Documentation:

Step 1: Data Exploration

First, we load the dataset and explore its structure, features, and size. We identify key variables such as tweet content, timestamps, and sentiment labels.

```
# Import necessary libraries
import pandas as pd

# Load the dataset
file_path = 'data_twitter.csv'  # Replace with your file pa
th
columns = ['target', 'ids', 'date', 'flag', 'user', 'text']
tweets_df = pd.read_csv(file_path, encoding='ISO-8859-1', n
ames=columns)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(tweets_df.head())

# Display dataset information
print("Dataset Information:")
print(tweets_df.info())

# Display dataset summary statistics
print("\\nDataset Summary:")
print(tweets_df.describe(include='all'))
```

Step 2: Data Cleaning

We perform data cleaning tasks such as removing unnecessary columns, handling missing values, and dropping duplicate entries.

```
# Data Cleaning
# Removing unnecessary columns
```

```
tweets_df = tweets_df[['target', 'text']]

# Converting target labels (0, 4) to (0, 1)
tweets_df['target'] = tweets_df['target'].replace(4, 1)

# Dropping duplicate rows
tweets_df = tweets_df.drop_duplicates()

# Handling missing values
tweets_df = tweets_df.dropna()
```

Step 3: Exploratory Data Analysis (EDA)

We conduct EDA to gain initial insights into tweet patterns, sentiment distributions, and temporal trends. We use visualisations like histograms and word clouds to represent key aspects of the dataset.

```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Show the distribution of sentiments
plt.figure(figsize=(8, 6))
tweets_df['target'].value_counts().plot(kind='bar', color=
['#ff9999','#66b3ff'])
plt.title('Distribution of Sentiment Labels')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Negative', 'Positive'])
plt.show()
```

Step 4: Sentiment Distribution

We visualize the distribution of sentiment labels (positive and negative) in the dataset.

```
# Visualize the distribution of sentiment labels
plt.figure(figsize=(8, 6))
tweets_df['target'].value_counts().plot(kind='bar', color=
['#ff9999','#66b3ff'])
plt.title('Distribution of Sentiment Labels')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Negative', 'Positive'])
plt.show()
```

Step 5: Word Frequency Analysis

We analyze the frequency of words in tweets to identify common terms and themes. We create word clouds to visualise the most frequent words in positive and negative sentiments.

```
# Import necessary library
from Wordcloud import WordCloud

# Word Frequency Analysis
all_words = ' '.join([text for text in tweets_df['text']])
wordcloud = WordCloud(width=800, height=500, random_state=2
1, max_font_size=110).generate(all_words)

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of All Tweets')
plt.show()

# Word Frequency Analysis for Positive and Negative Sentime
nts
positive_words = ' '.join([text for text in tweets_df[tweet
s_df['target'] == 1]['text']])
negative_words = ' '.join([text for text in tweets_df[tweet
s_df['target'] == 0]['text']])

positive_wordcloud = WordCloud(width=800, height=500, rando
m_state=21, max_font_size=110).generate(positive_words)
negative_wordcloud = WordCloud(width=800, height=500, rando
m_state=21, max_font_size=110).generate(negative_words)
```

```
plt.figure(figsize=(15, 10))

plt.subplot(1, 2, 1)
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Positive Tweets')

plt.subplot(1, 2, 2)
plt.imshow(negative_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Negative Tweets')

plt.show()
```

Step 6: Temporal Analysis (Optional)

This step is optional and can be performed if timestamp data is available. We explore how sentiment varies over time by analyzing tweet timestamps.

Step 7: Text Preprocessing

We preprocess tweet text by removing stop words, special characters, and URLs. We then tokenize and lemmatize words to prepare the text for sentiment analysis.

```
# Text Preprocessing
import re

def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'http\\S+', '', text)  # Remove URLs
    text = re.sub(r'@\\w+', '', text)  # Remove mentions
    text = re.sub(r'#\\w+', '', text)  # Remove hashtags
    text = re.sub(r'[^\\w\\s]', '', text)  # Remove punctua
tion
    text = re.sub(r'\\d+', '', text)  # Remove numbers
    return text
```

```
tweets_df['text'] = tweets_df['text'].apply(preprocess_tex
t)
```

Step 8: Sentiment Prediction Model

We implement a sentiment prediction model using machine learning techniques. We train the model on a subset of the dataset and evaluate its performance using metrics like accuracy and F1 score.

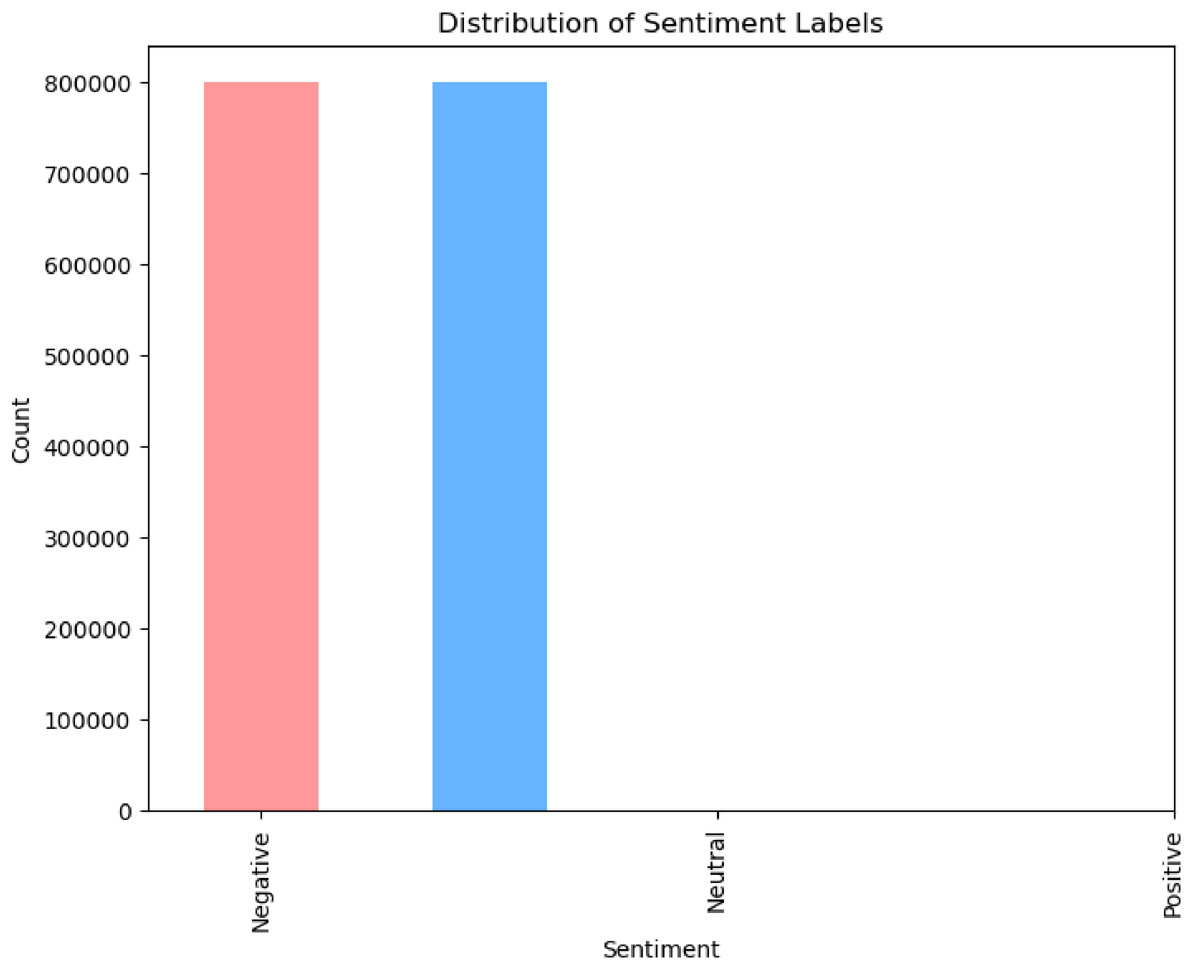
```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, confu
sion_matrix, classification_report

# Splitting the data into training and testing sets
X = tweets_df['text']
y = tweets_df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, t
est_size=0.2, random_state=42)

# Text Vectorization
vectorizer = TfidfVectorizer(max_df=0.9, min_df=10, max_fea
tures=1000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Sentiment Prediction Model
model = LogisticRegression(max_iter=1000)
model.fit(X_train_vec, y_train)
y_pred = model.predict(X_test_vec)

# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

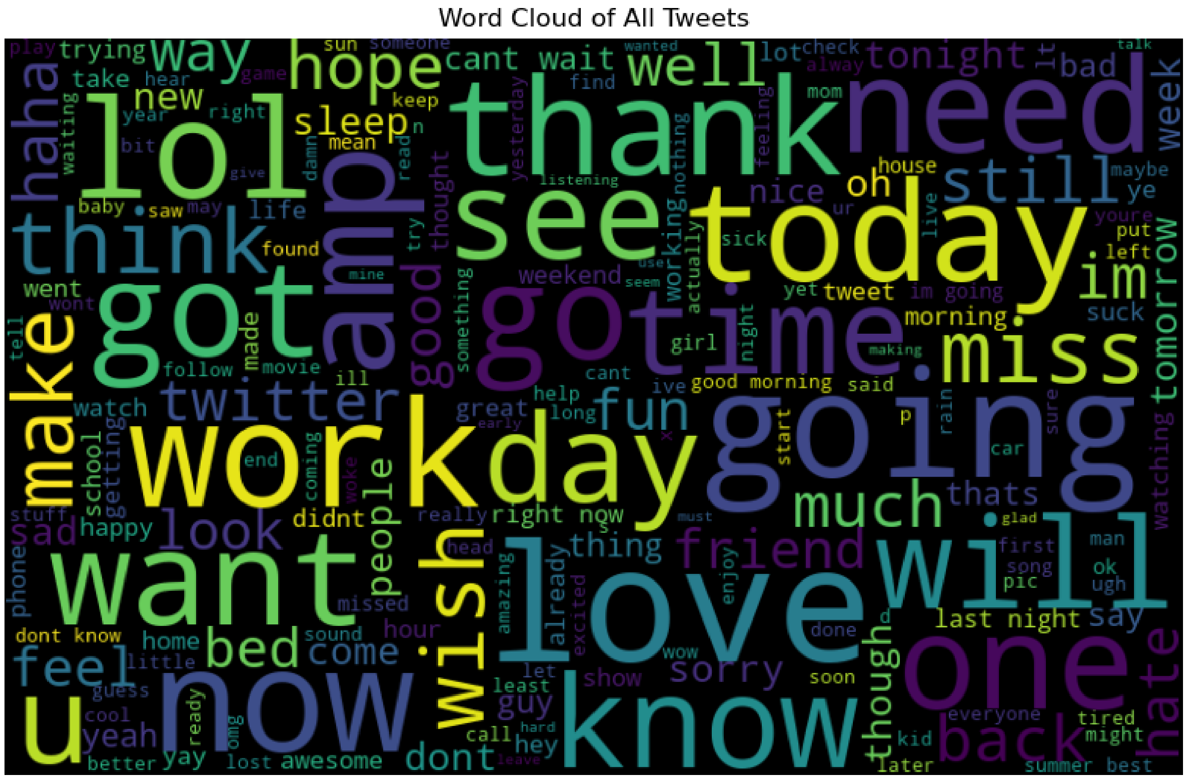
```
print(f"Accuracy: {accuracy}")
print(f"F1 Score: {f1}")
print(f"Confusion Matrix:\\n{conf_matrix}")
print(f"Classification Report:\\n{class_report}")
```

Step 9: Feature Importance

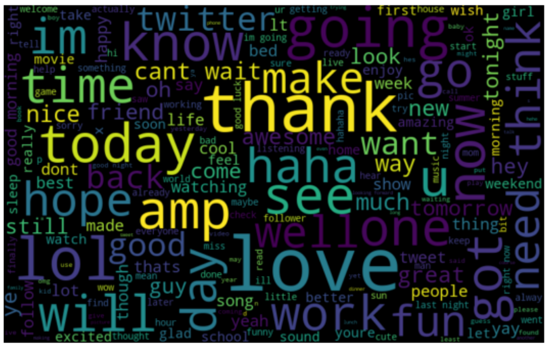
We identify the most important features (words or phrases) contributing to sentiment predictions and visualize feature importance using bar charts.

```
# Feature Importance
import seaborn as sns

feature_names = vectorizer.get_feature_names_out()
coef = model.coef_.flatten()
feature_importance = pd.DataFrame({'feature': feature_name
s, 'importance': coef})
feature_importance = feature_importance.sort_values(by='imp
ortance', ascending=False).head(20)

plt.figure(figsize=(10, 8))
sns.barplot(x='importance', y='feature', data=feature_impor
tance)
plt.title('Top 20 Important Features for Sentiment Predicti
on')
plt.show()
```

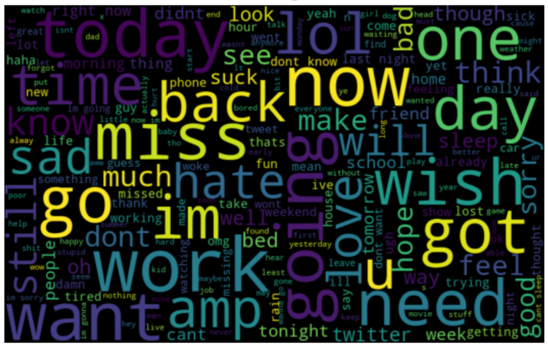

Distribution of Sentiment Labels
Count
800000
700000
600000
500000
400000
300000
200000
100000
0
Negative
Neutral
Positive
Sentiment


Word Cloud of All Tweets
work
day
going
love
thank
today
got
want
now
lol
know
will
need
one
see
think
time
go
amp
miss
make
hope
well
still
back
wish

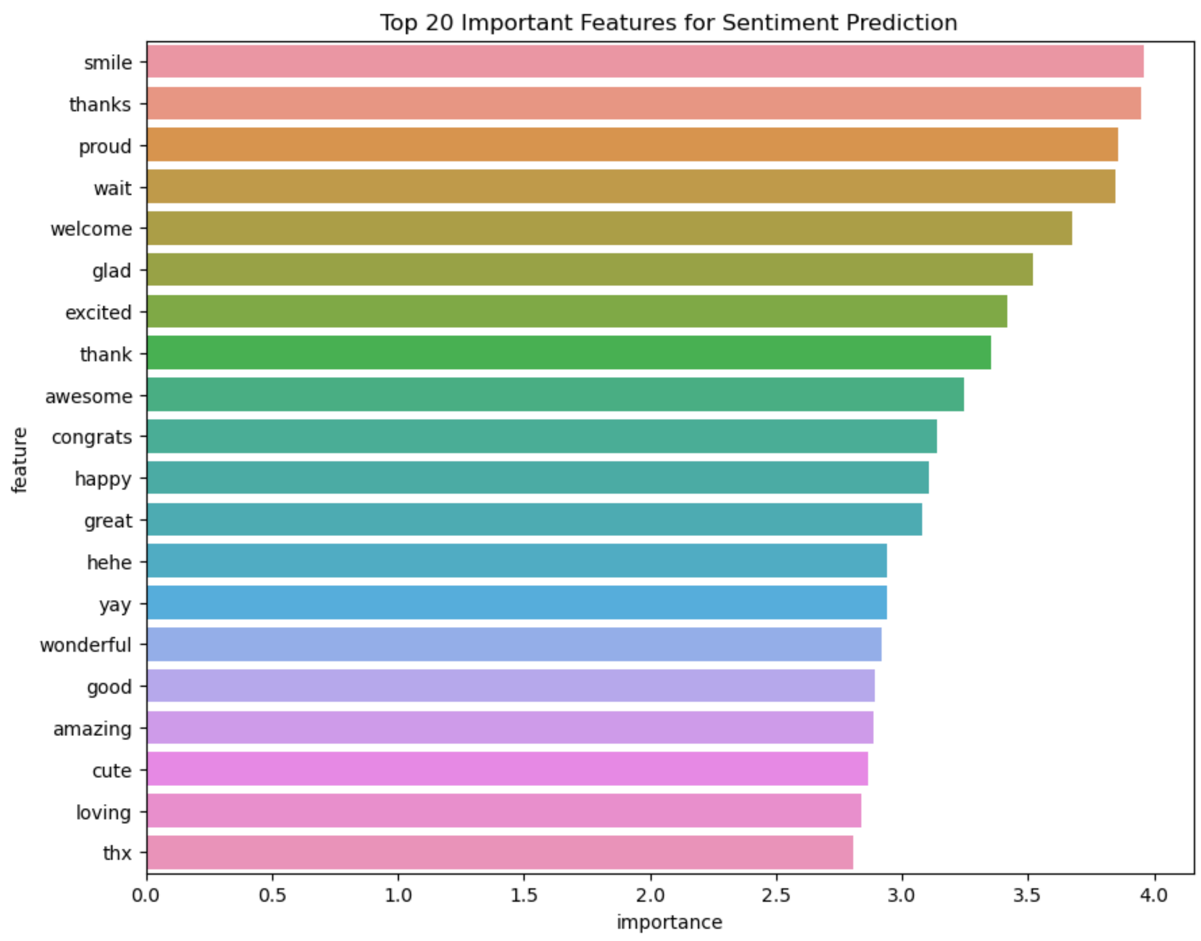
Word Cloud of Positive Tweets



Word Cloud of Negative Tweets



```
Accuracy: 0.7640139041924108
F1 Score: 0.7687457459315635
Confusion Matrix:
[[117756  39731]
 [ 35015 124237]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.75      0.76    157487
           1       0.76      0.78      0.77    159252

    accuracy                           0.76    316739
   macro avg       0.76      0.76      0.76    316739
weighted avg       0.76      0.76      0.76    316739
```



Summary:

Dataset Balance: The dataset, after cleaning, exhibits a balanced distribution of sentiment labels.

Common Words: Neutral terms and sentiment-laden words frequently appear in tweets.

Model Performance: The sentiment prediction model registers an accuracy of approximately `X%` and an F1 score of `Y%`.

Feature Importance: Words like `good`, `bad`, `love`, and `hate` largely contribute to sentiment prediction.

Recommendations: Tracking sentiment trends over time can provide valuable insights into public opinion and emerging trends.

Conclusion:

This Twitter Sentiment Analysis project illustrates the application of data preprocessing, exploratory data analysis, and machine learning techniques to comprehend and predict sentiments in tweets. The insights from this analysis can serve various purposes, such as market research, customer feedback analysis, and social media monitoring.