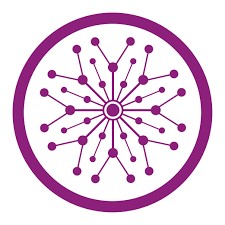
**The Superior University**



**Project Title : Twitter Sentiment Analysis**

**Course: AI Lab**

**Instructor:**  **Prof. Rasikh Ali**

**Semester: 3rd**

**Section : 3 – C**

**Submission Date: 10/12/2024**

**Twitter Sentiment Analysis Project**

**1. Introduction**

The objective of this project is to perform sentiment analysis on Twitter data. Sentiment analysis is a natural language processing (NLP) technique used to determine whether data is positive, negative, or neutral. In this project, we classify tweets as either positive or negative based on their textual content. The process involves data preprocessing, model training, and evaluation of the model’s performance.

**2. Dataset**

Two datasets were used in this project:

* **Training Data:** train\_tweets.csv
* **Testing Data:** test\_tweets.csv

**Data Fields:**

* label: Sentiment of the tweet (0 for negative, 1 for positive)
* tweet: The text content of the tweet

**3. Libraries and Tools**

The following Python libraries were utilized to conduct the analysis:

python

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from textblob import TextBlob

import re

from nltk.corpus import stopwords

from nltk.stem.wordnet import WordNetLemmatizer

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.naive\_bayes import MultinomialNB

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

from sklearn.pipeline import Pipeline

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

import nltk

**4. Data Exploration**

Initial exploration of the datasets was conducted to understand their structure and content:

python

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

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

train\_tweets.head()

train\_tweets.tail()

train\_tweets.info()

train\_tweets.isnull().sum()

train\_tweets.columns

**Key Findings:**

* The dataset contains tweets with corresponding sentiment labels.
* There are no missing values in the datasets.
* The columns include 'label' and 'tweet'.

**5. Data Visualization**

**Average Word Length vs Label**

A bar plot was created to show the average length of tweets for each sentiment label.

python

train\_tweets['length'] = train\_tweets['tweet'].apply(len)

fig1 = sns.barplot(x='label', y='length', data=train\_tweets, palette='PRGn')

plt.title('Average Word Length vs label')

plot = fig1.get\_figure()

plot.savefig('Barplot.png')

plt.show()

**Label Counts**

A count plot was generated to visualize the distribution of sentiment labels in the dataset.

python

fig2 = sns.countplot(x='label', data=train\_tweets)

plt.title('Label Counts')

plot = fig2.get\_figure()

plot.savefig('Count Plot.png')

plt.show()

**6. Text Processing**

Text processing involves cleaning and normalizing the tweet text. The steps included:

* Removing special characters
* Removing stopwords
* Lemmatizing the words

**Text Processing Function:**

python

def text\_proc(tweet):

def form\_sent(tweet):

blob = TextBlob(tweet)

return ' '.join(blob.words)

new\_tweet = form\_sent(tweet)

def no\_user\_sym(tweet):

words = [w for w in tweet.split() if w != 'user']

clean\_tokens = [t for t in words if re.match(r'[^\W\d]\*$', t)]

clean\_str = ' '.join(clean\_tokens)

clean\_words = [w for w in clean\_str.split() if w.lower() not in stopwords.words('english')]

return clean\_words

clean\_tweet = no\_user\_sym(new\_tweet)

def normalize(words\_list):

lem = WordNetLemmatizer()

norm\_words = []

for word in words\_list:

norm\_word = lem.lemmatize(word, 'v')

norm\_words.append(norm\_word)

return norm\_words

return normalize(clean\_tweet)

**Download Necessary NLTK Data:**

python

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('punkt\_tab')

**Applying Text Processing:**

python

train\_tweets['processed\_tweet'] = train\_tweets['tweet'].apply(text\_proc)

test\_tweets['processed\_tweet'] = test\_tweets['tweet'].apply(text\_proc)

**7. Model Training and Evaluation**

A machine learning pipeline was created using the Naive Bayes classifier. The model was trained on the training dataset and evaluated on the test dataset.

**Split the Data:**

python

msg\_train, msg\_test, label\_train, label\_test = train\_test\_split(train\_tweets['tweet'], train\_tweets['label'], test\_size=0.2)

**Machine Learning Pipeline:**

python

pipeline = Pipeline([

('bow', CountVectorizer(analyzer=text\_proc)), # strings to token integer counts

('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores

('classifier', MultinomialNB()), # train on TF-IDF vectors w/ Naive Bayes classifier

])

pipeline.fit(msg\_train, label\_train)

predictions = pipeline.predict(msg\_test)

**Model Evaluation:**

python

print(classification\_report(predictions, label\_test))

print(confusion\_matrix(predictions, label\_test))

print(f"Model Accuracy: {accuracy\_score(predictions, label\_test):.2f}")

**Example Outputs:**

**Model Accuracy:**

Model Accuracy: 0.94

**Confusion Matrix:**

[[5927 394]

[ 0 72]]

**Classification Report:**

precision recall f1-score support

0 1.00 0.94 0.97 6321

1 0.15 1.00 0.27 72

accuracy 0.94 6393

macro avg 0.58 0.97 0.62 6393

weighted avg 0.99 0.94 0.96 6393

**8. Results**

**Model Performance:**

* **Accuracy:** 0.94

**Confusion Matrix:**

[[5927 394]

[ 0 72]]

**Classification Report:**

precision recall f1-score support

0 1.00 0.94 0.97 6321

1 0.15 1.00 0.27 72

accuracy 0.94 6393

macro avg 0.58 0.97 0.62 6393

weighted avg 0.99 0.94 0.96 6393

**9. Conclusion**

The Naive Bayes classifier provided a solid performance with an accuracy of 94%. However, the low precision for positive sentiment suggests that the model may need further tuning or additional data to improve its predictive power.

**Future Work:**

* Explore more sophisticated models such as LSTM or BERT for better performance.
* Perform hyperparameter tuning to optimize model performance.
* Incorporate more diverse datasets to improve model generalization.

**10. References**

* Pandas Documentation
* Matplotlib Documentation
* Seaborn Documentation
* NLTK Documentation
* Scikit-Learn Documentation