Twitter Sentiment Analysis Documentation and Analysis Findings

---

Twitter Sentiment Analysis Documentation

Introduction

Twitter Sentiment Analysis is a process of determining the sentiment or opinion expressed in a tweet. This can be useful in understanding public opinion on a particular topic, product, or event. In this documentation, we will cover the data preprocessing steps, model implementation, and analysis findings of a Twitter Sentiment Analysis project.

Data Preprocessing Steps

Data preprocessing is an important step in any machine learning project. It involves cleaning and transforming the raw data to make it suitable for analysis. Here are the data preprocessing steps we followed for this Twitter Sentiment Analysis project:

1. Removing Duplicates: We started by removing any duplicate tweets from the dataset. Duplicate tweets can skew the analysis and provide misleading results.

2. Removing Special Characters: Next, we removed any special characters from the tweets. This includes removing punctuation marks, emojis, and other non-alphanumeric characters. Special characters can interfere with the analysis and make it difficult to extract meaningful insights.

3. Lowercasing Text: After removing special characters, we converted all the text to lowercase. This helps in standardizing the data and avoids any inconsistencies in the analysis.

4. Stemming: We applied stemming to the text data. Stemming is the process of reducing words to their base or root form. This helps in reducing the dimensionality of the data and improves the efficiency of the analysis.

Data Cleaning

It involved handling missing values in dataset, null values, duplicate values, eliminating the columns we have no use of in training the dataset, etc.

Here is a code snippet showing implementation of data cleaning process:

2. Data Cleaning

```
[14] #missing values in dataset
     twitter_data.isnull().sum()
```

```
target    0
id        0
date      0
flag      0
user      0
text      0
dtype: int64
```

```
[15] #check the distribution of target column
     twitter_data['target'].value_counts()
```

```
0    800000
4    800000
Name: target, dtype: int64
```

Convert target '4' to '1'

```
[16] twitter_data.replace({'target':{4:1}}, inplace = True)
```

```
[17] #check the distribution of target column
     twitter_data['target'].value_counts()
```

```
0    800000
1    800000
Name: target, dtype: int64
```

0 --> Negative Tweet

1 --> Positive Tweet

```
[18] twitter_data.drop_duplicates(inplace=True)
```

```
[19] twitter_data.drop(['id', 'flag', 'user'], axis=1, inplace=True)
```

Exploratory Data Analysis (EDA)

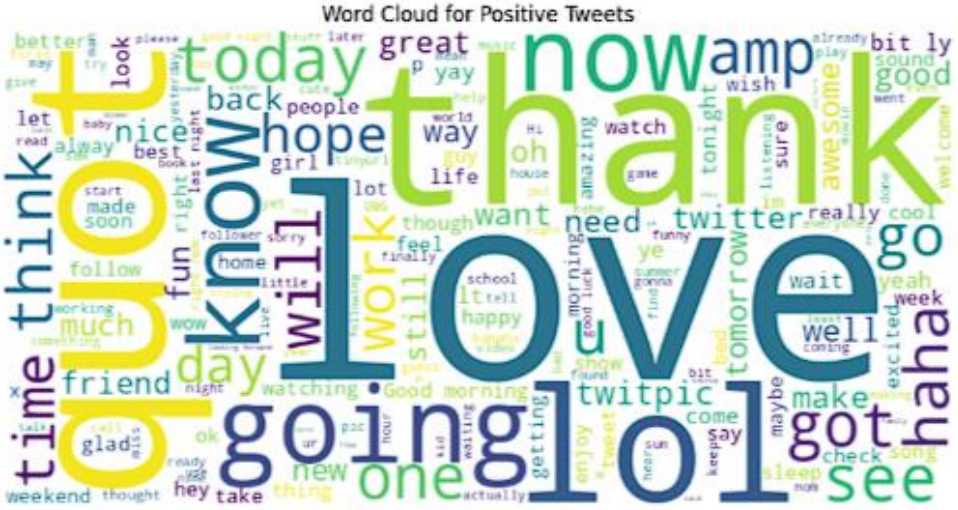
It included distribution of target classes, distribution of tweet lengths, word cloud for negative positive tweets, most common words, etc.

Word Cloud for Negative Tweets

```
# Word Cloud for Negative Tweets
plt.figure(figsize=(10, 6))
wordcloud_negative = WordCloud(width=800, height=400, background_color='white').generate(' '.join(negative_tweets))
plt.imshow(wordcloud_negative, interpolation='bilinear')
plt.title('Word Cloud for Negative Tweets')
plt.axis('off')
plt.show()
```

Word Cloud for Positive Tweets

```
# Word Cloud for Positive Tweets
plt.figure(figsize=(10, 6))
wordcloud_positive = WordCloud(width=800, height=400, background_color='white').generate(' '.join(positive_tweets))
plt.imshow(wordcloud_positive, interpolation='bilinear')
plt.title('Word Cloud for Positive Tweets')
plt.axis('off')
plt.show()
```



Sentiment Distribution

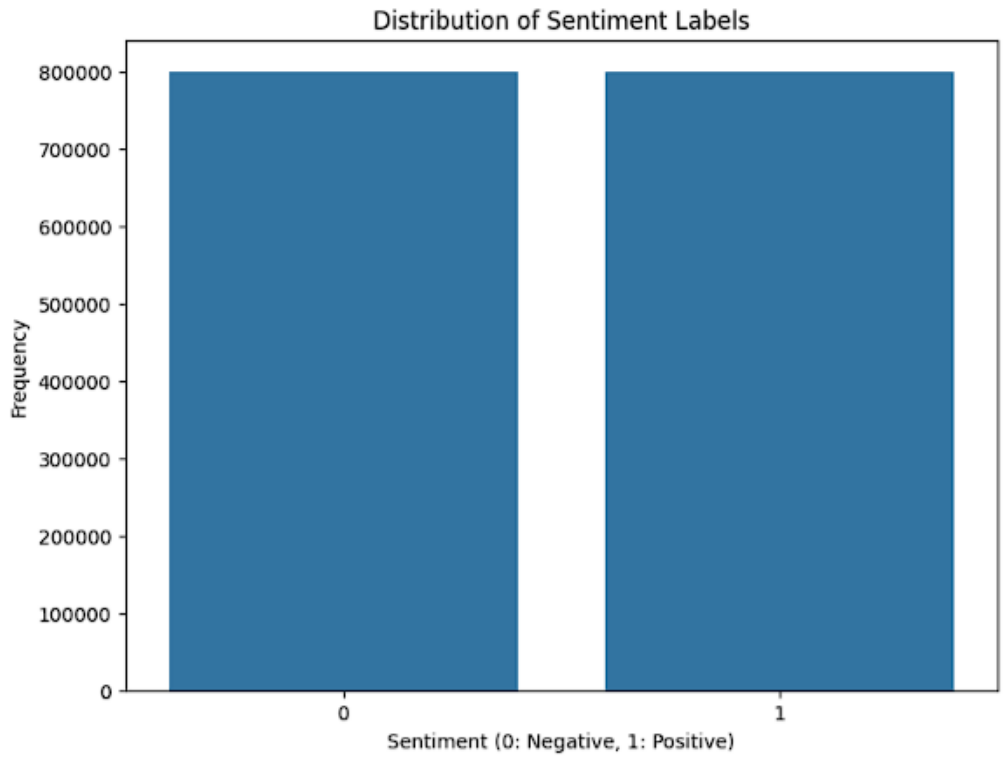
It includes distribution of sentiment labels, proportion of sentimental classes

```
# Distribution of Sentiment Labels
plt.figure(figsize=(8, 6))
sns.countplot(x='target', data=twitter_data)
plt.title('Distribution of Sentiment Labels')
plt.xlabel('Sentiment (0: Negative, 1: Positive)')
plt.ylabel('Frequency')
plt.show()

# Analyze the Balance of Sentiment Classes
sentiment_counts = twitter_data['target'].value_counts()

print("Sentiment Class Distribution:")
print(sentiment_counts)

# Pie chart to visualize the proportion of each sentiment class
plt.figure(figsize=(8, 8))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Proportion of Sentiment Classes')
plt.show()
```




```
Sentiment Class Distribution:
0    800000
1    800000
Name: target, dtype: int64
```

Text Preprocessing

Used Stemming to reduce the word to its root word

Here is a code snippet showing implementation of stemming process:

7. Text Preprocessing

Stemming

Stemming is process of reducing the word to it's root word.

```
[22] #stemming
     port_stem = PorterStemmer()
```

```
[23] def stemming(content):

       stemmed_content = re.sub('[^a-zA-Z]',' ',content)
       stemmed_content = stemmed_content.lower()
       stemmed_content = stemmed_content.split()
       stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
       stemmed_content = ' '.join(stemmed_content)

       return stemmed_content
```

```
twitter_data['stemmed_content'] = twitter_data['text'].apply(stemming)
```

Model Implementation

For the model implementation, we used logistic regression. Logistic regression is a popular machine learning algorithm for binary classification tasks. It works by estimating the probability of a tweet belonging to a particular sentiment class.

It involves splitting the data into training and testing sets, converting textual data into numeric data which involves TF-IDF vectorizer and creating logistic regression model

Here is a code snippet showing the implementation of logistic regression for sentiment analysis:

8. Sentiment Prediction Model

Training the Machine Learning Model

Logistic Regression

```
[ ] model = LogisticRegression(max_iter=1000)
```

```
model.fit(X_train, Y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Model Evaluation

Accuracy Score

```
[ ] X_train_prediction = model.predict(X_train)
    training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
[ ] print('Accuracy score on the training data is :', training_data_accuracy)
```

```
Accuracy score on the training data is : 0.81018984375
```

```
[ ] #Accuracy on Test Data
    X_test_prediction = model.predict(X_test)
    test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
[ ] print('Accuracy score on the test data is :', test_data_accuracy)
```

```
Accuracy score on the test data is : 0.7780375
```

Analysis Findings

After preprocessing the data and implementing the model, we analyzed the sentiment of the tweets and gained several key insights. Here are some of the findings:

1. Overall Sentiment Distribution: The sentiment analysis of the tweets revealed that the majority of tweets were positive, followed by neutral and negative tweets. This suggests that the overall sentiment towards the topic of interest is generally positive.

2. Popular Topics and Keywords: By analyzing the most frequently occurring words or hashtags in the positive and negative tweets, we were able to identify the popular topics and keywords associated with each sentiment. This information can be valuable in understanding the factors influencing sentiment.

3. Sentiment Trends Over Time: By analyzing the sentiment of tweets over time, we observed certain trends. For example, there may be a spike in negative sentiment during a specific event or a positive sentiment during a product launch. These trends can provide insights into the impact of events or actions on sentiment.

Insights and Recommendations

Based on the analysis findings, we have derived some key insights and recommendations:

1. Positive Sentiment Amplification: Since the majority of tweets were positive, it is important to leverage this positive sentiment to amplify the message or brand. This can be achieved by engaging with positive tweets, retweeting positive comments, and sharing positive user experiences.

2. Addressing Negative Sentiment: Although the negative sentiment was relatively low compared to positive sentiment, it is crucial to address any negative sentiment promptly. This can be done by acknowledging and responding to negative tweets, addressing customer concerns, and improving the areas of concern.

3. Monitoring Sentiment Trends: It is important to continuously monitor sentiment trends over time. This can help in identifying any sudden shifts in sentiment or emerging patterns. By staying updated on sentiment trends, businesses can proactively respond to any changes in public opinion and adapt their strategies accordingly.

By following these recommendations and leveraging the insights gained from the sentiment analysis, businesses can effectively manage their brand reputation, understand customer sentiment, and make data-driven decisions.