11. Documentation:

Data Exploration:

The initial step involves loading the dataset and inspecting its structure. The code below demonstrates how the data is loaded into a pandas DataFrame.

```
# Data Exploration
import pandas as pd
column_names = ['target','ids','date','flag','user','text']
df = pd.read_csv("training.1600000.processed.noemoticon.csv", names=column_names,
encoding='ISO-8859-1')
```

Data Cleaning:

Data cleaning is crucial to ensure the dataset's quality. In this project, irrelevant columns are dropped, missing values are handled, and duplicate entries are removed.

```
# Data Cleaning
# (i) Dropping irrelevant columns
# df = df[['target', 'text']]

# (i) Handling missing values
df.dropna(inplace=True)

# (ii) Dropping duplicate entries
df.drop_duplicates(inplace=True)
```

Exploratory Data Analysis (EDA):

EDA provides preliminary insights into the dataset. Visualization of the sentiment distribution is showcased using seaborn and matplotlib.

```
# Exploratory Data Analysis (EDA)
import matplotlib.pyplot as plt
import seaborn as sns

# Visualizing sentiment distribution
sns.countplot(x='target', data=df)
plt.title('Sentiment Distribution')
plt.show()
```

Sentiment Distribution:

Further analysis includes visualizing the distribution of sentiment labels and exploring the balance of sentiment classes.

```
# Sentiment Distribution
# Visualizing the distribution of sentiment labels
sns.countplot(x='target', data=df)
plt.title('Sentiment Distribution')
plt.show()

# Analyzing the balance of sentiment classes
sentiment_counts = df['target'].value_counts()
print(sentiment_counts)
```

Word Frequency Analysis:

Word frequency analysis involves creating word clouds for both positive and negative sentiments.

```
# Word Frequency Analysis
from wordcloud import WordCloud

# Analyzing word frequency in tweets
positive_tweets = df[df['target'] == 4]['text']
negative_tweets = df[df['target'] == 0]['text']

# Creating word clouds for positive and negative sentiments
positive_wordcloud = WordCloud().generate(' '.join(positive_tweets))
negative_wordcloud = WordCloud().generate(' '.join(negative_tweets))

# Displaying the word clouds
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.title('Positive Word Cloud')
plt.axis('off')
plt.show()

plt.imshow(negative_wordcloud, interpolation='bilinear')
plt.title('Negative Word Cloud')
plt.axis('off')
plt.show()
```

Temporal Analysis:

Temporal analysis explores how sentiment varies over the pseudo-time index, assuming the index represents the order of tweets.

```
# Temporal Analysis
# Converting the index to datetime
df.index = pd.to_datetime(df.index, unit='s')

# Exploring how sentiment varies over the "pseudo-time" index
plt.figure(figsize=(12, 6))
sns.lineplot(x=df.index, y='target', data=df)
plt.title('Temporal Analysis of Sentiment (Based on Tweet Order)')
plt.show()
```

Text Preprocessing:

Text preprocessing involves cleaning and preparing the tweet text for analysis, including tasks such as removing URLs, special characters, and stopwords.

```
# Text Preprocessing
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    # Text preprocessing steps
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters and numbers
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens if token.lower() not in
stop_words]
    return ' '.join(tokens)

df2 = df.head(15000)
df2['processed_text'] = df2['text'].apply(preprocess_text)
```

Sentiment Prediction Model:

Building a sentiment prediction model involves splitting the data into training and testing sets, vectorizing the text, and training a RandomForestClassifier.

```
# Sentiment Prediction Model
X_train, X_test, y_train, y_test = train_test_split(df2['processed_text'], df2['target'], test_size=0.2,
random_state=42)
```

```
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_tfidf, y_train)

# Evaluate the model
y_pred = model.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
print(f'Accuracy: {accuracy:.2f}')
print(f'F1 Score: {f1:.2f}')
```

Feature Importance:

Analyzing feature importance helps identify the most influential words in sentiment prediction.

```
# Feature Importance
feature_names = vectorizer.get_feature_names_out()
feature_importance = model.feature_importances_

# Creating a DataFrame to store feature names and their importance scores
feature_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importance})
top_features = feature_df.nlargest(20, 'Importance')

# Visualizing feature importance
plt.figure(figsize=(10, 6))
plt.bar(top_features['Feature'], top_features['Importance'])
plt.xticks(rotation=45, ha='right')
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Top 20 Features Importance')
plt.show()
```

12. Insights and Recommendations:

Key Insights:

Sentiment Distribution:

- The sentiment distribution indicates that there is a balance between positive and negative sentiments in the dataset.

Word Frequency Analysis:

- Word clouds provide a visual representation of frequently occurring words in positive and negative tweets.

Temporal Analysis:

- Temporal analysis shows the trend of sentiment over the dataset's pseudo-time index.

Feature Importance:

- The top features indicate the most influential words in sentiment prediction.

Recommendations:

Engagement Strategies:

- Engage with users expressing positive sentiments to enhance brand loyalty.
- Address concerns raised in negative sentiments to improve overall sentiment.

Content Optimization:

- Optimize content creation based on frequently occurring words in positive sentiments.
- Consider adjusting strategies for words influencing negative sentiments.

Temporal Insights:

- Explore the temporal trend of sentiments to align marketing campaigns with periods of positive sentiment.

Feature Importance:

- Consider the top features in content creation to enhance the effectiveness of sentiment prediction.