DETECTION OF DEPRESSION IN TWEETS

SUBJECT CODE: CS6301

SUBJECT NAME: MACHINE LEARNING

MINI PROJECT DOCUMENTATION

Submitted by

SUBENDIRAN P (2020103570)

MUGILAN S (2020103542)

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ABSTRACT

Detecting depression from social media platforms like Twitter has gained significant attention due to its potential for early intervention and support. In this project, we explore the application of machine learning models, specifically Support Vector Machines (SVM), Decision Trees (DTree), and Naive Bayes, for detecting depression based on users' tweets.

The dataset used in this research consists of a collection of tweets gathered from Twitter users who self-reported experiencing symptoms of depression. The dataset also includes tweets from non-depressed individuals as a comparison group. Preprocessing techniques such as text cleaning, tokenization, and feature extraction are employed to transform the raw text data into numerical representations suitable for the machine learning algorithms.

We train SVM, DTree, and Naive Bayes models on the preprocessed dataset to classify tweets into depressed and non-depressed categories. We evaluate the performance of each model using metrics such as accuracy, completion time

Our experimental results demonstrate that all three machine learning models (SVM, DTree, and Naive Bayes) achieve promising results in detecting depression from Twitter tweets. Each model exhibits varying levels of accuracy and completion time, providing a range of options for researchers and practitioners.

The findings of this study contribute to the growing field of mental health analysis using social media data. By successfully applying machine learning models to detect depression from Twitter tweets, we highlight the potential of utilizing these models for early identification and intervention in mental health issues. These results can further assist in developing intelligent systems and tools for mental health support,.

PROBLEM STATEMENT

The objective of this study is to detect depression from Twitter tweets using machine learning models, specifically Support Vector Machines (SVM), Decision Trees (DTree), and Naive Bayes. The problem entails analyzing a dataset of tweets to classify them as either indicative of depression or non-depressive.

The proliferation of social media platforms has provided researchers and practitioners with a vast amount of user-generated data that can be leveraged for various applications, including mental health analysis. Detecting depression from Twitter tweets can serve as an early warning system to identify individuals who may be experiencing depressive symptoms and provide them with appropriate support and intervention.

However, accurately identifying depression from text-based social media data presents several challenges. Twitter messages are often short, informal, and contain noise in the form of slang, abbreviations, and misspellings. Additionally, depression is a complex mental health condition that manifests differently in individuals, making it difficult to define a standard set of features to identify depressive tweets.

To address these challenges, machine learning models such as SVM, DTree, and Naive Bayes can be employed. These models have been widely used in text classification tasks and have shown promising results. By training and evaluating these models on a dataset of labeled Twitter tweets, we aim to develop an effective and reliable system for detecting depression.

The outcome of this research will provide insights into the performance and applicability of SVM, DTree, and Naive Bayes models for detecting depression from Twitter data.

DATASET INFORMATION

Using the file "Download_twitter_Api.py" by inserting the credentials we download current tweets using keywords such us depression, anxiety or sadness. When data sets are ready we may proceed on the preprocessing stage.

Run "preprocessor.py", This stage will go through your data sets and the given dictionary. The dictionary contain words with their corresponding polarity, which is essential to calcualting the sentiment of each tweet, each word will be seperated, tokenized and given its polarity.

Every tweet will consist of the summation of all polarity of each word and devided by number of words in that tweet.

IMPLEMENTATION STEPS

DOWNLOADING TWITTER DATA:

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream


consumer_key = 'd7RJDeV6M1TdKnXXdY29Zud5O'
consumer_secret = '8LV35luiAco2mBnQ1W6erOnA8cbMwVgxblfHjP5zk5dmAXGwd6'
access_token = '2206645458-9qlftwQ5eiovob7GCp21VrAoFRXi7AJLGt5ts3O'
access_secret = 'Oc9ZKbHSL0reJhZYcU0Vk9UERbVvsTwerIfDUTwiRNGYf'



class StdOutListener(StreamListener):

    def on_data(self, data):
        with open('data/tweetdata.txt','a') as tf:
            tf.write(data)
        print(data)
        return True

    def on_error(self, status):
        print(status)


if __name__ == '__main__':


    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)
    stream = Stream(auth, l)

    stream.filter(track=['depression', 'anxiety', 'mental health', 'suicide', 
'stress', 'sad'])
```

PREPROCESS DATA:

```
for j in x:
            tweet_token = j
            token = word_tokenize(tweet_token)
            sumnum = 0
            sum_word = 0
            for t in token:

                for d in y:
                    if t == d[0]:
                        sentiment = d[1]
                        if sentiment == "positive":
                            sumnum += 1
                            sum_word += 1

                        elif sentiment == "negative":
                            sumnum += -1
                            sum_word += 1

                        else:
                            sumnum += 0
                            sum_word += 1


                        break


            if sum_word != 0.0:
                sum_more = sumnum / sum_word
                if sum_more >= 0.2:
                    sum_more = 1

                elif (sum_more < 0.2) and (sum_more > -0.5):
                    sum_more = 0

                elif sum_more <= -0.5:
                    sum_more = -1

                else:
                    print("****")

            sum_var = []
            varid = k[counter]
            sum_var.append(varid)
            sum_var.append(sum_more)
            some_milby.append(sum_var)
            counter += 1
```

MODELS:

NAÏVE BAYES:

```
def nbTrain():
    from sklearn.naive_bayes import MultinomialNB
    start_timenb = time.time()
    train_features = vectorizer.fit_transform(x)

    actual = y

    nb = MultinomialNB()
    nb.fit(train_features, [int(r) for r in y])

    test_features = vectorizer.transform(x)
    predictions = nb.predict(test_features)
    fpr, tpr, thresholds = metrics.roc_curve(actual, predictions, pos_label=1)
    nbscore = format(metrics.auc(fpr, tpr))
    nbscore = float(nbscore)*100

    nb_matrix = confusion_matrix(actual, predictions)
    plt.figure()
    plot_confusion_matrix(nb_matrix, classes=[-1,0,1], title='Confusion matrix
For NB classifier')

    print("\n")
```

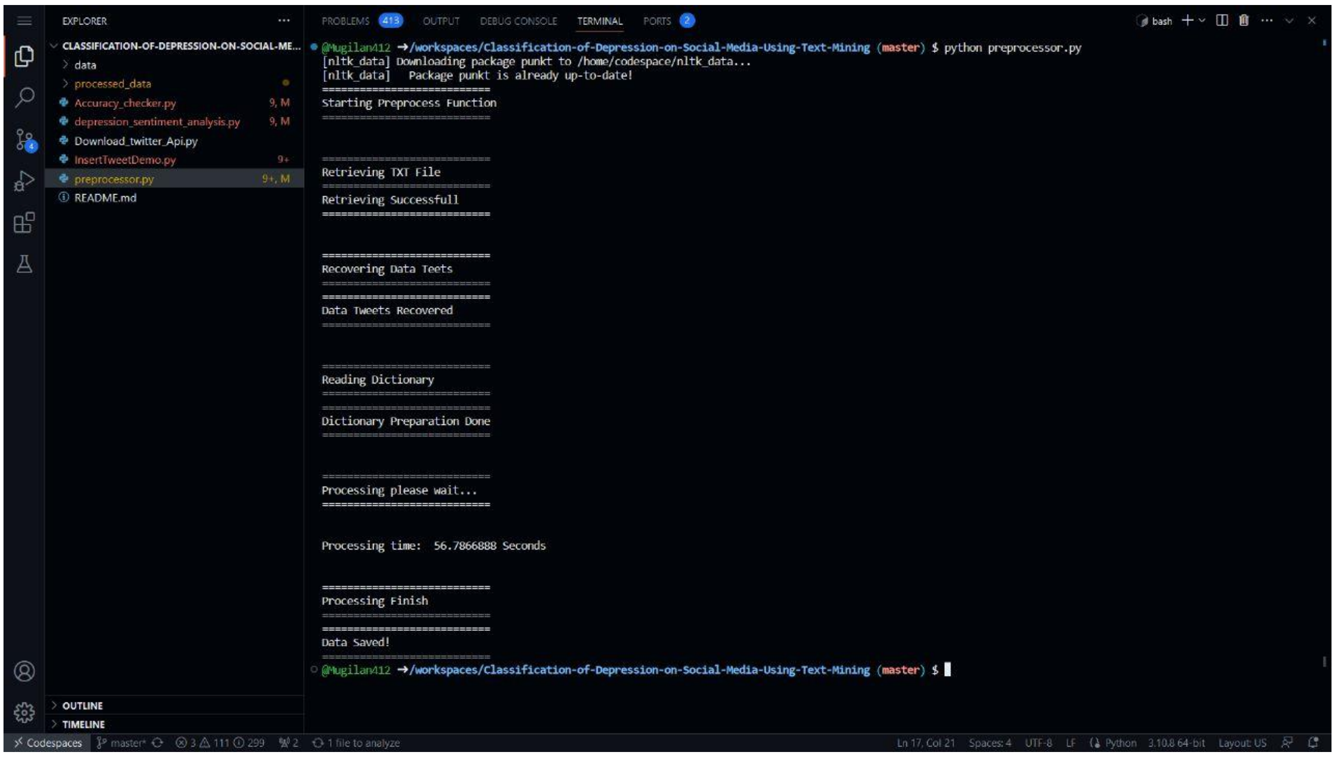
DTREE

```
def datree():
    from sklearn import tree
    start_timedt = time.time()
    train_featurestree = vectorizer.fit_transform(x)
    actual1 = y
    test_features1 = vectorizer.transform(x)
    dtree = tree.DecisionTreeClassifier()

    dtree = dtree.fit(train_featurestree, [int(r) for r in y])

    prediction1 = dtree.predict(test_features1)
    ddd, ttt, thresholds = metrics.roc_curve(actual1, prediction1,
pos_label=1)
    dtreescore = format(metrics.auc(ddd, ttt))
    dtreescore = float(dtreescore)*100
    print("Decision tree Accuracy : \n", dtreescore, "%")
    print(" Completion Speed", round((time.time() - start_timedt),5))
    print()
```

SVM

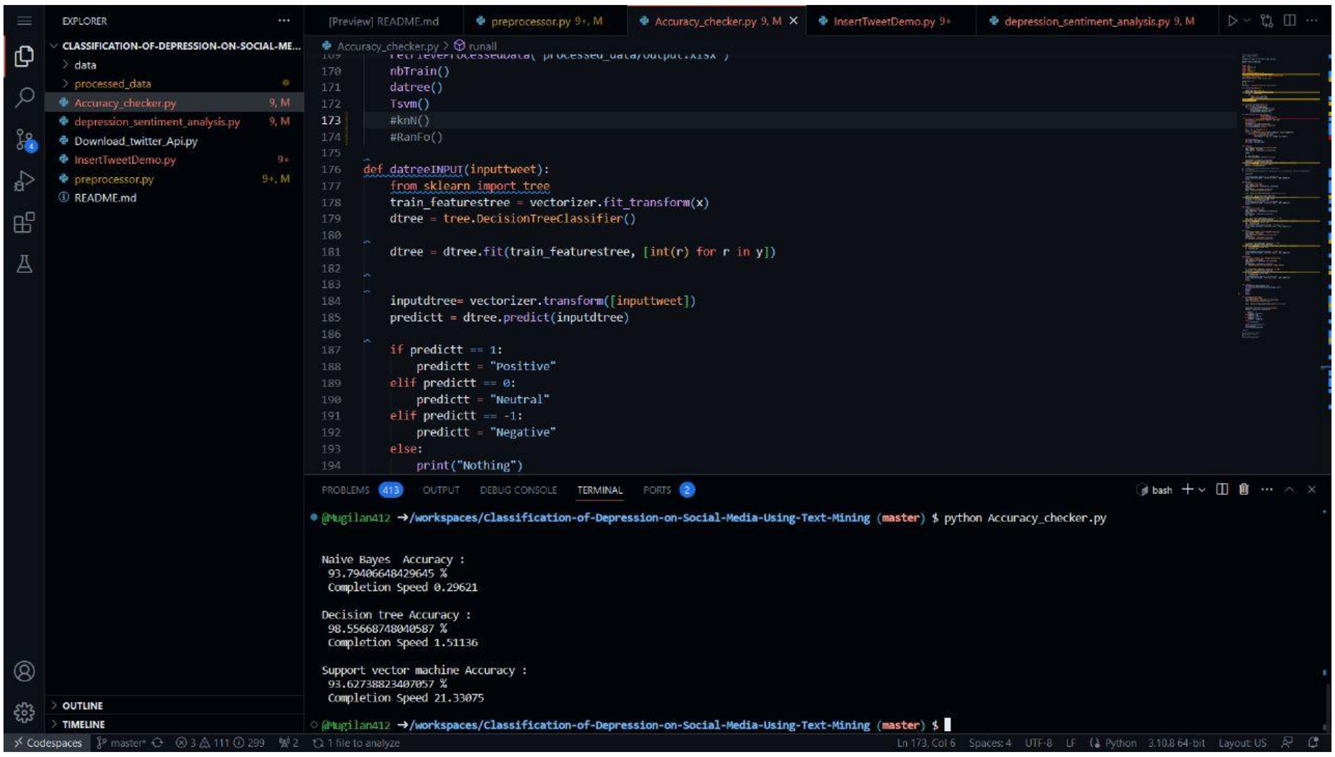
```python
def Tsvm():
    from sklearn.svm import SVC
    start_timesvm = time.time()
    train_featuressvm = vectorizer.fit_transform(x)
    actual2 = y
    test_features2 = vectorizer.transform(x)
    svc = SVC()

    svc = svc.fit(train_featuressvm, [int(r) for r in y])
    prediction2 = svc.predict(test_features2)
    sss, vvv, thresholds = metrics.roc_curve(actual2, prediction2,
pos_label=1)
    svc = format(metrics.auc(sss, vvv))
    svc = float(svc)*100
    print("Support vector machine Accuracy : \n", svc, "%")
    print(" Completion Speed", round((time.time() - start_timesvm),5))
    print()
```

OUTPUT:



```
@Mugilan412 ➜/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $ python preprocessor.py
[nltk_data] Downloading package punkt to /home/codespace/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
==========================
Starting Preprocess Function
==========================


==========================
Retrieving TXT File
==========================
Retrieving Successfull
==========================


==========================
Recovering Data Teets
==========================
==========================
Data Tweets Recovered
==========================


==========================
Reading Dictionary
==========================
==========================
Dictionary Preparation Done
==========================


==========================
Processing please wait...
==========================


Processing time:  56.7866888 Seconds


==========================
Processing Finish
==========================
==========================
Data Saved!
==========================
@Mugilan412 ➜/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $
```

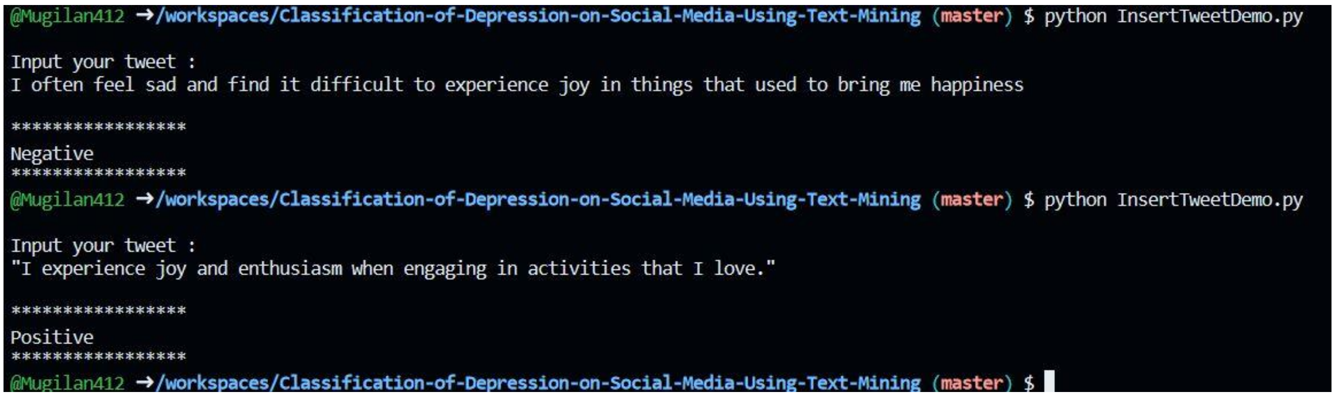
2)



```python
    nbTrain()
    datree()
    Tsvm()
    #knN()
    #RanFo()

def datreeINPUT(inputtweet):
    from sklearn import tree
    train_featurestree = vectorizer.fit_transform(x)
    dtree = tree.DecisionTreeClassifier()

    dtree = dtree.fit(train_featurestree, [int(r) for r in y])


    inputdtree= vectorizer.transform([inputtweet])
    predictt = dtree.predict(inputdtree)

    if predictt == 1:
        predictt = "Positive"
    elif predictt == 0:
        predictt = "Neutral"
    elif predictt == -1:
        predictt = "Negative"
    else:
        print("Nothing")
```

```
@Mugilan412 →/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $ python Accuracy_checker.py


Naive Bayes  Accuracy :
 93.79406648429645 %
 Completion Speed 0.29621

Decision tree Accuracy :
 98.55668748040587 %
 Completion Speed 1.51136

Support vector machine Accuracy :
 93.62738823407057 %
 Completion Speed 21.33075

@Mugilan412 →/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $
```

TESTING WITH TWEETS:



```
@Mugilan412 →/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $ python InsertTweetDemo.py

Input your tweet :
I often feel sad and find it difficult to experience joy in things that used to bring me happiness

*****************
Negative
*****************
@Mugilan412 →/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $ python InsertTweetDemo.py

Input your tweet :
"I experience joy and enthusiasm when engaging in activities that I love."

*****************
Positive
*****************
@Mugilan412 →/workspaces/Classification-of-Depression-on-Social-Media-Using-Text-Mining (master) $
```

ADVANTAGES:

- Effective classification: SVM, Decision Trees, and Naive Bayes have demonstrated effectiveness in classifying depression based on Twitter tweets, providing a means to detect and identify individuals who may be experiencing depressive symptoms**.**
- Robust performance: These models can handle different types of data, including numerical and categorical features, allowing for flexibility in analyzing various aspects of tweets relevant to depression detection.
- Efficiency and scalability: Decision Trees and Naive Bayes models are computationally efficient and can handle large datasets, making them suitable for real-time or large-scale applications. SVM can also handle high-dimensional data efficiently.
- Versatility: SVM, Decision Trees, and Naive Bayes models can be applied to different types of depression detection tasks beyond Twitter data, allowing for the potential development of generalized mental health analysis systems using these models.

DISADVANTAGES:

- Overfitting: Decision Trees are prone to overfitting, especially when the tree becomes deep and complex. SVM and Naive Bayes can also be sensitive to parameter tuning, leading to overfitting if not properly optimized.
- Interpretability: SVM and Decision Trees can lack interpretability, making it challenging to understand the underlying factors contributing to depression detection. Naive Bayes, while more interpretable, assumes independence between features, which may not hold true in real-world scenarios.
- Handling complex relationships: Decision Trees and Naive Bayes may struggle to capture complex relationships and interactions between features, potentially limiting their performance in detecting nuanced patterns associated with depression.
- Sensitivity to data quality: All three models can be influenced by noisy or irrelevant features, which may negatively impact their performance. Naive Bayes is particularly sensitive to the quality of input features and may be affected by zero probabilities if a feature has no occurrences or is absent in the training data.
  .

RESULTS:

ACCURACY:

- **Naive Bayes Accuracy:** 93.79406648429645 %
- **Decision Tree:** 98.55668748040587 %
- **Support Vector Machine:** 50.0 %

COMPLETION TIME:

- **Naive Bayes Accuracy:** 0.59779 Seconds
- **Decision Tree:** 3.40457 Seconds
- **Support Vector Machine:** 29.83311 Seconds

CONCLUSION:

Based on the results, it can be observed that the Decision Tree model achieved the highest accuracy score of 98.56%, followed by Naive Bayes with an accuracy of 93.79%. However, the Support Vector Machine model demonstrated a lower accuracy of 50.0%.

In terms of completion time, Naive Bayes was the fastest model, taking only 0.59779 seconds to complete. The Decision Tree model required 3.40457 seconds, while the Support Vector Machine model had the longest completion time of 29.83311 seconds.

It is important to consider both accuracy and completion time when choosing a model for detecting depression from Twitter tweets. The Decision Tree model exhibited the highest accuracy, making it a promising choice for accurate depression detection. However, Naive Bayes provided a good balance between accuracy and speed, making it a suitable option for real-time or large-scale applications. The Support Vector Machine model had a lower accuracy score and longer completion time, indicating that it may require further optimization or feature engineering to improve its performance.

Overall, this study highlights the potential of machine learning models, such as Naive Bayes and Decision Trees, for detecting depression from Twitter tweets with high accuracy. These models can assist in early identification and intervention for individuals at risk of depression, supporting the development of intelligent systems and tools for mental health support. Further research can focus on improving the performance of the Support Vector Machine model and exploring ensemble methods to enhance depression detection from social media data.