College: Sri Venkateswara college of Engineering & Technology(autonomous)

Branch: CSE(AI&ML)

Batch.No:34(4)

Course Name: Big Data Analytics

Project Name: Emotion Detection in Text

Group Members:

1. TALARI NITHIN
2. TARIGONDA RAJESH
3. TEKALAPALEM DIVAKAR REDDY
4. SUDHEER PULLURU

Resources:

- Kaggle

Tools:

- Jupyter Notebook
- Word
- Power Point Presentation

Technologies:

- Python
- NLTK
- Scikit-learn

INTRODUCTION

This project is designed to predict emotions from textual data using Machine Learning (ML) techniques. The model is trained on a dataset containing text samples labeled with emotions such as joy, sadness, anger, and fear. It uses Natural Language Processing (NLP) methods to process and vectorize the text, then applies a machine learning algorithm to classify the emotions in new, unseen text inputs.

The project is built using Python and popular libraries such as **scikit-learn** for machine learning, and **TfidfVectorizer** for text preprocessing. The trained model can accurately classify emotions based on the content of a sentence or paragraph, making it applicable for emotion detection in text.

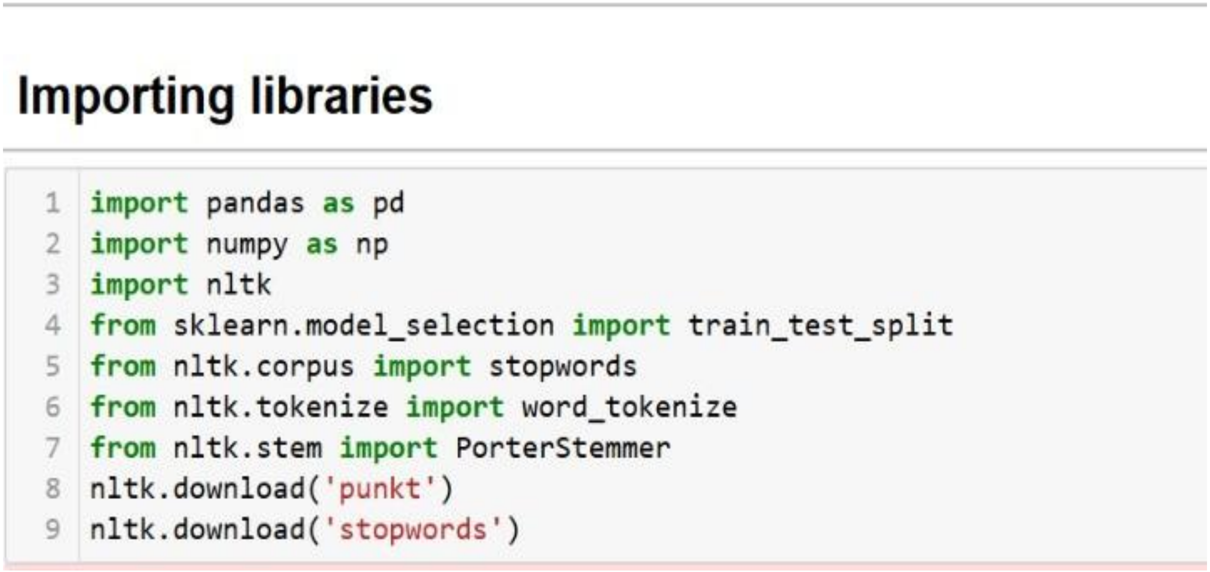
Goal of the project:

In this project, The primary goal t is to develop an **emotion detection system** capable of analyzing text and classifying it into different emotional categories, such as **joy, sadness, anger**, and **fear**. By leveraging **Machine Learning (ML)** algorithms and **Natural Language Processing (NLP)** techniques, the system will enable automated understanding of emotions expressed in written content.

About Dataset:

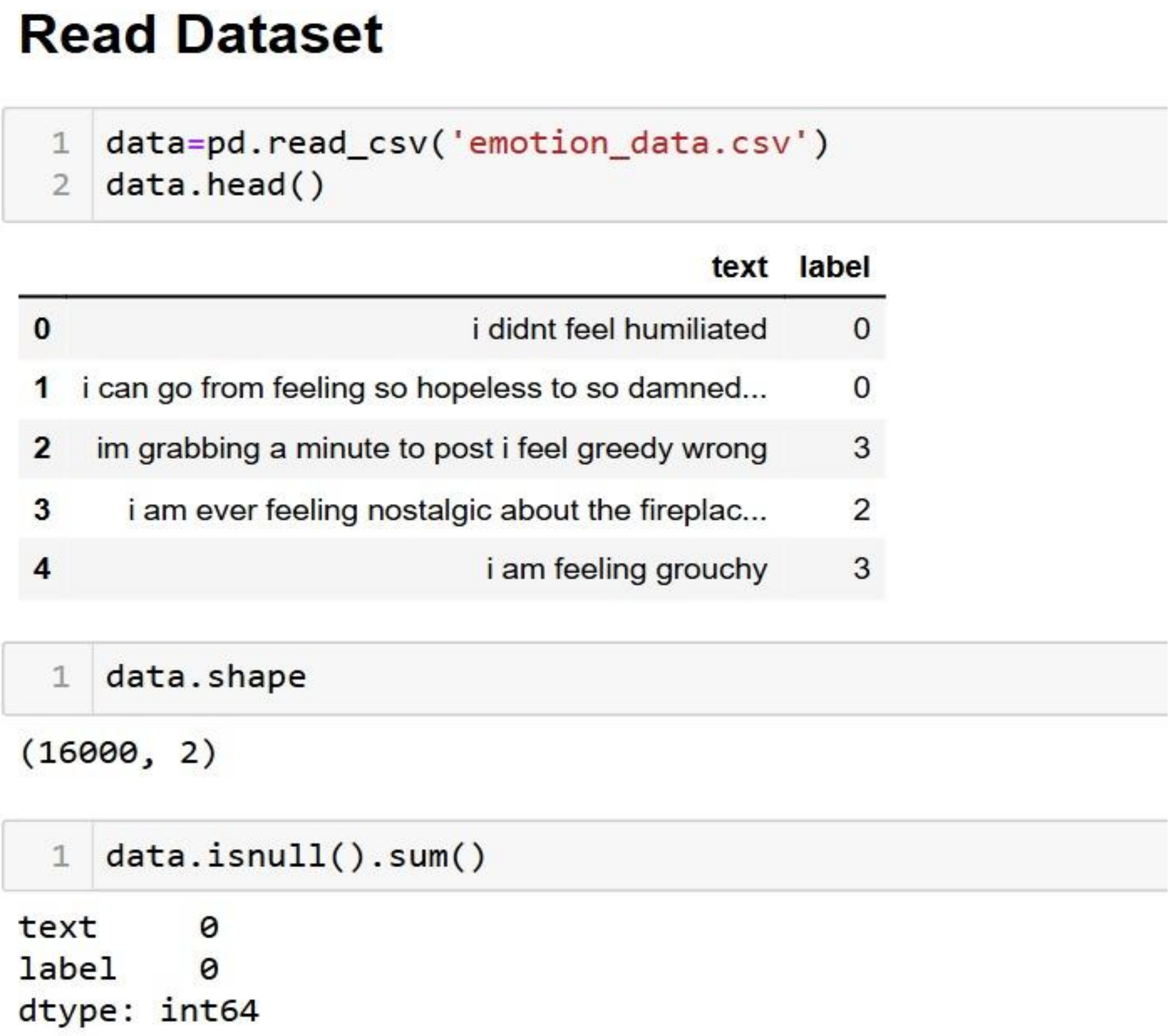
This dataset includes data on text and emotion labels which defines the emotion of text. The dataset is downloaded from the Kaggle.

Project Development:



Importing libraries

```
import pandas as pd
import numpy as np
import nltk
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
nltk.download('punkt')
nltk.download('stopwords')
```



Read Dataset

```
data=pd.read_csv('emotion_data.csv')
data.head()
```

	text	label
0	i didnt feel humiliated	0
1	i can go from feeling so hopeless to so damned...	0
2	im grabbing a minute to post i feel greedy wrong	3
3	i am ever feeling nostalgic about the fireplac...	2
4	i am feeling grouchy	3

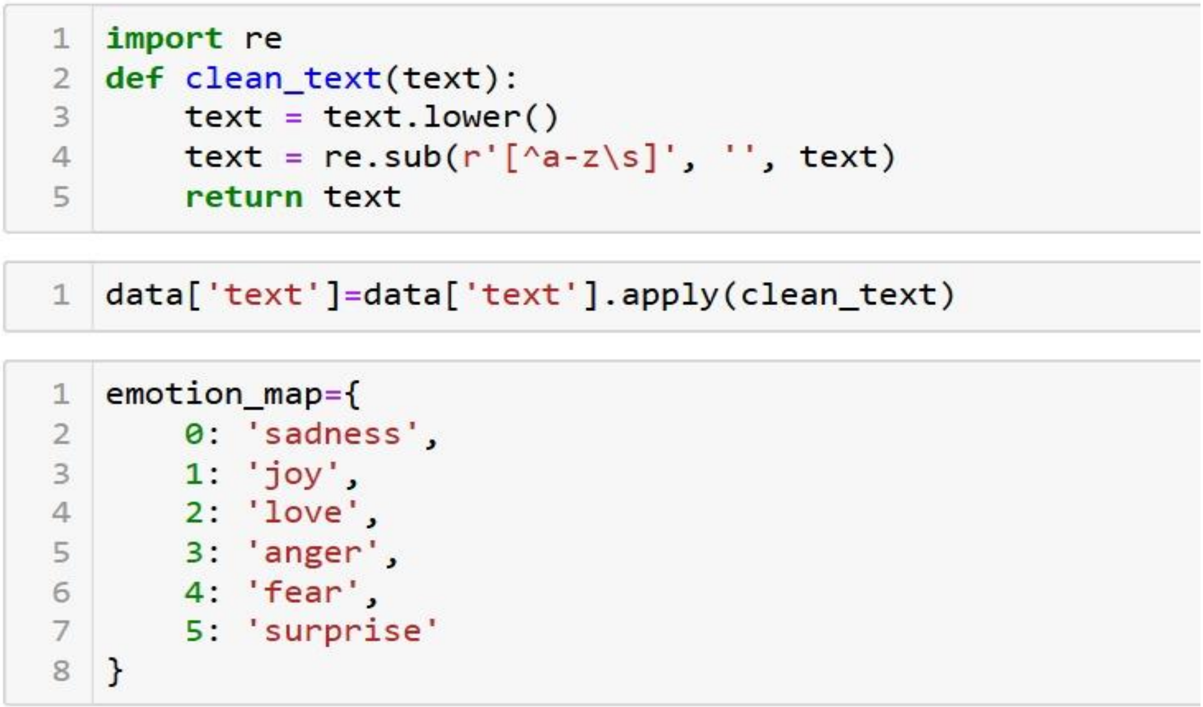
```
data.shape
```

```
(16000, 2)
```

```
data.isnull().sum()
```

```
text     0
label    0
dtype: int64
```

Clean the text

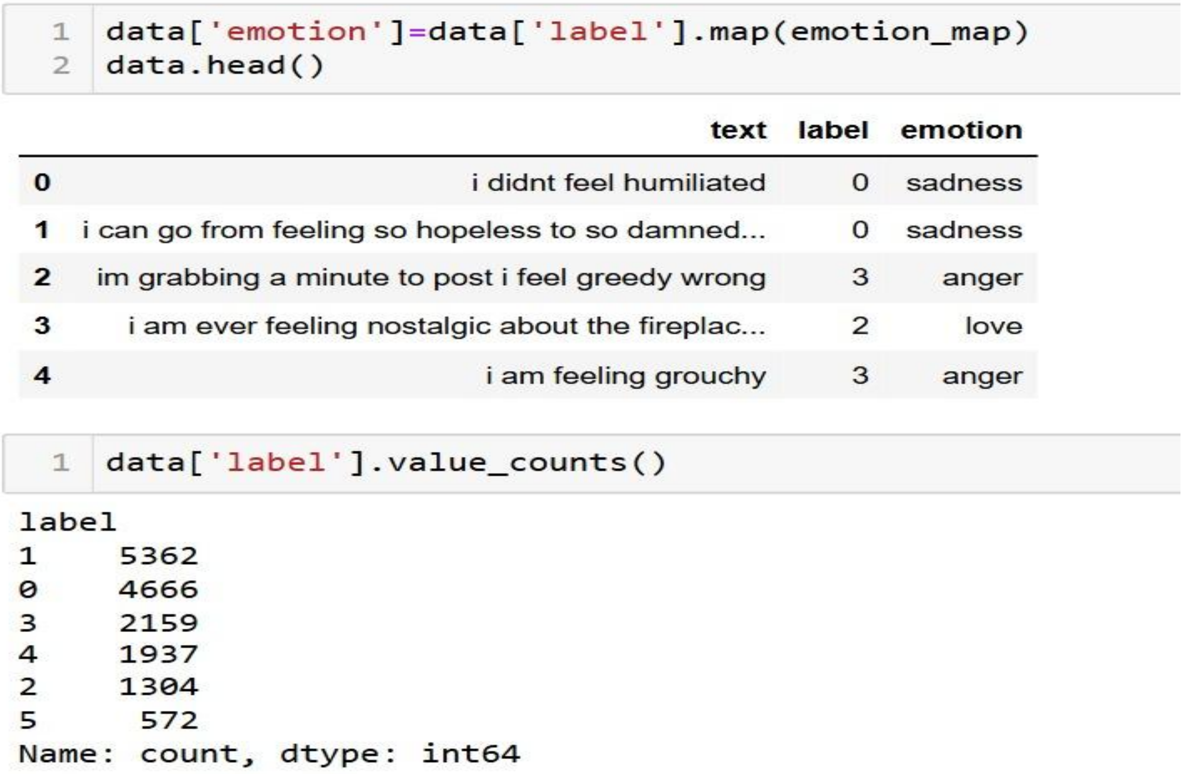


```python
import re
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)
    return text
```

```python
data['text']=data['text'].apply(clean_text)
```

```python
emotion_map={
    0: 'sadness',
    1: 'joy',
    2: 'love',
    3: 'anger',
    4: 'fear',
    5: 'surprise'
}
```

Mapping the emotion_map to label

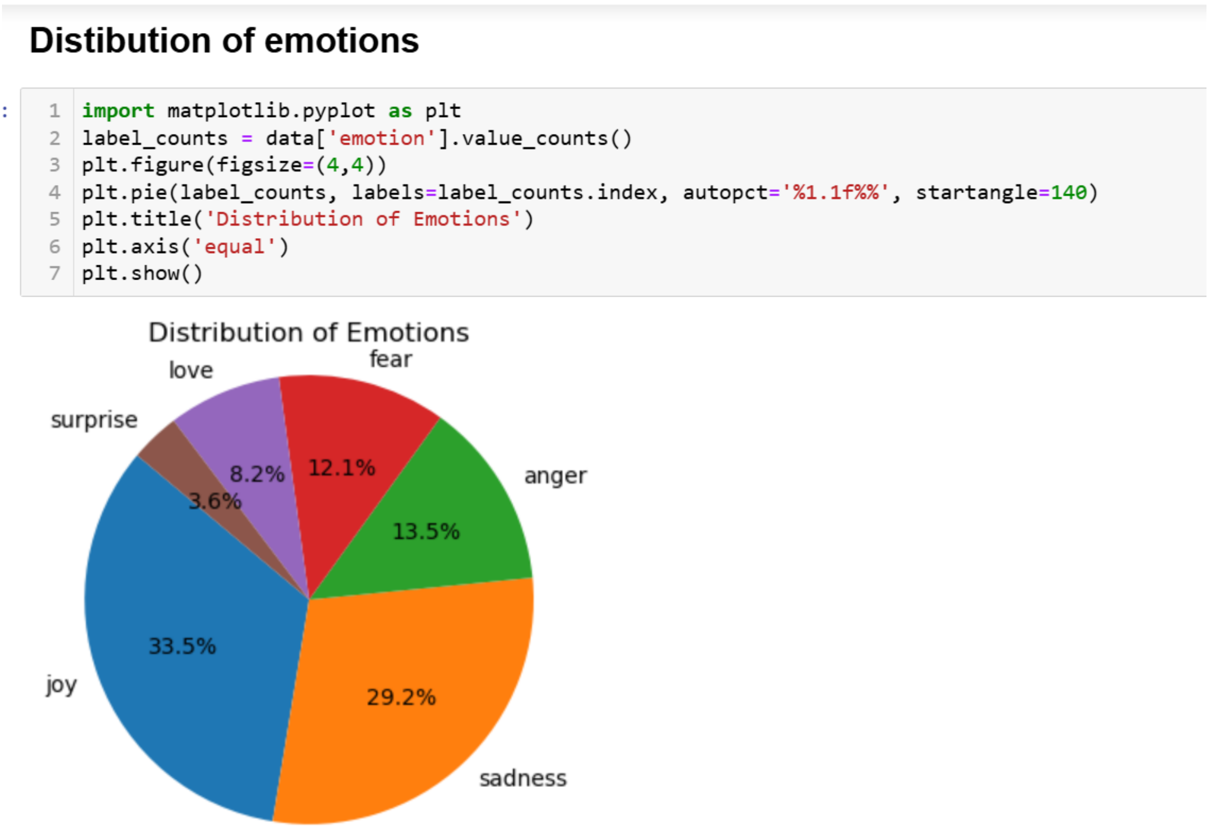


```python
data['emotion']=data['label'].map(emotion_map)
data.head()
```

	text	label	emotion
0	i didnt feel humiliated	0	sadness
1	i can go from feeling so hopeless to so damned...	0	sadness
2	im grabbing a minute to post i feel greedy wrong	3	anger
3	i am ever feeling nostalgic about the fireplac...	2	love
4	i am feeling grouchy	3	anger

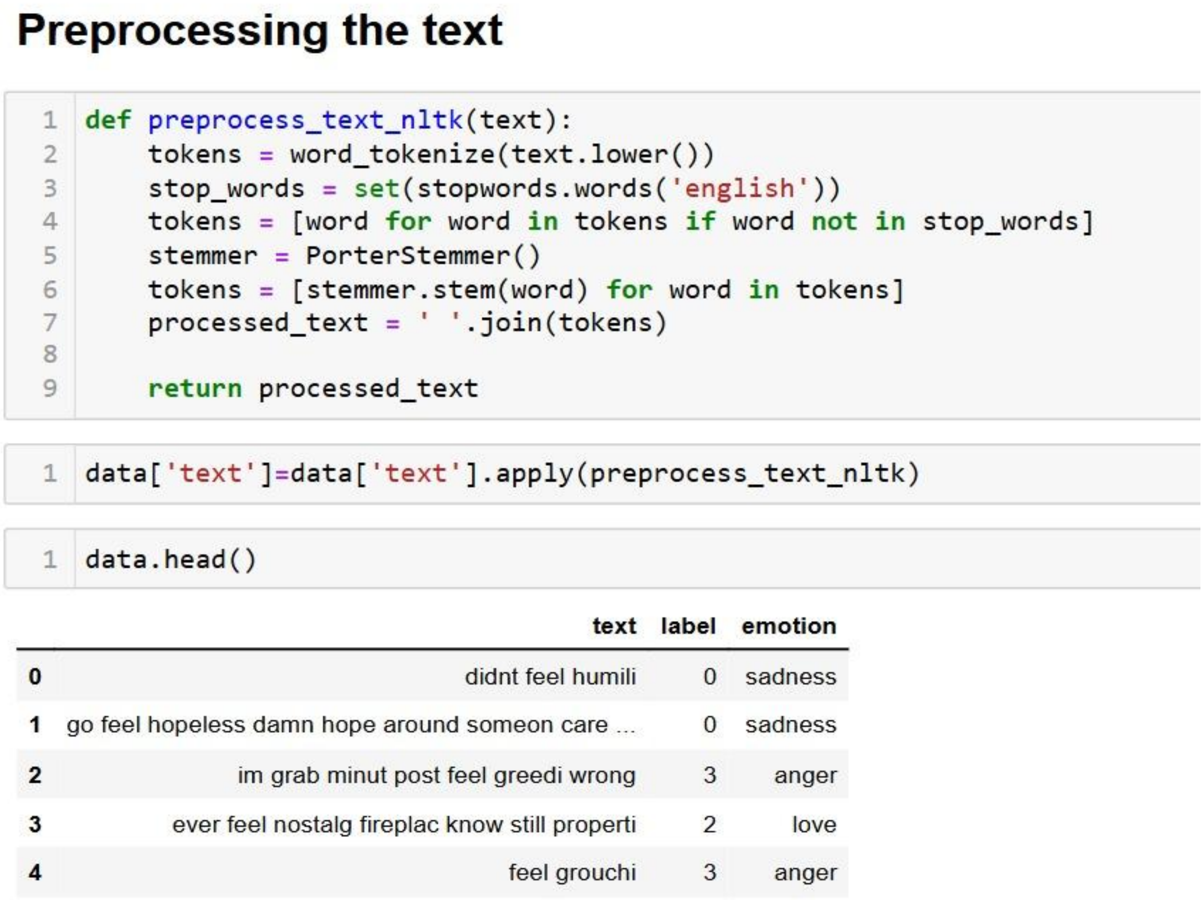
```python
data['label'].value_counts()
```

```
label
1    5362
0    4666
3    2159
4    1937
2    1304
5     572
Name: count, dtype: int64
```



Distibution of emotions

```python
import matplotlib.pyplot as plt
label_counts = data['emotion'].value_counts()
plt.figure(figsize=(4,4))
plt.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Emotions')
plt.axis('equal')
plt.show()
```



Preprocessing the text

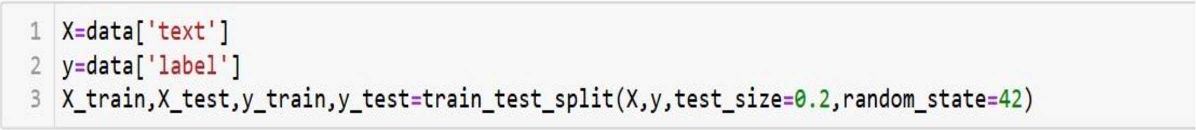
```python
def preprocess_text_nltk(text):
    tokens = word_tokenize(text.lower())
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    stemmer = PorterStemmer()
    tokens = [stemmer.stem(word) for word in tokens]
    processed_text = ' '.join(tokens)

    return processed_text
```

```python
data['text']=data['text'].apply(preprocess_text_nltk)
```

```python
data.head()
```

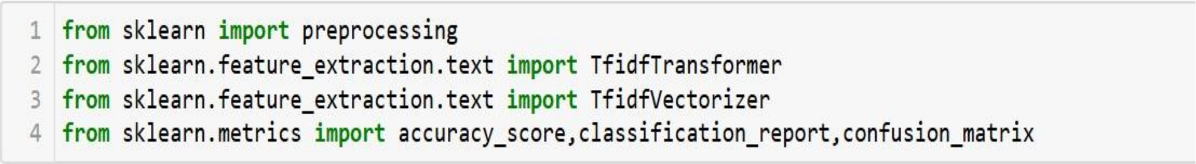
	text	label	emotion
0	didnt feel humili	0	sadness
1	go feel hopeless damn hope around someon care ...	0	sadness
2	im grab minut post feel greedi wrong	3	anger
3	ever feel nostalg fireplac know still properti	2	love
4	feel grouchi	3	anger

Splitting data into traing and testing data

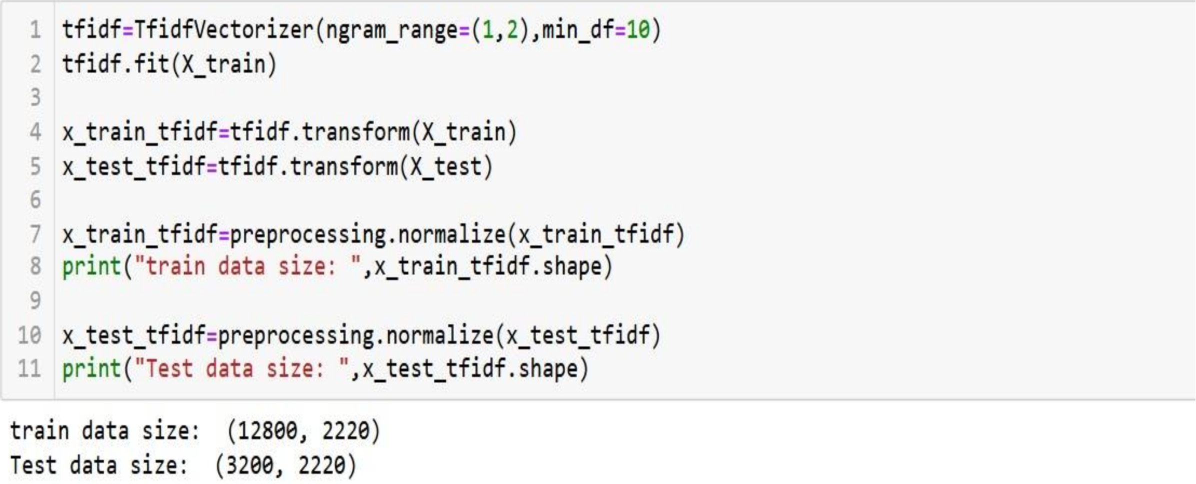


```
X=data['text']
y=data['label']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Feature Extraction



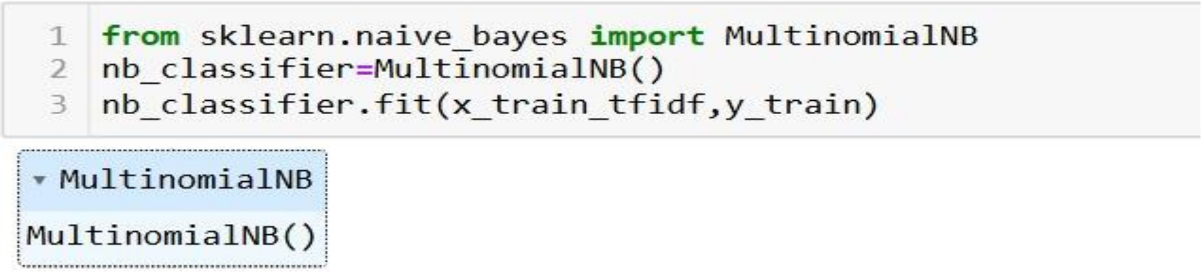
```
from sklearn import preprocessing
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```



```
tfidf=TfidfVectorizer(ngram_range=(1,2),min_df=10)
tfidf.fit(X_train)

x_train_tfidf=tfidf.transform(X_train)
x_test_tfidf=tfidf.transform(X_test)

x_train_tfidf=preprocessing.normalize(x_train_tfidf)
print("train data size: ",x_train_tfidf.shape)

x_test_tfidf=preprocessing.normalize(x_test_tfidf)
print("Test data size: ",x_test_tfidf.shape)
```

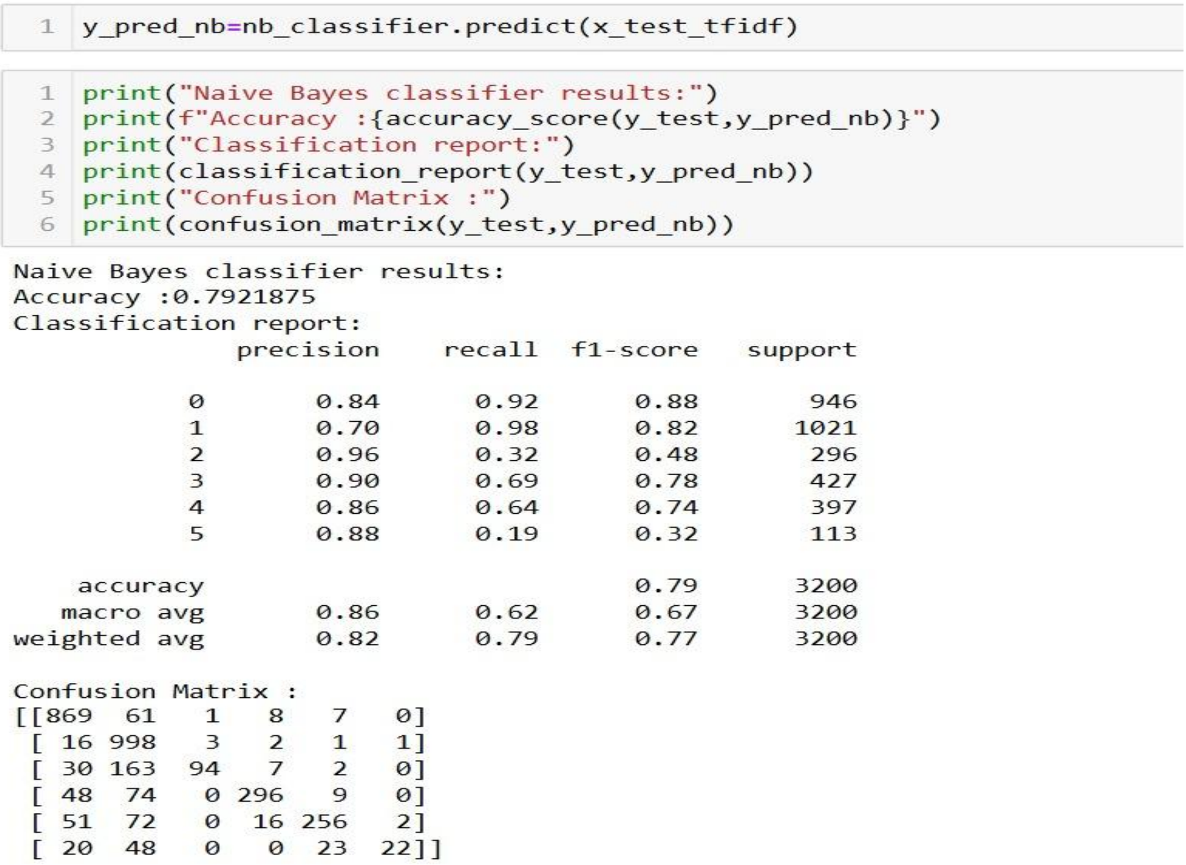
```
train data size:  (12800, 2220)
Test data size:  (3200, 2220)
```

MultinomialNB Model



```
from sklearn.naive_bayes import MultinomialNB
nb_classifier=MultinomialNB()
nb_classifier.fit(x_train_tfidf,y_train)
```

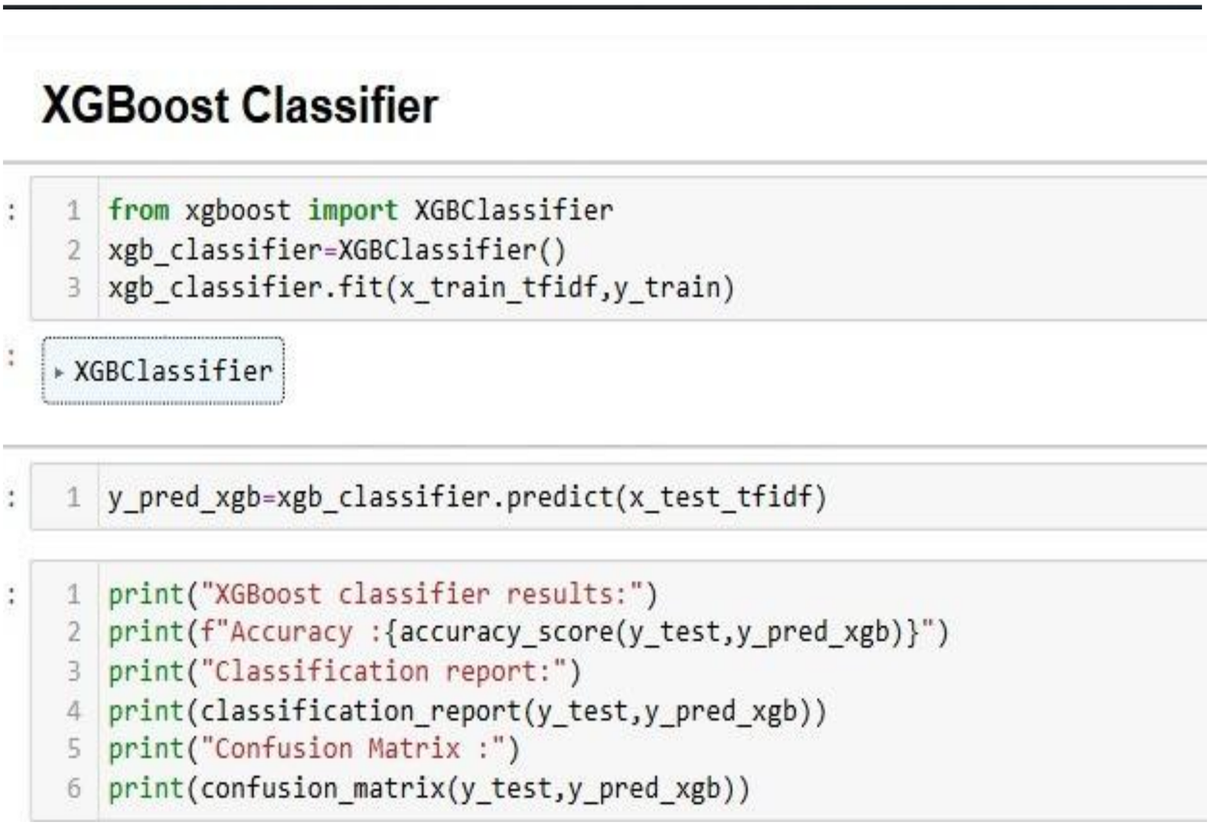
```
▾ MultinomialNB
MultinomialNB()
```



```
y_pred_nb=nb_classifier.predict(x_test_tfidf)
```

```
print("Naive Bayes classifier results:")
print(f"Accuracy :{accuracy_score(y_test,y_pred_nb)}")
print("Classification report:")
print(classification_report(y_test,y_pred_nb))
print("Confusion Matrix :")
print(confusion_matrix(y_test,y_pred_nb))
```

```
Naive Bayes classifier results:
Accuracy :0.7921875
Classification report:
              precision    recall  f1-score   support

           0       0.84      0.92      0.88       946
           1       0.70      0.98      0.82      1021
           2       0.96      0.32      0.48       296
           3       0.90      0.69      0.78       427
           4       0.86      0.64      0.74       397
           5       0.88      0.19      0.32       113

    accuracy                           0.79      3200
   macro avg       0.86      0.62      0.67      3200
weighted avg       0.82      0.79      0.77      3200

Confusion Matrix :
[[869  61   1   8   7   0]
 [ 16 998   3   2   1   1]
 [ 30 163  94   7   2   0]
 [ 48  74   0 296   9   0]
 [ 51  72   0  16 256   2]
 [ 20  48   0   0  23  22]]
```



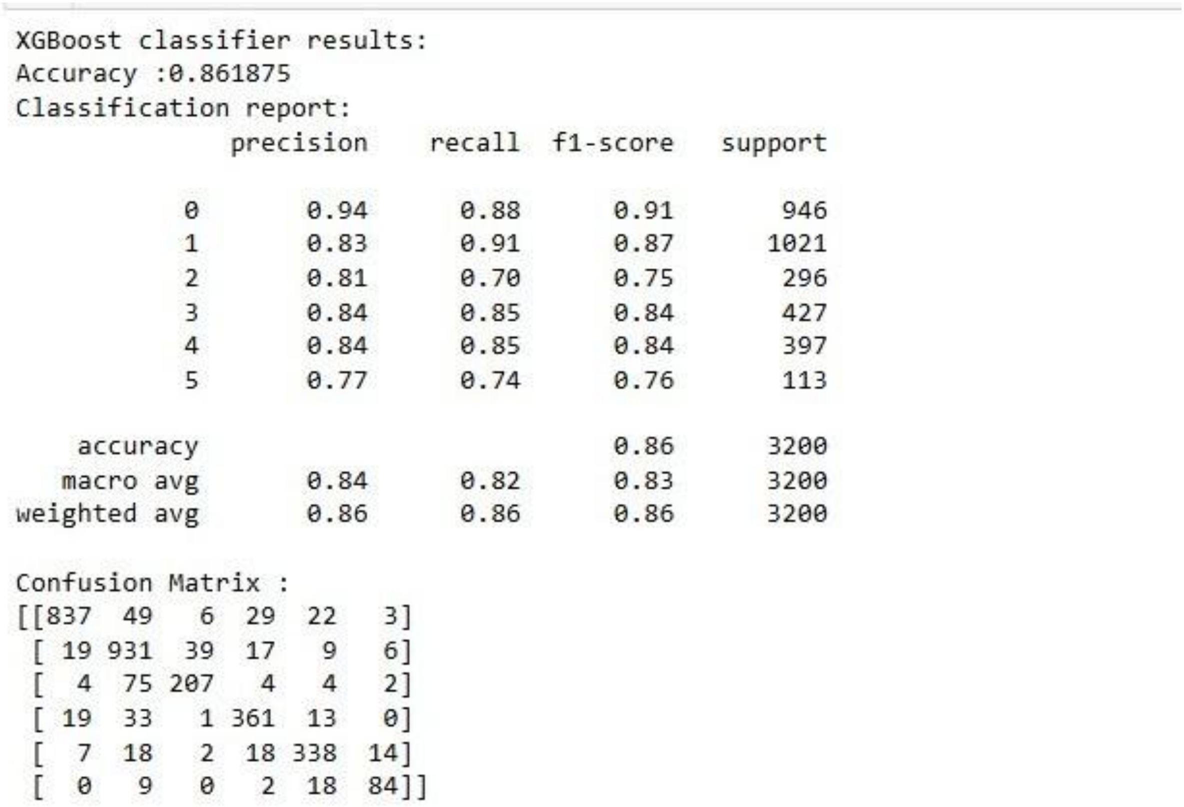
XGBoost Classifier

```
from xgboost import XGBClassifier
xgb_classifier=XGBClassifier()
xgb_classifier.fit(x_train_tfidf,y_train)
```

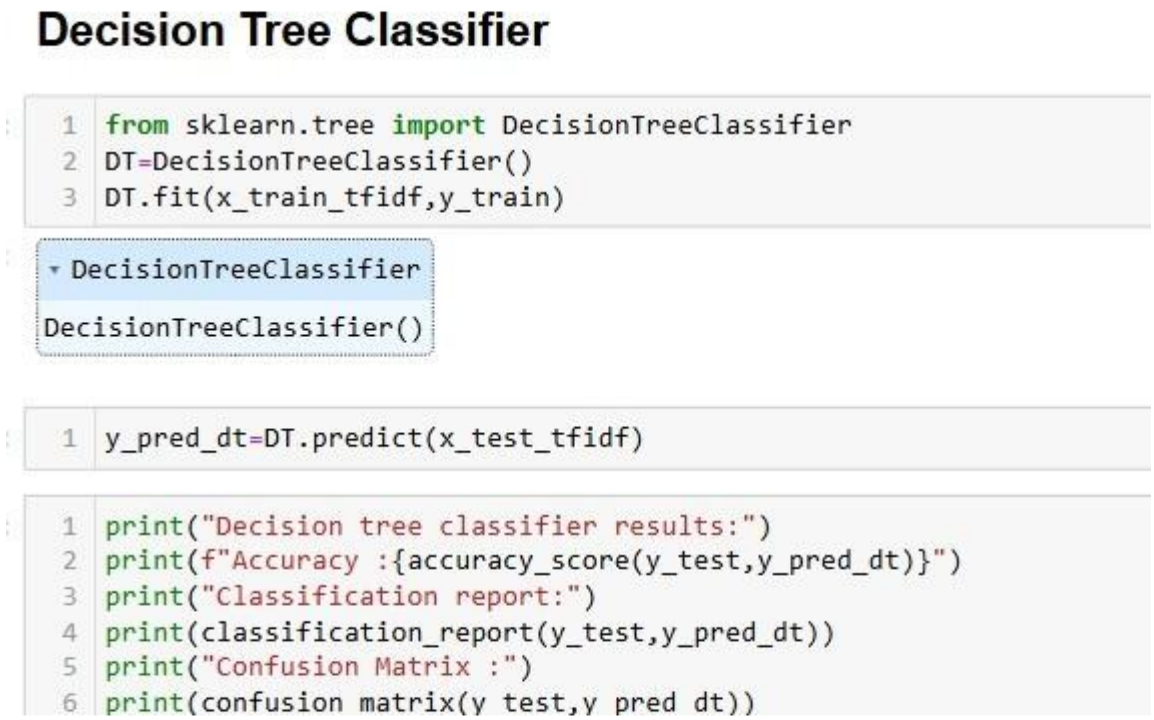
```
▸ XGBClassifier
```

```
y_pred_xgb=xgb_classifier.predict(x_test_tfidf)
```

```
print("XGBoost classifier results:")
print(f"Accuracy :{accuracy_score(y_test,y_pred_xgb)}")
print("Classification report:")
print(classification_report(y_test,y_pred_xgb))
print("Confusion Matrix :")
print(confusion_matrix(y_test,y_pred_xgb))
```



```
XGBoost classifier results:
Accuracy :0.861875
Classification report:
              precision    recall  f1-score   support

           0       0.94      0.88      0.91       946
           1       0.83      0.91      0.87      1021
           2       0.81      0.70      0.75       296
           3       0.84      0.85      0.84       427
           4       0.84      0.85      0.84       397
           5       0.77      0.74      0.76       113

    accuracy                           0.86      3200
   macro avg       0.84      0.82      0.83      3200
weighted avg       0.86      0.86      0.86      3200

Confusion Matrix :
[[837  49   6  29  22   3]
 [ 19 931  39  17   9   6]
 [  4  75 207   4   4   2]
 [ 19  33   1 361  13   0]
 [  7  18   2  18 338  14]
 [  0   9   0   2  18  84]]
```



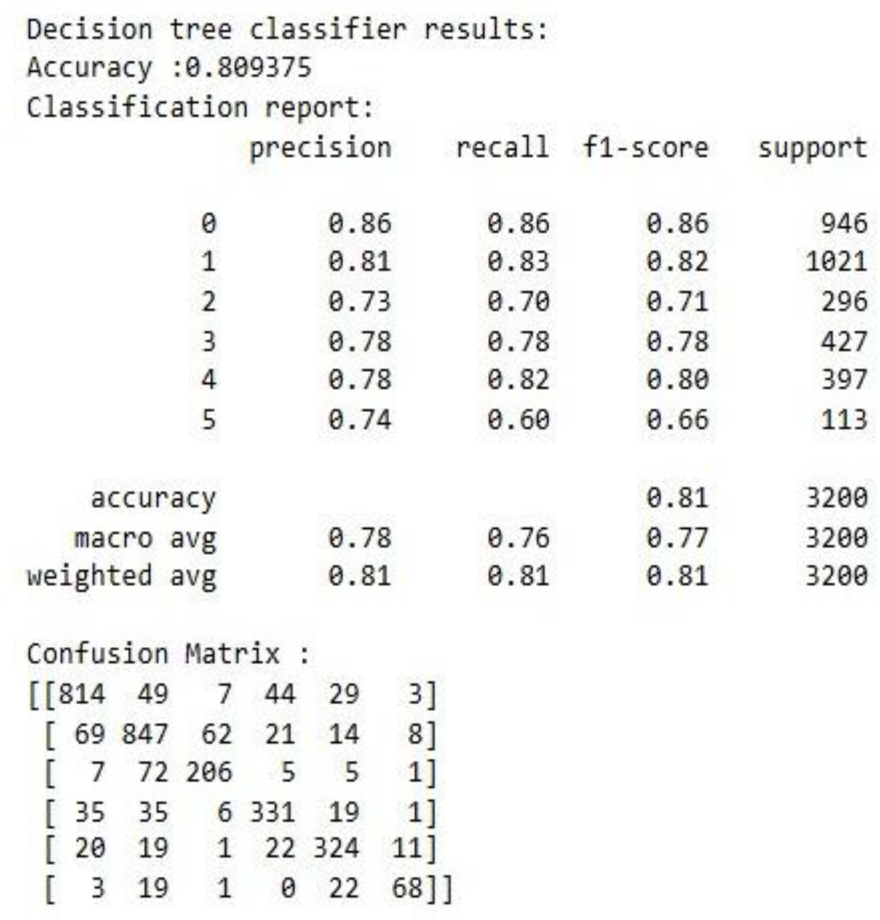
Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
DT=DecisionTreeClassifier()
DT.fit(x_train_tfidf,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred_dt=DT.predict(x_test_tfidf)
```

```
print("Decision tree classifier results:")
print(f"Accuracy :{accuracy_score(y_test,y_pred_dt)}")
print("Classification report:")
print(classification_report(y_test,y_pred_dt))
print("Confusion Matrix :")
print(confusion_matrix(y_test,y_pred_dt))
```

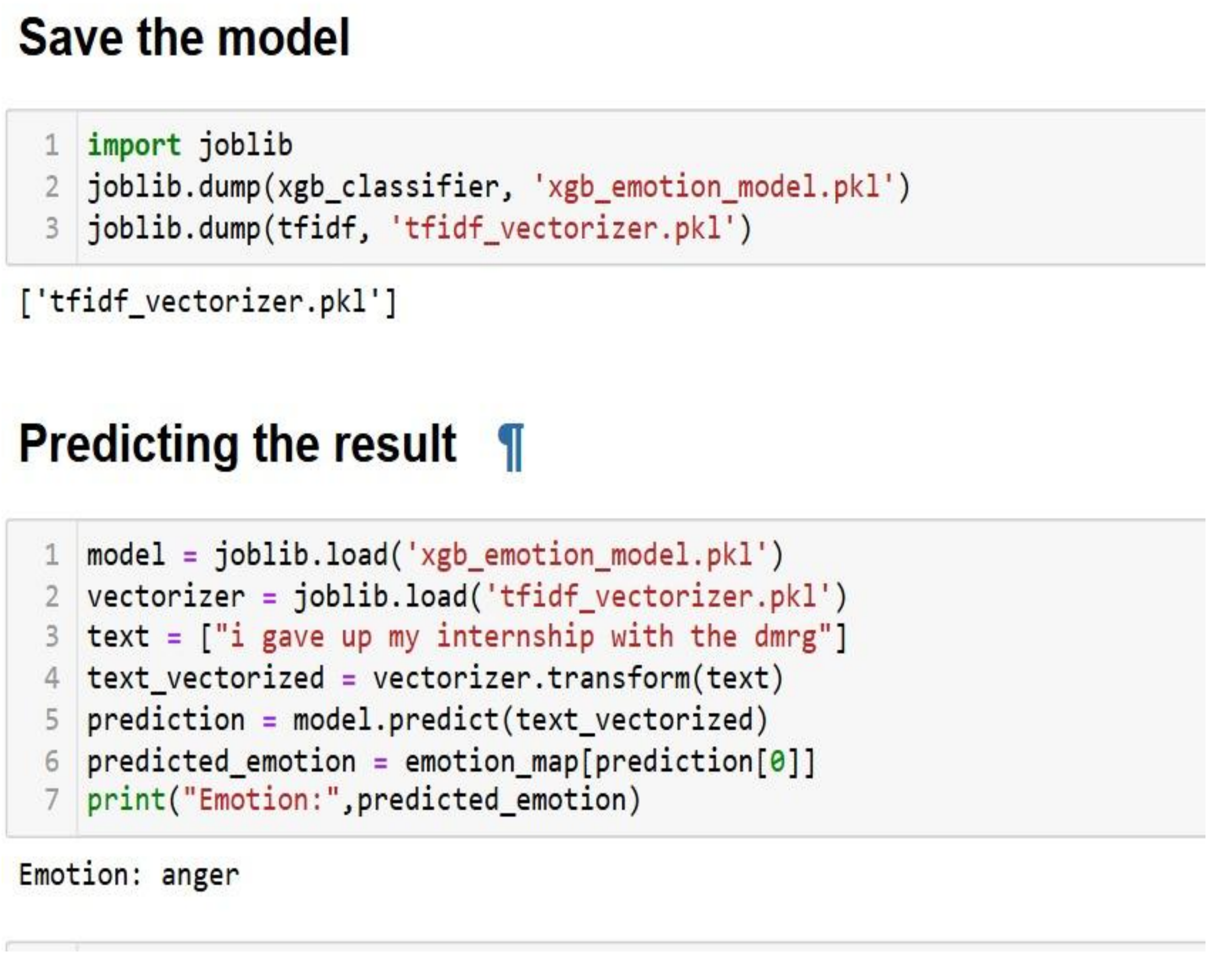


```
Decision tree classifier results:
Accuracy :0.809375
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86       946
           1       0.81      0.83      0.82      1021
           2       0.73      0.70      0.71       296
           3       0.78      0.78      0.78       427
           4       0.78      0.82      0.80       397
           5       0.74      0.60      0.66       113

    accuracy                           0.81      3200
   macro avg       0.78      0.76      0.77      3200
weighted avg       0.81      0.81      0.81      3200

Confusion Matrix :
[[814  49   7  44  29   3]
 [ 69 847  62  21  14   8]
 [  7  72 206   5   5   1]
 [ 35  35   6 331  19   1]
 [ 20  19   1  22 324  11]
 [  3  19   1   0  22  68]]
```

Comparing Models:

Model	**Accuracy**
Multinomial NB	0.7921875
XG Boost	0.861875
Decision Tree	0.809375

From the above table we can observe that **XG Boost** is the best model.Now we save the model and predict the emotion in text.



Save the model

```
import joblib
joblib.dump(xgb_classifier, 'xgb_emotion_model.pkl')
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
```

```
['tfidf_vectorizer.pkl']
```

Predicting the result ¶

```
model = joblib.load('xgb_emotion_model.pkl')
vectorizer = joblib.load('tfidf_vectorizer.pkl')
text = ["i gave up my internship with the dmrg"]
text_vectorized = vectorizer.transform(text)
prediction = model.predict(text_vectorized)
predicted_emotion = emotion_map[prediction[0]]
print("Emotion:",predicted_emotion)
```

```
Emotion: anger
```

GitHub: Emotion Detection in Text Project

Conclusion:

In this project, we successfully developed a text-based emotion detection model capable of classifying user input into emotional categories. By applying ML techniques, the system demonstrated high accuracy and reliable prediction results.The results validate the potential of automated emotion recognition in fields such as customer service ,mental health support and social media monitoring. However, recognizing emotions from text remains challenging due to the complexity and subjectivity of human emotions.

Result:

The developed system was tested using sample text inputs and successfully classified them into correct emotional categories. Step-by-step implementation of model included in this document. Which shows the model training process, prediction process, prediction samples and classification outputs. The system able to predict the emotions like joy, sadness, anger and fear, surprise from the given input text.