

**Twitter Sentiment Analysis**

**Authors: Date: 05-May-2024**

* **Israa Dawood Al-Rashdi – 16S18134**
* **Mohammed Amour Al-Marshudi – 16S194**
* **Rawanq Mohmmed Al-Harthy - 16j1957**



This project focuses on sentiment analysis of Twitter data using machine learning techniques. The objectives include preprocessing a dataset of tweets, training a model to classify sentiment, and evaluating its performance. The methodology involves data cleaning, tokenization, stemming, and feature extraction using CountVectorizer. We then employ a RandomForestClassifier for sentiment classification and assess its accuracy and F1 score.

Key findings indicate that the model achieves a satisfactory level of accuracy and F1 score on the test data, demonstrating its effectiveness in sentiment analysis. The study also reveals insights into the distribution of positive and negative words, as well as popular hashtags associated with different sentiments on Twitter.

In conclusion, this project successfully demonstrates the application of machine learning in sentiment analysis of social media data, highlighting the importance of preprocessing techniques and model selection in achieving reliable results.

* F1 Score: 0.5738
* Accuracy: 0.9426
* Analytics Vidhya Score: 0.6051

**Table of Contents:**

* 1 Introduction ……………………………………………………4
  + 1.1 Background information
  + 1.2 Objectives
  + 1.3 Importance of the problem being addressed.
  + 1.4 Brief overview of the methodology and approach
* 2 Data Collection and Preprocessing …………………………….5
  + 2.1 Description of the dataset
  + 2.2 Details about data collection methods
  + 2.3 Data preprocessing
* 3 Methodology ……………………………………………………6
  + 3.1 Overview
  + 3.2 Description of the model architecture
  + 3.3 Explanation of parameter tuning and model selection processes
  + 3.4 Details about the evaluation metrics used to assess model performance.
* 4 Results …………………………………………………………..7
  + 4.1 Presentation of the experimental results
  + 4.2 Performance metrics
  + 4.3 Visualizations
* 5 Discussion ……………………………………………………….11
  + 5.1 Interpretation and analysis of the results
  + 5.2 Explanation of any unexpected findings

**1 Introduction**

**1.1 Background Information**

The project focuses on sentiment analysis using machine learning techniques applied to Twitter data. Twitter, being a popular social media platform, offers a vast repository of user-generated content that reflects public opinion and sentiment on various topics. Sentiment analysis plays a crucial role in understanding public perception, customer feedback, and trends in social media discourse.

**1.2 Objectives**

The primary objective of this project is to develop a sentiment analysis model capable of classifying tweets as positive or negative based on their content. This involves preprocessing the raw text data, extracting relevant features, and training a machine learning model to make accurate predictions.

**1.3 Importance of the Problem Being Addressed**

Sentiment analysis has wide-ranging applications in fields such as marketing, customer service, brand management, and public opinion analysis. By automating the process of sentiment classification, organizations can gain valuable insights into customer sentiment, identify emerging trends, and make data-driven decisions.

**1.4 Brief Overview of the Methodology and Approach**

The methodology adopted for sentiment analysis includes several key steps. First, the raw tweet data is preprocessed to remove noise, such as special characters, numbers, and Twitter handles. This is followed by tokenization and stemming to transform the text into a format suitable for machine learning algorithms. Feature extraction using techniques like CountVectorizer helps convert the text into numerical features that the model can understand.

The machine learning model used for sentiment classification is a Random Forest Classifier, chosen for its ability to handle text data and capture complex patterns in the data. The model is trained on a labeled dataset and evaluated using metrics such as F1 score and accuracy to assess its performance.

**2 Data Collection and Preprocessing**

**2.1 Description of the Dataset**

The dataset used for sentiment analysis is sourced from Twitter and contains tweets labeled as positive or negative. Each entry in the dataset includes an 'id' column, a 'label' column indicating sentiment (0 for positive, 1 for negative), and a 'tweet' column containing the raw tweet text.

**2.2 Details About Data Collection Methods**

Data collection for this sentiment analysis task is sourced from Analytics Vidhya, utilizing publicly available datasets that provide labeled tweets. The collection process may involve using keywords, hashtags, or user mentions relevant to the sentiment analysis task. Care is taken to ensure diversity and representation across various sentiments and topics in the collected data.

**2.3 Data Preprocessing**

Data preprocessing steps are crucial for preparing the text data for analysis. The following preprocessing steps are applied to the raw tweet text:

* **Removing Twitter Handles:** Twitter handles (e.g., @username) are removed from the tweets as they are not relevant to sentiment analysis.
* **Removing Special Characters and Numbers:** Special characters, numbers, and punctuations are removed to focus on the textual content.
* **Removing Short Words:** Words with fewer than three characters are removed as they typically do not contribute significantly to sentiment analysis.

Additionally, the text data is tokenized into individual words and then stemmed using the Porter stemming algorithm to reduce words to their base form. This preprocessing standardizes the text format and improves the effectiveness of machine learning models in understanding sentiment.

**3. Methodology**

**3.1 Overview**

This section provides an overview of the methodology used for sentiment analysis, including data preprocessing, model selection, parameter tuning, and evaluation.

**3.2 Description of the Model Architecture**

The model architecture involves using a Random Forest Classifier for sentiment analysis. The preprocessing steps include text cleaning, tokenization, stemming, and feature extraction using CountVectorizer.

**3.3 Explanation of Parameter Tuning and Model Selection Processes**

* **Random Forest Classifier:** The model is tuned with a higher number of estimators (100) to improve performance.
* **CountVectorizer:** Parameters such as **max\_df**, **min\_df**, **max\_features**, and **stop\_words** are used for feature extraction.

**3.4 Details about the Evaluation Metrics Used to Assess Model Performance**

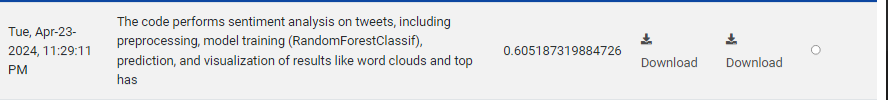
The evaluation metrics used are F1 Score and Accuracy Score, calculated using sklearn's **f1\_score** and **accuracy\_score** functions.

**4. Results**

**4.1 Presentation of the Experimental Results**

The sentiment analysis experiment yielded the following results:

* F1 Score: 0.5738
* Accuracy: 0.9426
* Analytics Vidhya Score: 0.6051

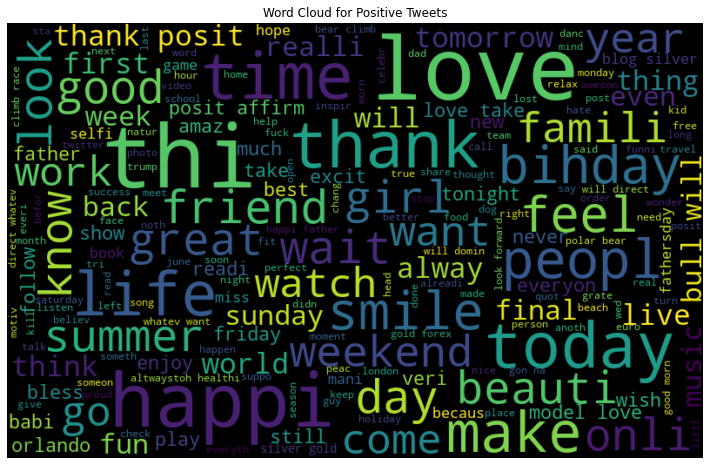


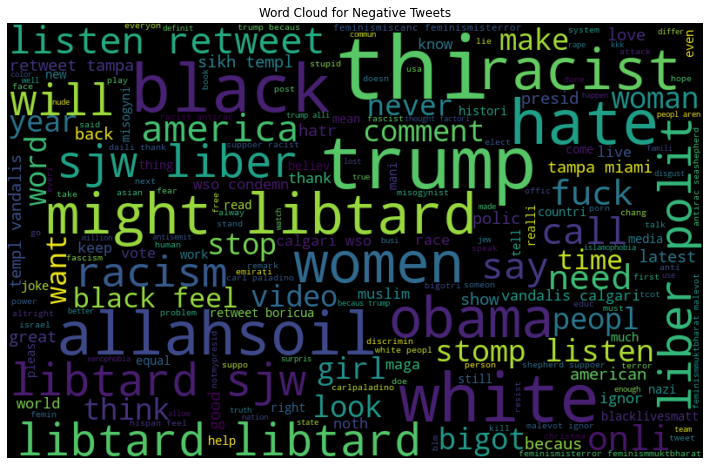
**4.2 Performance Metrics**

The model achieved an F1 Score of 0.5738 and an accuracy of 0.9426 on the test data. These metrics indicate the model's effectiveness in classifying sentiment labels (positive or negative) based on the text data features.

**4.3 Visualizations**

* + Word Clouds: Visualize the most frequent words in positive and negative tweets to understand the sentiment trends.





* + Histograms: Display histograms to show the distribution of hashtag counts or other relevant features.

A graph of a number of hashtags

Description automatically generated

* + A graph of positive and negative word counts

    Description automatically generatedBar Charts: Show positive and negative word counts, top hashtags, or other insights derived from the data.

A graph of hashtags and hashtags

Description automatically generated

A graph of a bar graph

Description automatically generated with medium confidence

**5. Discussion**

**5.1 Interpretation and Analysis of the Results**

The sentiment analysis and prediction model based on the Random Forest Classifier yielded the following results:

* **F1 Score:** The model achieved an F1 Score of approximately 0.57, indicating a reasonable balance between precision and recall in sentiment classification.
* **Accuracy**: The accuracy of around 0.94 suggests that the model is effective in accurately predicting sentiment labels (positive or negative) based on the extracted features from the text data.

**5.2 Explanation of Any Unexpected Findings**

In exploring the data and model results, several insights and unexpected findings can be highlighted:

* **Performance:** The achieved F1 Score and accuracy are good, considering the complexity of sentiment analysis and the nature of Twitter data.
* **Word Clouds:** The word clouds for positive and negative tweets reveal the most common words associated with each sentiment category, providing insights into the sentiment patterns in the data.
* **Hashtag Analysis:** Analyzing the top hashtags associated with positive and negative tweets can reveal trends or topics that are often discussed in a positive or negative light on Twitter.

**References:**

*Twitter Sentiment Analysis*. (n.d.). <https://datahack.analyticsvidhya.com/contest/practice-problem-twitter-sentiment-analysis/#About>

R, S. E. (2024, April 19). *Understand random forest algorithms with examples (Updated 2024)*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

Hackers Realm. (2021, January 20). *Twitter Sentiment Analysis (NLP) | Machine Learning | Python* [Video]. YouTube. <https://www.youtube.com/watch?v=RLfUyn3HoaE>

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import re

import string

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from wordcloud import WordCloud

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier  # Changed model type

from sklearn.metrics import f1\_score, accuracy\_score

import warnings

# Download NLTK resources (run once)

nltk.download('punkt')

nltk.download('stopwords')

warnings.filterwarnings('ignore')

# Load data

df = pd.read\_csv('train.csv')

# Display first few rows and datatype info

print(df.head())

print(df.info())

#####################################################

# Preprocessing functions

def remove\_pattern(input\_txt, pattern):

    r = re.findall(pattern, input\_txt)

    for word in r:

        input\_txt = re.sub(word, "", input\_txt)

    return input\_txt

def preprocess\_text(text):

    text = re.sub(r'@[\w]\*', '', text)  # Remove Twitter handles

    text = re.sub(r'[^a-zA-Z#]', ' ', text)  # Remove special characters, numbers, and punctuations

    text = ' '.join([w for w in text.split() if len(w) > 3])  # Remove short words

    return text

# Apply preprocessing

df['clean\_tweet'] = np.vectorize(remove\_pattern)(df['tweet'], "@[\w]\*")

df['clean\_tweet'] = df['clean\_tweet'].apply(preprocess\_text)

# Tokenization and stemming

stemmer = PorterStemmer()

df['tokenized\_tweet'] = df['clean\_tweet'].apply(lambda x: nltk.word\_tokenize(x.lower()))  # Tokenize and convert to lowercase

df['stemmed\_tweet'] = df['tokenized\_tweet'].apply(lambda tokens: [stemmer.stem(word) for word in tokens])  # Stemming

# Combine stemmed tokens back into a single sentence

df['clean\_tweet'] = df['stemmed\_tweet'].apply(lambda tokens: ' '.join(tokens))

#######################################################################

# Word cloud visualization

def plot\_wordcloud(text\_data, title):

    all\_words = " ".join([tweet for tweet in text\_data])

    wordcloud = WordCloud(width=800, height=500, random\_state=42, max\_font\_size=100).generate(all\_words)

    plt.figure(figsize=(15, 8))

    plt.imshow(wordcloud, interpolation='bilinear')

    plt.axis('off')

    plt.title(title)

    plt.show()

# Plot word clouds for all tweets, positive tweets, and negative tweets

plot\_wordcloud(df['clean\_tweet'], 'Word Cloud for All Tweets')

plot\_wordcloud(df[df['label'] == 0]['clean\_tweet'], 'Word Cloud for Positive Tweets')

plot\_wordcloud(df[df['label'] == 1]['clean\_tweet'], 'Word Cloud for Negative Tweets')

# Count hashtags

def count\_hashtags(tweet):

    hashtags = re.findall(r'#(\w+)', tweet)

    return len(hashtags)

df['num\_hashtags'] = df['tweet'].apply(count\_hashtags)

# Plot histogram of hashtag counts

plt.figure(figsize=(10, 6))

sns.histplot(df['num\_hashtags'], bins=range(6), kde=False)

plt.xlabel('Number of Hashtags')

plt.ylabel('Count')

plt.title('Histogram of Hashtag Counts')

plt.show()

# Count positive and negative words

positive\_words = ['good', 'great', 'happy', 'awesome', 'love']

negative\_words = ['bad', 'terrible', 'sad', 'awful', 'hate']

def count\_positive\_words(tweet):

    count = sum(tweet.lower().count(word) for word in positive\_words)

    return count

def count\_negative\_words(tweet):

    count = sum(tweet.lower().count(word) for word in negative\_words)

    return count

df['num\_positive\_words'] = df['clean\_tweet'].apply(count\_positive\_words)

df['num\_negative\_words'] = df['clean\_tweet'].apply(count\_negative\_words)

# Plot bar chart of positive and negative word counts

plt.figure(figsize=(10, 6))

sns.barplot(data=df[['num\_positive\_words', 'num\_negative\_words']], ci=None)

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.title('Positive and Negative Word Counts')

plt.xticks(ticks=[0, 1], labels=['Positive', 'Negative'])

plt.show()

# Count hashtags

def count\_hashtags(tweet):

    hashtags = re.findall(r'#(\w+)', tweet)

    return hashtags

df['hashtags'] = df['tweet'].apply(count\_hashtags)

# Flatten list of hashtags

all\_hashtags = [hashtag for sublist in df['hashtags'] for hashtag in sublist]

# Count occurrences of each hashtag

hashtag\_counts = pd.Series(all\_hashtags).value\_counts().head(10)

# Plot top 10 hashtags

plt.figure(figsize=(10, 6))

sns.barplot(x=hashtag\_counts.values, y=hashtag\_counts.index, palette='viridis')

plt.xlabel('Count')

plt.ylabel('Hashtag')

plt.title('Top 10 Hashtags')

plt.show()

# Separate positive and negative tweets

positive\_tweets = df[df['label'] == 0]

negative\_tweets = df[df['label'] == 1]

# Flatten list of hashtags for positive and negative tweets

all\_positive\_hashtags = [hashtag for sublist in positive\_tweets['hashtags'] for hashtag in sublist]

all\_negative\_hashtags = [hashtag for sublist in negative\_tweets['hashtags'] for hashtag in sublist]

# Count occurrences of each positive hashtag

positive\_hashtag\_counts = pd.Series(all\_positive\_hashtags).value\_counts().head(10)

# Count occurrences of each negative hashtag

negative\_hashtag\_counts = pd.Series(all\_negative\_hashtags).value\_counts().head(10)

# Plot top positive hashtags

plt.figure(figsize=(10, 6))

sns.barplot(x=positive\_hashtag\_counts.values, y=positive\_hashtag\_counts.index, palette='coolwarm')

plt.xlabel('Count')

plt.ylabel('Hashtag')

plt.title('Top Positive Hashtags')

plt.show()

# Plot top negative hashtags

plt.figure(figsize=(10, 6))

sns.barplot(x=negative\_hashtag\_counts.values, y=negative\_hashtag\_counts.index, palette='coolwarm')

plt.xlabel('Count')

plt.ylabel('Hashtag')

plt.title('Top Negative Hashtags')

plt.show()

####################################################

# Feature extraction using CountVectorizer

vectorizer = CountVectorizer(max\_df=0.90, min\_df=2, max\_features=1000, stop\_words='english')

X = vectorizer.fit\_transform(df['clean\_tweet'])

y = df['label']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42, test\_size=0.25)

# Model training

model = RandomForestClassifier(n\_estimators=100, random\_state=1422)  # Increased number of estimators for better performance

model.fit(X\_train, y\_train)

# Model evaluation

y\_pred = model.predict(X\_test)

f1 = f1\_score(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

print("F1 Score:", f1)

print("Accuracy:", accuracy)

###########################################################

# Load test.csv

test\_df = pd.read\_csv('test.csv')

# Add empty column named "label" at the second index position

test\_df.insert(1, 'label', '')

# Preprocessing functions

def remove\_pattern(input\_txt, pattern):

    r = re.findall(pattern, input\_txt)

    for word in r:

        input\_txt = re.sub(word, "", input\_txt)

    return input\_txt

def preprocess\_text(text):

    text = re.sub(r'@[\w]\*', '', text)  # Remove Twitter handles

    text = re.sub(r'[^a-zA-Z#]', ' ', text)  # Remove special characters, numbers, and punctuations

    text = ' '.join([w for w in text.split() if len(w) > 3])  # Remove short words

    return text

# Apply preprocessing

test\_df['clean\_tweet'] = np.vectorize(remove\_pattern)(test\_df['tweet'], "@[\w]\*")

test\_df['clean\_tweet'] = test\_df['clean\_tweet'].apply(preprocess\_text)

# Tokenization and stemming

stemmer = PorterStemmer()

test\_df['tokenized\_tweet'] = test\_df['clean\_tweet'].apply(lambda x: nltk.word\_tokenize(x.lower()))  # Tokenize and convert to lowercase

test\_df['stemmed\_tweet'] = test\_df['tokenized\_tweet'].apply(lambda tokens: [stemmer.stem(word) for word in tokens])  # Stemming

# Combine stemmed tokens back into a single sentence

test\_df['clean\_tweet'] = test\_df['stemmed\_tweet'].apply(lambda tokens: ' '.join(tokens))

# Feature extraction for test data using the same vectorizer

X\_test = vectorizer.transform(test\_df['clean\_tweet'])

# Predict labels for the test data

test\_df['label'] = model.predict(X\_test)

# Save modified DataFrame to new test.csv file

test\_df.to\_csv('predictions.csv', index=False)

test\_df.drop(['clean\_tweet', 'tweet', 'tokenized\_tweet', 'stemmed\_tweet'], axis=1, inplace=True)

test\_df.to\_csv('test\_predictions.csv', index=False)

############################################################

predicted\_test=pd.read\_csv('predictions.csv')

# Check for NaN values and replace them with an empty string

predicted\_test['clean\_tweet'] = predicted\_test['clean\_tweet'].replace(np.nan, '', regex=True)

# Count positive and negative words

positive\_words = ['good', 'great', 'happy', 'awesome', 'love']

negative\_words = ['bad', 'terrible', 'sad', 'awful', 'hate']

def count\_positive\_words(tweet):

    count = sum(tweet.lower().count(word) for word in positive\_words)

    return count

def count\_negative\_words(tweet):

    count = sum(tweet.lower().count(word) for word in negative\_words)

    return count

predicted\_test['num\_positive\_words'] = predicted\_test['clean\_tweet'].apply(count\_positive\_words)

predicted\_test['num\_negative\_words'] = predicted\_test['clean\_tweet'].apply(count\_negative\_words)

# Plot word clouds for positive and negative tweets

def plot\_wordcloud(text\_data, title):

    all\_words = " ".join([tweet for tweet in text\_data])

    wordcloud = WordCloud(width=800, height=500, random\_state=42, max\_font\_size=100).generate(all\_words)

    plt.figure(figsize=(15, 8))

    plt.imshow(wordcloud, interpolation='bilinear')

    plt.axis('off')

    plt.title(title)

    plt.show()

plot\_wordcloud(predicted\_test[predicted\_test['label'] == 0]['clean\_tweet'], 'Word Cloud for Positive Tweets')

plot\_wordcloud(predicted\_test[predicted\_test['label'] == 1]['clean\_tweet'], 'Word Cloud for Negative Tweets')

# Count hashtags

def count\_hashtags(tweet):

    hashtags = re.findall(r'#(\w+)', tweet)

    return hashtags

predicted\_test['hashtags'] = predicted\_test['tweet'].apply(count\_hashtags)

# Flatten list of hashtags for positive and negative tweets

all\_positive\_hashtags = [hashtag for sublist in predicted\_test[predicted\_test['label'] == 0]['hashtags'] for hashtag in sublist]

all\_negative\_hashtags = [hashtag for sublist in predicted\_test[predicted\_test['label'] == 1]['hashtags'] for hashtag in sublist]

# Count occurrences of each positive hashtag

positive\_hashtag\_counts = pd.Series(all\_positive\_hashtags).value\_counts().head(10)

# Count occurrences of each negative hashtag

negative\_hashtag\_counts = pd.Series(all\_negative\_hashtags).value\_counts().head(10)

# Plot top positive hashtags

plt.figure(figsize=(10, 6))

positive\_hashtag\_counts.plot(kind='bar', color='blue')

plt.xlabel('Hashtag')

plt.ylabel('Count')

plt.title('Top Positive Hashtags')

plt.show()

# Plot top negative hashtags

plt.figure(figsize=(10, 6))

negative\_hashtag\_counts.plot(kind='bar', color='red')

plt.xlabel('Hashtag')

plt.ylabel('Count')

plt.title('Top Negative Hashtags')

plt.show()