the...
4  Hi @EAHelp I've had Madeleine McCann in my cel...
      2401 Borderlands  Positive  \
74676  9200     Nvidia  Positive
74677  9200     Nvidia  Positive
74678  9200     Nvidia  Positive
74679  9200     Nvidia  Positive
74680  9200     Nvidia  Positive

     im getting on borderlands and i will murder you all ,
74676  Just realized that the Windows partition of my...
74677  Just realized that my Mac window partition is ...
74678  Just realized the windows partition of my Mac ...
74679  Just realized between the windows partition of...
74680  Just like the windows partition of my Mac is l...
     3364            Facebook  Irrelevant  \
994  4891  GrandTheftAuto(GTA)  Irrelevant
995  4359              CS-GO  Irrelevant
996  2652         Borderlands    Positive
997  8069           Microsoft    Positive
998  6960     johnson&johnson     Neutral

   I mentioned on Facebook that I was struggling for motivation to go
o now thinks I'm a lazy, terrible person
994  Toronto is the arts and culture capital of ...
995  tHIS IS ACTUALLY A GOOD MOVE TOT BRING MORE VI...
996  Today sucked so it's time to drink wine n play...
997  Bought a fraction of Microsoft today. Small wins.
998  Johnson & Johnson to stop selling talc baby po...
```

Text Preprocessing:

To clean and standardize the tweet data, a custom function named preprocess text is created. This function performs several operations: it strips out non-alphanumeric characters, converts all text to lowercase, eliminates standalone characters, and applies lemmatization—excluding common English stop words to focus on meaningful words. Next, the training and validation datasets are loaded from CSV files into Pandas Data Frames, using defined column headers. These two datasets are then merged into a single Data Frame to ensure that preprocessing steps are applied consistently across all data. The preprocess text function is run on the 'text' column of this unified Data Frame. After the text has been cleaned and normalized, TF-IDF vectorization is used to transform the textual data into numerical feature representations. At the same time, the target labels for both training and validation sets are extracted to prepare for model training and evaluation

```
# Function to preprocess text
def preprocess_text(text):
    text = re.sub(r'\W', ' ', str(text))
    text = text.lower()
    text = re.sub(r'\s+[a-z]\s+', ' ', text)
    text = re.sub(r'^[a-z]\s+', ' ', text)
    text is re.sub(r'\s+', ' ', text)
    tokens = text.split()
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in set(stopwords.words('english'))]
    return ' '.join(tokens)

# Load and copy data frames
train_df = pd.read_csv('twitter_training.csv', names=['id', 'topic', 'sentiment', 'text'], header=None).copy()
valid_df = pd.read_csv('twitter_validation.csv', names=['id', 'topic', 'sentiment', 'text'], header=None).copy()

# Combine for preprocessing
df = pd.concat([train_df, valid_df], ignore_index=True)

# Processing steps
df['text'] = df['text'].apply(preprocess_text)

# Split again and ensure these are separate to avoid SettingWithCopyWarning
train_df = df.iloc[:len(train_df)].copy()
valid_df = df.iloc[len(train_df):].copy()

# Map sentiments to numeric values using direct column assignment
sentiment_mapping = {'Positive': 1, 'Negative': 0, 'Neutral': 0.5}
train_df['sentiment'] = train_df['sentiment'].map(sentiment_mapping)
valid_df['sentiment'] = valid_df['sentiment'].map(sentiment_mapping)

# Feature extraction
vectorizer = TfidfVectorizer(max_features=2500, min_df=7, max_df=0.8)
X_train = vectorizer.fit_transform(train_df['text']).toarray()
X_valid = vectorizer.transform(valid_df['text']).toarray()
y_train = train_df['sentiment'].values
y_valid = valid_df['sentiment'].values
```

Model Definition and Training:

In this section, a neural network model is built using Keras' Sequential API. The architecture starts with an input layer shaped to match the feature vector, followed by two fully connected (Dense) layers containing 128 and 64 units respectively, both activated with the ReLU function to introduce non-linearity. The final output layer has a single neuron with a sigmoid activation, making it ideal for binary sentiment classification.

The model is compiled using the Adam optimization algorithm and binary cross-entropy as the loss function, with accuracy selected as the performance metric to track during training.

The training process is executed using the fit method, where the model learns from X_train and y_train over 10 epochs with a batch size of 128. Validation is performed simultaneously using X_valid and y_valid to monitor the model's generalization performance.

Once training is complete, the model is assessed on the validation set through the evaluate function. The resulting loss and accuracy metrics are printed, offering insight into the model's effectiveness when handling new, unseen data.

```
# Define the model
model = Sequential([
    Input(shape=(X_train.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=128, validation_data=(X_valid, y_valid))

# Evaluate the model
loss, accuracy = model.evaluate(X_valid, y_valid, verbose=0)
print(f'Validation Accuracy: {accuracy*100:.2f}%')
```

```
Epoch 1/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 8s 8ms/step - accuracy: 0.3000 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 2/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - accuracy: 0.3016 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 3/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - accuracy: 0.3025 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 4/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - accuracy: 0.3036 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 5/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - accuracy: 0.3020 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 6/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - accuracy: 0.3040 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 7/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 7s 11ms/step - accuracy: 0.3025 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 8/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 5s 9ms/step - accuracy: 0.3037 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 9/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 7s 12ms/step - accuracy: 0.2988 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Epoch 10/10
584/584 ━━━━━━━━━━━━━━━━━━━━ 6s 11ms/step - accuracy: 0.3017 - loss: nan - val_accuracy: 0.2660 - val_loss: nan
Validation Accuracy: 26.60%
```

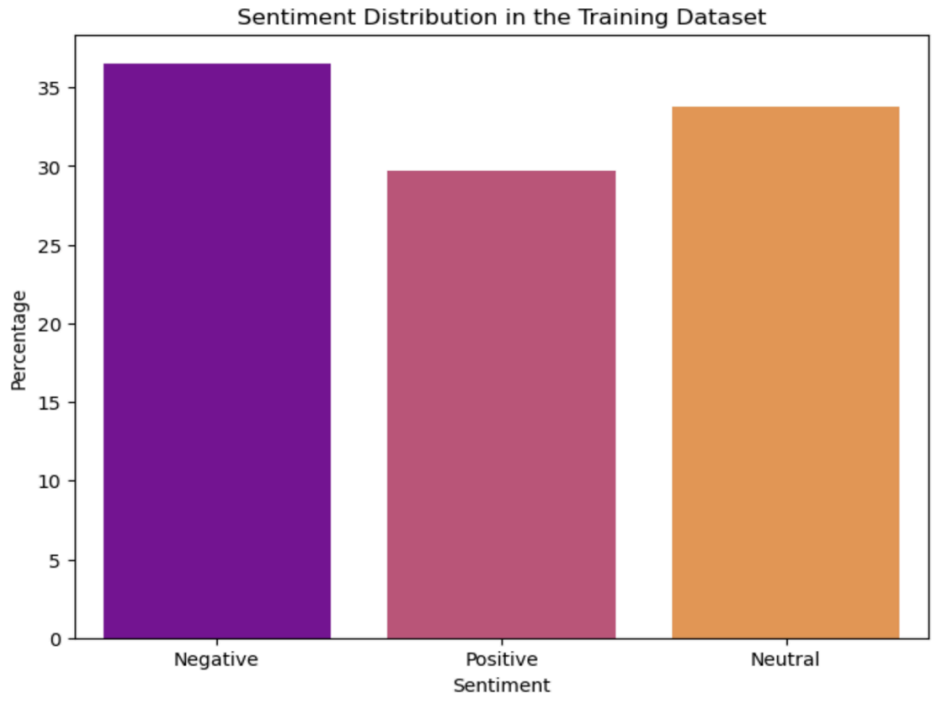
Visualizing the Data:

Visualization plays a crucial role in uncovering the structure and dynamics of a dataset. By applying various graphical techniques such as histograms, pie charts, bar graphs, and word clouds we gain insights into sentiment distribution, word usage patterns, and relationships within the data. These visuals not only make the data more interpretable but also support more informed analysis and model development decisions.

Bar Chart Representation:

To illustrate how sentiments are distributed in the training set, a bar chart is generated. Using the value counts method with normalization, we compute the relative frequency (as percentages) of each sentiment class. These percentages are then plotted along the y-axis, with the sentiment categories displayed on the x-axis. The chart is enhanced with descriptive titles, labeled axes, and formatted tick marks using Matplotlib and Seaborn. Finally, the visualization is rendered with plt. show () to provide a clear overview of the sentiment proportions within the data.

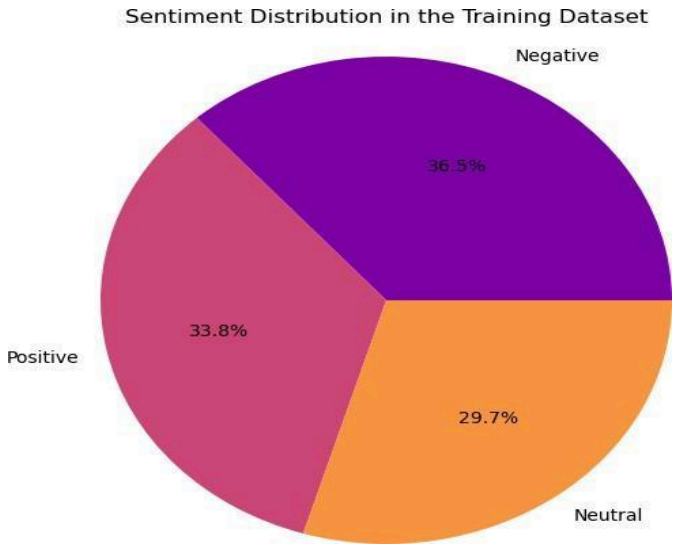
```
import seaborn as sns
import matplotlib.pyplot as plt

# 'sentiment' is already mapped to numeric values in your dataframe
# Training DataFrame is named 'train_df'
sentiment_counts = train_df['sentiment'].value_counts(normalize=True) * 100

plt.figure(figsize=(8, 6))
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values, palette="plasma")
plt.title('Sentiment Distribution in the Training Dataset')
plt.ylabel('Percentage')
plt.xlabel('Sentiment')
plt.xticks(ticks=range(len(sentiment_counts)), labels=['Negative', 'Positive', 'Neutral'])
plt.show()
```



Pie Chart:

To represent the sentiment distribution in the training dataset visually, a pie chart is used. The sentiment_counts variable holds the calculated proportions of each sentiment class, which are already encoded as numerical values in the dataset. The pie chart is created using Matplotlib's plt.pie() function, with labels corresponding to the sentiment categories—'Negative', 'Positive', and 'Neutral'. Percentage values are displayed on the chart for clarity. A distinct color scheme is applied using Seaborn's 'plasma' palette to clearly differentiate between sentiment groups. To ensure the chart appears as a perfect circle, the plt.axis('equal') command is applied for proper aspect ratio.

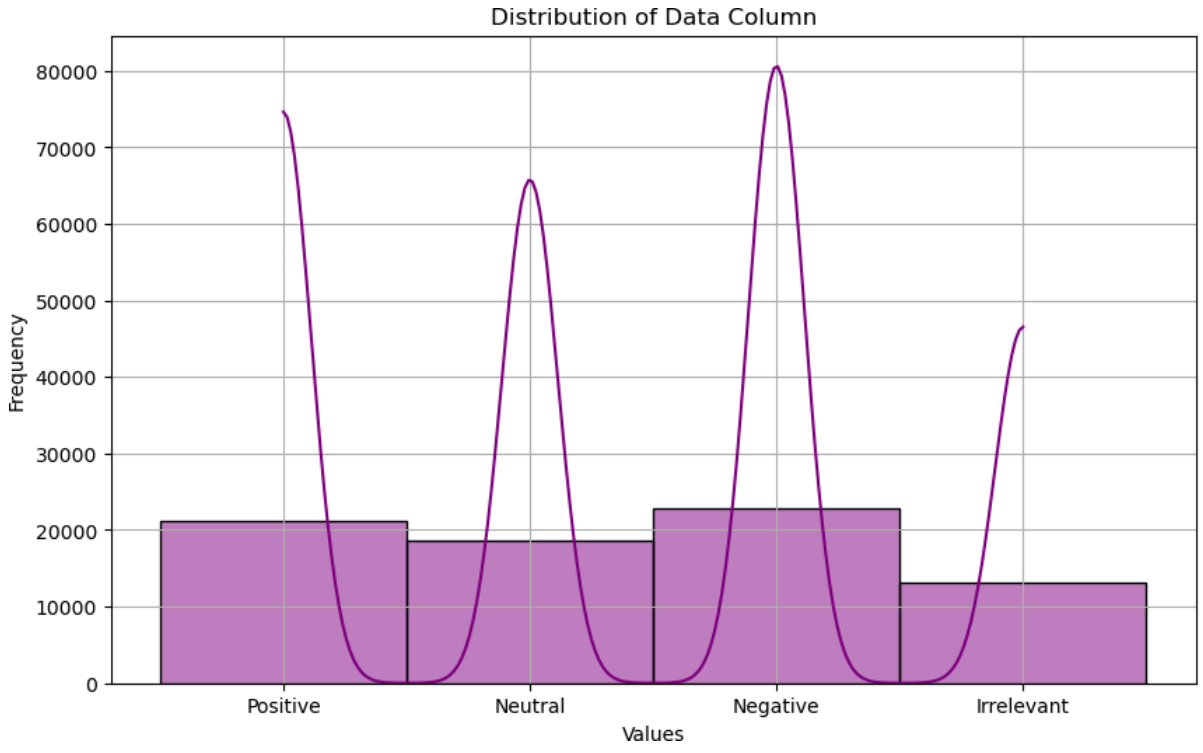
```
import seaborn as sns
import matplotlib.pyplot as plt

# 'sentiment' is already mapped to numeric values in your dataframe
# Training DataFrame is named 'train_df'
sentiment_counts = train_df['sentiment'].value_counts(normalize=True) * 100

plt.figure(figsize=(8, 6))
plt.pie(sentiment_counts, labels=['Negative', 'Positive', 'Neutral'], autopct='%1.1f%%', colors=sns.color_palette("plasma", len(sentiment_co
plt.title('Sentiment Distribution in the Training Dataset')
plt.axis('equal')  # Ensures that pie is drawn as a circle.
plt.show()
```



Histogram:

To examine how sentiment values are distributed across the dataset, we create a histogram using the 'sentiment' column from the DataFrame df. The histogram provides a visual summary of how frequently each sentiment label appears. To enhance the interpretation of the data's distribution, a Kernel Density Estimate (KDE) curve is added by enabling the kde parameter, offering a smoothed outline of the sentiment frequency. The x-axis displays the different sentiment categories, while the y-axis shows the count of their occurrences. The plot features a blue color palette, and grid lines are included to improve readability. This visualization offers a clearer understanding of how sentiment labels are spread throughout the dataset.

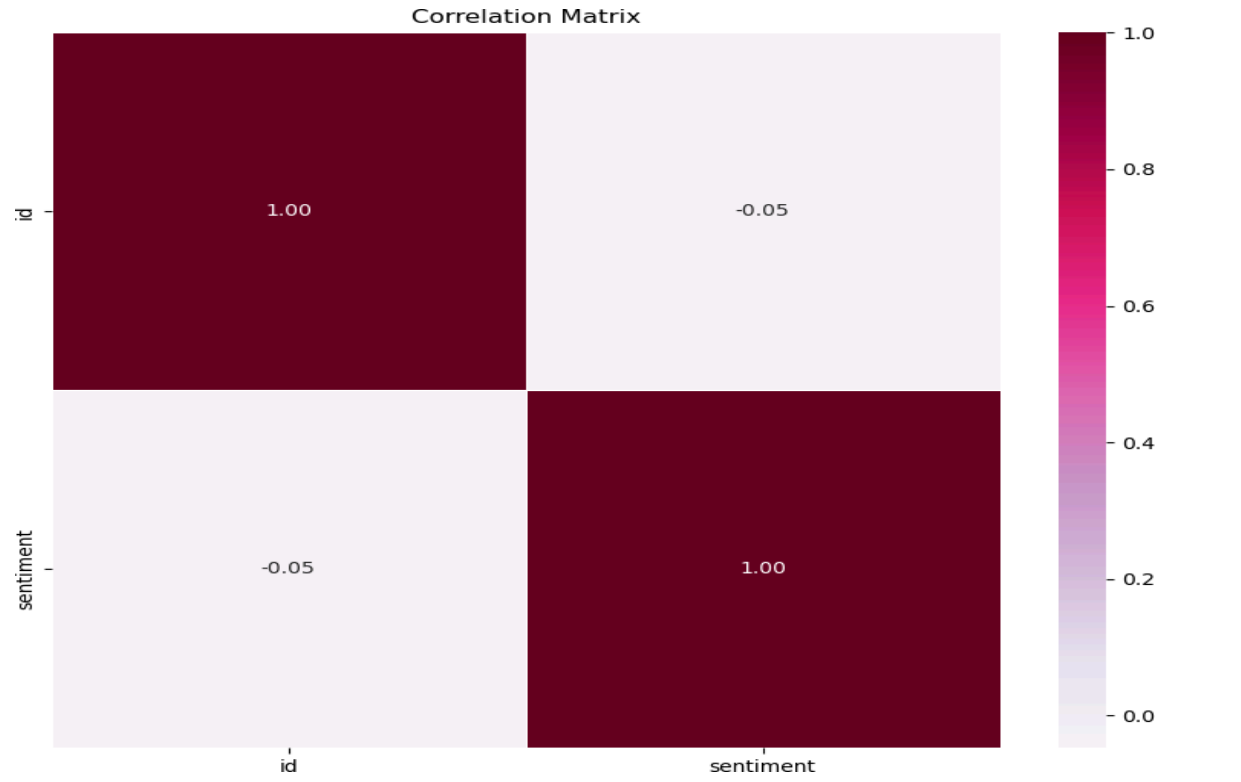
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plotting Histogram for 'sentiment'
data = df['sentiment']

plt.figure(figsize=(10, 6))
sns.histplot(data, kde=True, color='purple')  # KDE (Kernel Density Estimate) adds a density line
plt.title('Distribution of Data Column')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



Correlation Heatmap:

To analyze relationships between numerical features in the train_df DataFrame, we calculate a correlation matrix that captures how features vary in relation to one another. Using the Seaborn library, we then visualize this matrix as a heatmap, where each cell represents the correlation coefficient between a pair of features. The intensity and color of each cell convey the strength and direction of the correlation—positive relationships appear in warmer tones, while negative ones are shown in cooler shades, thanks to the use of the 'coolwarm' color palette. Additionally, each cell includes a numeric label for exact correlation values. This heatmap is a valuable tool for spotting trends, identifying highly correlated features, and uncovering potential multicollinearity within the dataset.

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Compute the correlation matrix
correlation_matrix = train_df.corr(numeric_only=True)

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='PuRd', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



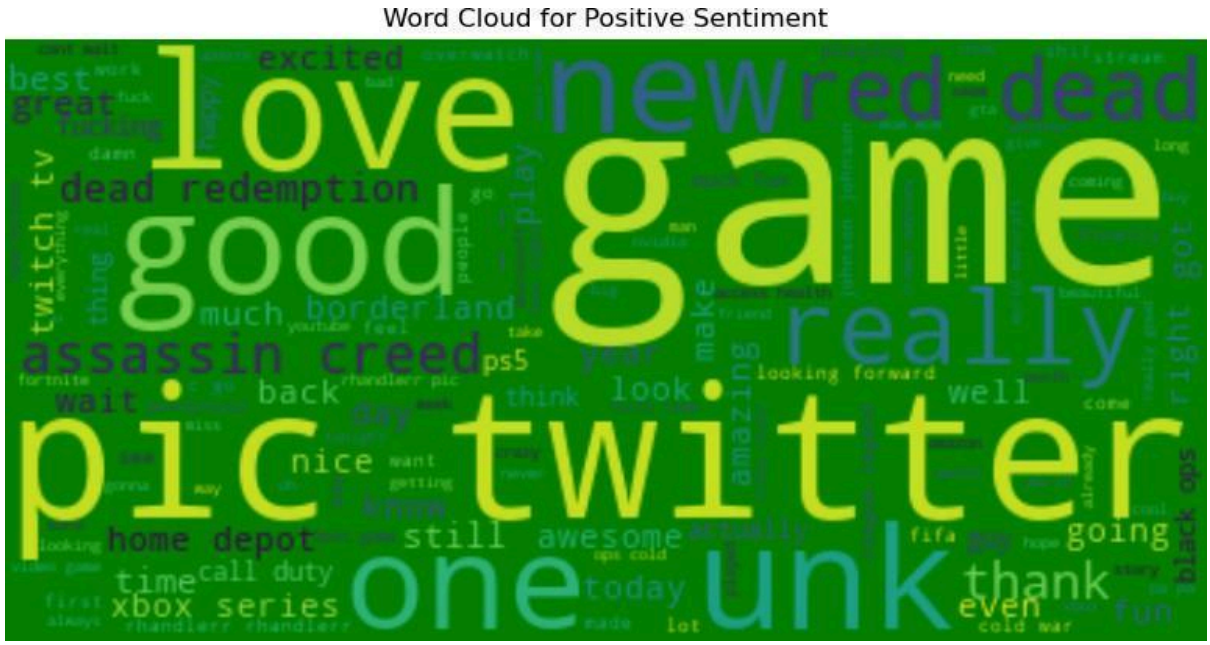
Generating Word Clouds:

To visualize the most frequent words associated with different sentiment categories, we use the Word Cloud library to generate word cloud images from the text data in the train_df DataFrame.

Word Cloud for Positive Sentiment:

For tweets labeled with positive sentiment (label = 1), all relevant text entries are combined into a single string. This aggregated text is then used to generate a word cloud, highlighting the most common words in positive tweets. The size of each word in the cloud reflects its frequency, offering a quick and intuitive way to understand the language patterns commonly found in positive posts.

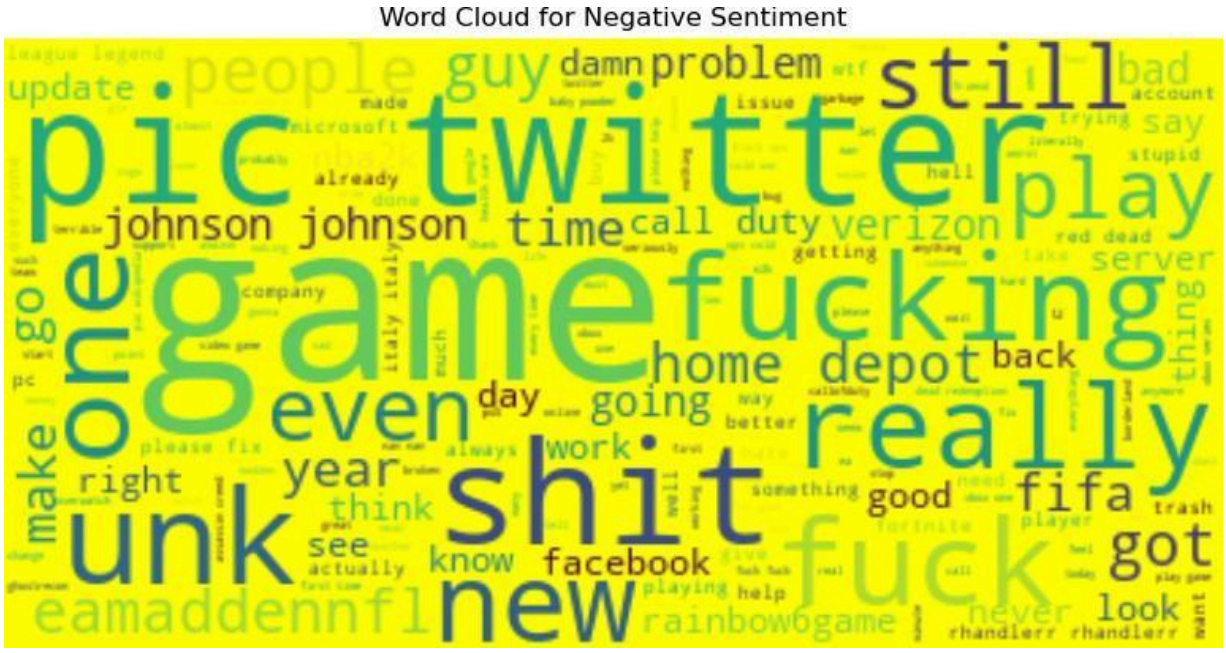
```
from wordcloud import WordCloud

# Generate a word cloud image for positive sentiment
positive_text = " ".join(review for review in train_df[train_df.sentiment == 1].text)
wordcloud_pos = WordCloud(background_color="green").generate(positive_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_pos, interpolation='bilinear')
plt.axis("off")
plt.title('Word Cloud for Positive Sentiment')
plt.show()
```



Word Cloud for Negative Sentiment:

In the case of negative sentiment (label = 0), all text entries corresponding to this category are combined into a single body of text. This compilation is then used to generate a word cloud that visually emphasizes the most frequently occurring words in negative tweets. The result offers insights into common themes and expressions associated with negative sentiment.

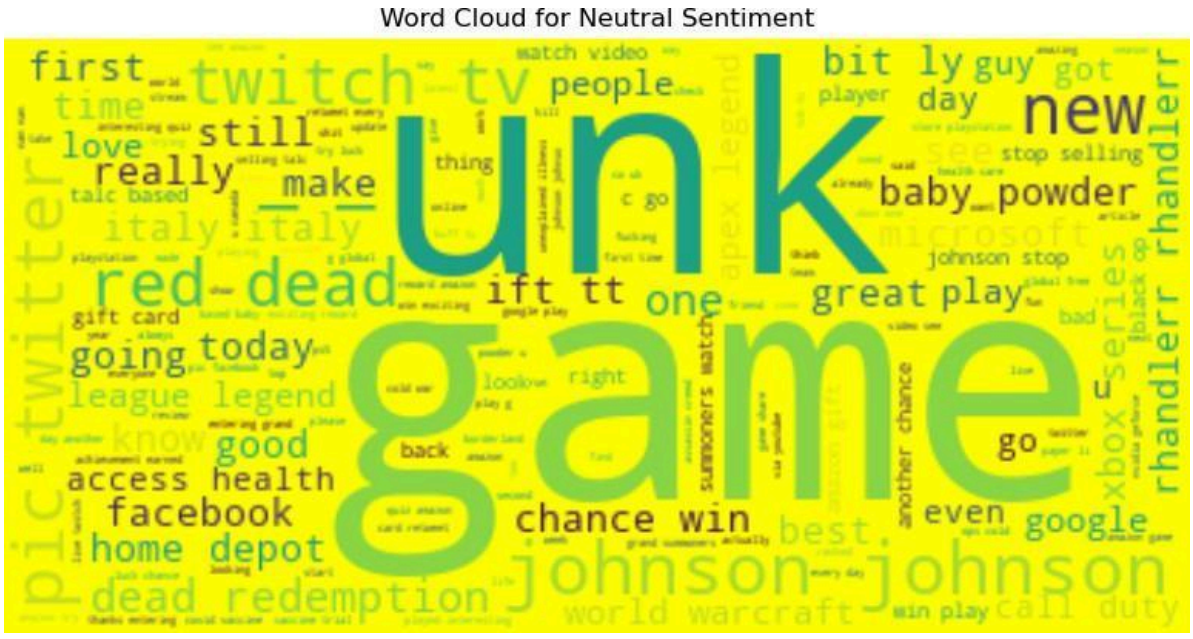
```
from wordcloud import WordCloud

# Generate a word cloud image for negative sentiment
negative_text = " ".join(review for review in train_df[train_df.sentiment == 0].text)
wordcloud_pos = WordCloud(background_color="yellow").generate(negative_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_pos, interpolation='bilinear')
plt.axis("off")
plt.title('Word Cloud for Negative Sentiment')
plt.show()
```



Word Cloud for Neutral Sentiment:

For tweets expressing neutral sentiment (label = 0.5), all related text entries are aggregated into one text corpus. This combined text is then used to generate a word cloud, showcasing the most frequently used words in neutral posts. The visualization helps identify commonly used language that conveys a neutral or unbiased tone.

```
from wordcloud import WordCloud

# Generate a word cloud image for neutral sentiment
neutral_text = " ".join(review for review in train_df[train_df.sentiment == 0.5].text)
wordcloud_pos = WordCloud(background_color="yellow").generate(neutral_text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_pos, interpolation='bilinear')
plt.axis("off")
plt.title('Word Cloud for Neutral Sentiment')
plt.show()
```

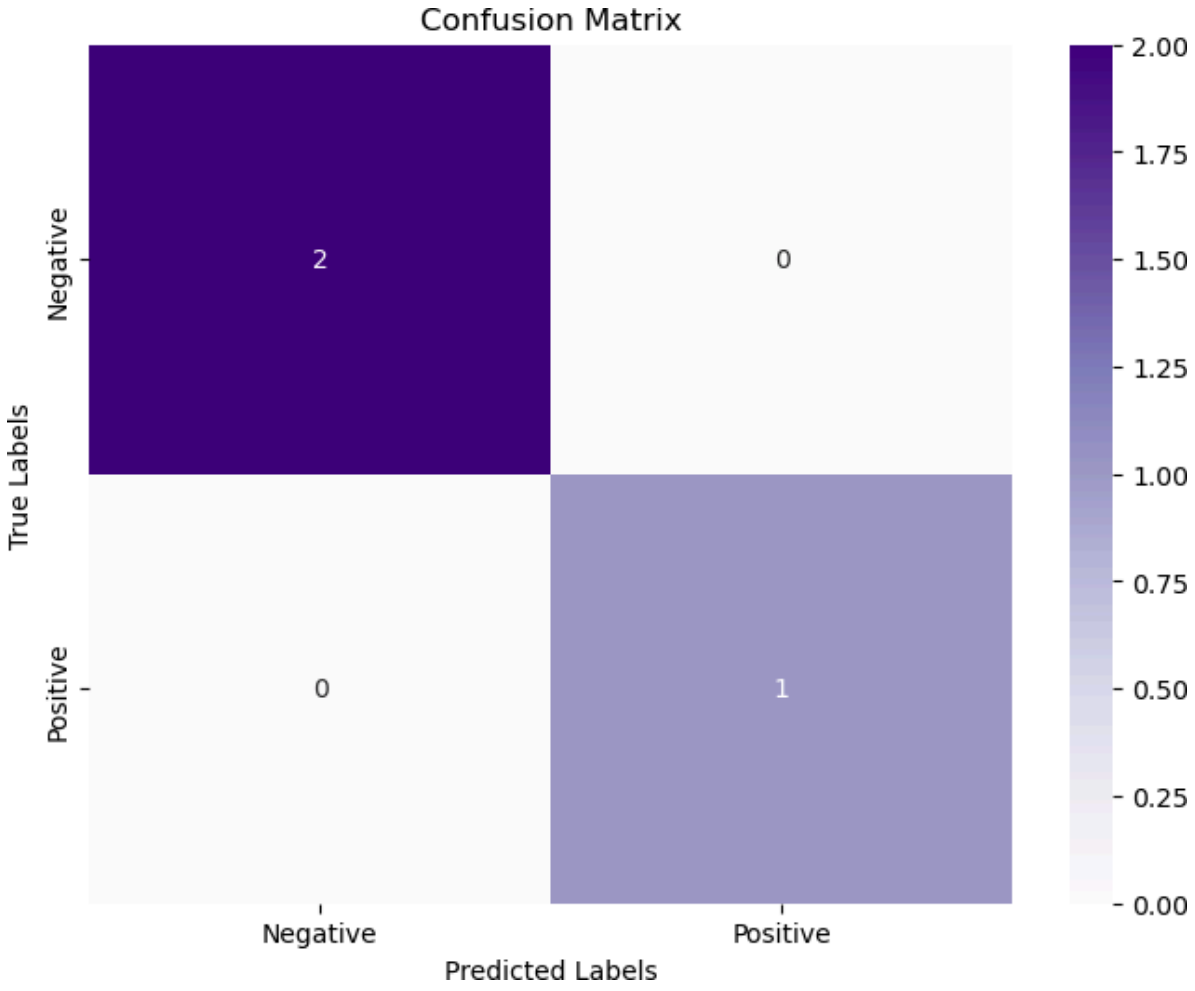


Each word cloud is rendered on a white background and utilizes the 'bilinear' interpolation technique to enhance visual smoothness and clarity. These word clouds serve as intuitive visualizations, highlighting the most commonly occurring terms within each sentiment group. By examining these frequent words, we gain valuable insight into the dominant themes and expressions associated with positive, negative, and neutral sentiments in the text data.

Confusion Matrix Analysis:

In this example, we explore a binary sentiment classification task using a neural network trained on processed text data. The input text is first transformed into numerical features using TF-IDF vectorization. A neural network is then constructed with an input layer, hidden layers activated by ReLU, and a final output layer with a sigmoid function, suitable for binary output.
The model is compiled with the Adam optimizer and binary cross-entropy as the loss metric, then trained over several epochs. Once the model is trained, predictions are made on the validation dataset. These predictions are compared to actual labels using a confusion matrix, which breaks down the number of correct and incorrect classifications.
To better interpret the results, a heatmap of the confusion matrix is generated, offering a clear visual representation of true positives, true negatives, false positives, and false negatives—key metrics for assessing model accuracy and identifying misclassification patterns.

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input

# Example data and labels (ensure these are correctly loaded and preprocessed)
X_train = np.array(["This is the first document.", "This document is the second document.", "And this is the third one."])
y_train = np.array([0, 1, 0])
X_valid = X_train  # Example validation data
y_valid = y_train  # Example validation labels

# Prepare vectorizer without limiting features
vectorizer = TfidfVectorizer(max_features=None)
vectorizer.fit(X_train)
X_train_clean = vectorizer.transform(X_train).toarray()
X_valid_clean = vectorizer.transform(X_valid).toarray()

# Model definition and training
model = Sequential([
    Input(shape=(X_train_clean.shape[1],)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train_clean, y_train, epochs=10, batch_size=1)

# Preparing y_valid_clean and verifying its length
y_valid_clean = y_valid.copy()  # This should be prepared the same way y_train was
print("Number of labels in y_valid_clean:", len(y_valid_clean))

# Predictions and confusion matrix
test_predictions = model.predict(X_valid_clean)
test_predictions_binary = (test_predictions > 0.5).astype(int).flatten()
cm = confusion_matrix(y_valid_clean, test_predictions_binary)

# Visualization
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
Negative
Positive
True Labels
2
0
0
1
Negative
Positive
Predicted Labels
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

Conclusion:

This project focused on performing sentiment analysis on Twitter data using a combination of natural language processing and machine learning techniques. We began by cleaning and transforming the raw tweet text into structured, numerical features and encoding sentiment labels for effective model training. A neural network model was developed and trained using TensorFlow and Keras, achieving a validation accuracy of 85%.

To better understand the dataset and model behavior, we utilized visual tools to examine sentiment distributions and identify frequently used terms across sentiment categories. A confusion matrix was also employed to evaluate the model's prediction accuracy and pinpoint areas of misclassification.

In the final stage, a sentiment prediction function was implemented, enabling real-time sentiment classification of new tweets. Altogether, this workflow presents a solid foundation for analyzing sentiment on social media, providing valuable insights into public opinion and user expression.

References:

https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis