Mood Prediction Application Project Using NLP Classification

1. Introduction:

Mental health issues such as anxiety, stress, and depression have become increasingly prevalent, affecting millions of people globally. Recognizing these mental states early on can aid in timely intervention. This project aims to develop a Mood Prediction Application that leverages Natural Language Processing (NLP) to analyse user responses during a conversation and predict their emotional state, including categories like anxiety, stress, and depression.

This project is designed to predict mental health conditions based on textual data. The classification model is trained to identify three mental health categories: Normal, Depression, and Suicidal. The dataset includes mental health records annotated with various mental health conditions such as Anxiety, Bipolar Disorder, Stress, Personality Disorders, etc. However, for this project, only Normal, Depression, and Suicidal records were used for training and evaluation.

2.Objective:

The primary objective of the project is to:

- Provide an interactive chat-based application where users can engage in a conversation with an assistant.
- Leverage NLP techniques to process and clean user text responses.
- Use a pre-trained machine learning model to predict the mood of the user based on their inputs.
- Provide a result indicating the emotional state: Anxious, Depressed, Stressed, or Normal.

3. Overview of the application:

The Mood Prediction Chat Application is built using Streamlit, which enables users to engage in a real-time chat with a virtual assistant. During the conversation, users answer questions, and based on their input, the system

aggregates the responses and runs them through an NLP model to predict their mood.

Key Features of the Application:

- Interactive chat interface where the assistant asks a series of questions.
- The assistant's typing simulation adds a human touch to the interaction.
- User input is cleaned, pre-processed, and passed through a trained machine learning model to generate predictions.
- Final mood prediction is displayed to the user based on their responses.

4. About Dataset:

Dataset Source: The dataset is from kaggle.com. This comprehensive dataset is a meticulously curated collection of mental health statuses tagged from various statements. The dataset amalgamates raw data from multiple sources, cleaned and compiled to create a robust resource for developing chatbots and performing sentiment analysis.

Data Structure:

Training Set: Contains 7243 data records for training.

Testing Set: Used to evaluate the performance of the trained model. Has 3105 records for testing.

The dataset used in this project contains records associated with different mental health conditions. The records consist of text, which provides insight into a person's mental state, making it suitable for training machine learning models that can predict mental health conditions based on text.

Text Preprocessing: Before feeding the data into the model, several Natural Language Processing (NLP) techniques were applied:

- Stopwords Removal: Common English stopwords, which carry minimal semantic information, were removed using NLTK's stopwords.
- Lemmatization: Text was lemmatized using the WordNetLemmatizer, reducing words to their base form (e.g., "running" becomes "run").
- TF-IDF Vectorization: The cleaned text was transformed into a numerical representation using Term Frequency-Inverse Document Frequency (TF-IDF). This technique helps in assigning importance to words in the text while ignoring commonly occurring but less significant words.

- Data Balancing: An imbalanced dataset was handled using undersampling techniques, specifically the RandomUnderSampler.

5. Model Architecture and Training:

Multiple machine learning models were considered for classification, including:

- Naive Bayes: Suitable for text classification due to its simplicity and effectiveness with high-dimensional data.
- Decision Trees: Helps in modeling non-linear relationships within the data.
- Random Forest: An ensemble learning method for better prediction accuracy.
- AdaBoost and Gradient Boosting: Boosting techniques for improving the model's predictive power.
- XGBoost: A highly efficient gradient boosting algorithm.

Model Selection:

The models were trained using a variety of approaches, with cross-validation and hyperparameter tuning using RandomizedSearchCV to optimize the models' performance.

5. Evaluation:

After training, the models were evaluated using classification metrics such as accuracy, precision, recall, and F1-score to determine their performance on predicting the target classes. The final model, which performed the best, was selected for integration into the Streamlit app.

Accuracy = TP+TN / (TP +FP +TN +FN)

Precision = TP/TP+FP

Recall = TP/TP+FN

F1-Score = 2 x Precision x Recall/(Precision + Recall)

TP : Count of True Positive predictions

FP : Count of False Positive predictions

TN : Count of True Negative predictions

FN : Count of False Positive predictions

Result: The final accuracy of the solution model on the test data is approximately 85%.

6. Streamlit App:

Overview: A web-based application was built using Streamlit to allow users to upload images of apples and classify them as fresh or rotten.

Features:

Upload images via a file uploader.The app displays the image and provides a prediction of whether the apple is fresh or rotten.

7. Application

The trained model was integrated into a user-friendly web app using Streamlit. The app interacts with users by asking a series of pre-defined questions, allowing them to respond, and based on these responses, it predicts their mental state.

App Workflow

- Users answer a series of questions, which simulate a natural conversation.
- Their responses are processed using NLP techniques to clean and vectorize the text.
- The trained machine learning model predicts whether the user is experiencing a normal state, depression, or is suicidal.
- Results are displayed to the user in real-time, providing a seamless interaction experience.

8. Relevance and Use Cases

This application is particularly relevant in mental health care and well-being monitoring. With an increasing number of individuals experiencing emotional distress, early detection of mood disorders through non-invasive methods is crucial.

Use Cases:

- Initial Mental Health Assessment: Aiding users in understanding their mental state by reflecting on their emotions and thoughts.

- Personal Well-Being Companion: The app can serve as a companion, prompting users to reflect on their mental health regularly.
- Support in Counseling: Counselors or therapists can use such tools for pre-screening clients before sessions.
- Educational Tool: The app can be used to educate users about the importance of emotional awareness and mental health.

9. Code:

ABOUT DATASET AND PROJECT :

This dataset is ideal for training machine learning models aimed at understanding and predicting mental health conditions based on textual data. This dataset has records of Normal, Depression, Suicidal, Anxiety, Bipolar,Stress,Personality disorder mental categories. Using the records of Normal, Depression and Suicidal ,ML models for classification is build and used for prediction

1.Importing necessary modules for NLP

```
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from imblearn.under_sampling import RandomUnderSampler
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import RandomizedSearchCV
```

Loading dataset

```
df=pd.read_csv('/content/drive/MyDrive/Datasets-2024july/Projects/Combined Data.csv')
df.head()
```

	Unnamed: 0	statement	status
0	0	oh my gosh	Anxiety
1	1	trouble sleeping, confused mind, restless hear...	Anxiety
2	2	All wrong, back off dear, forward doubt. Stay ...	Anxiety
3	3	I've shifted my focus to something else but I'...	Anxiety
4	4	I'm restless and restless, it's been a month n...	Anxiety

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[4] df['status'].unique()
```

```
array(['Anxiety', 'Normal', 'Depression', 'Suicidal', 'Stress', 'Bipolar',
       'Personality disorder'], dtype=object)
```

Seperating required records for ML model

```
df1=df.loc[df['status']=='Normal']
df2=df.loc[df['status']=='Depression']
df3=df.loc[df['status']=='Anxiety']
df4=df.loc[df['status']=='Stress']
df=pd.concat([df1,df2,df3,df4],axis=0,ignore_index=True)
df
```

	Unnamed: 0	statement	status
0	733	Gr gr dreaming of ex crush to be my game, God	Normal
1	734	wkwkwk what a joke	Normal
2	735	Leaves are also standby in front of the PC	Normal
3	736	Thank God even though it's just a ride through	Normal
4	737	wedding teaser concept using the song day6 - o...	Normal
...	...	...	...
38307	49464	How to fight PCOS with diet and nutrition? PCO...	Stress
38308	49465	Do you ever feel like anxiety and depression a...	Stress
38309	49466	Is it normal to feel a gurgling in your chest ...	Stress

2.Data Preprocessing and EDA

checking for null records

```
df.isnull().sum()
```

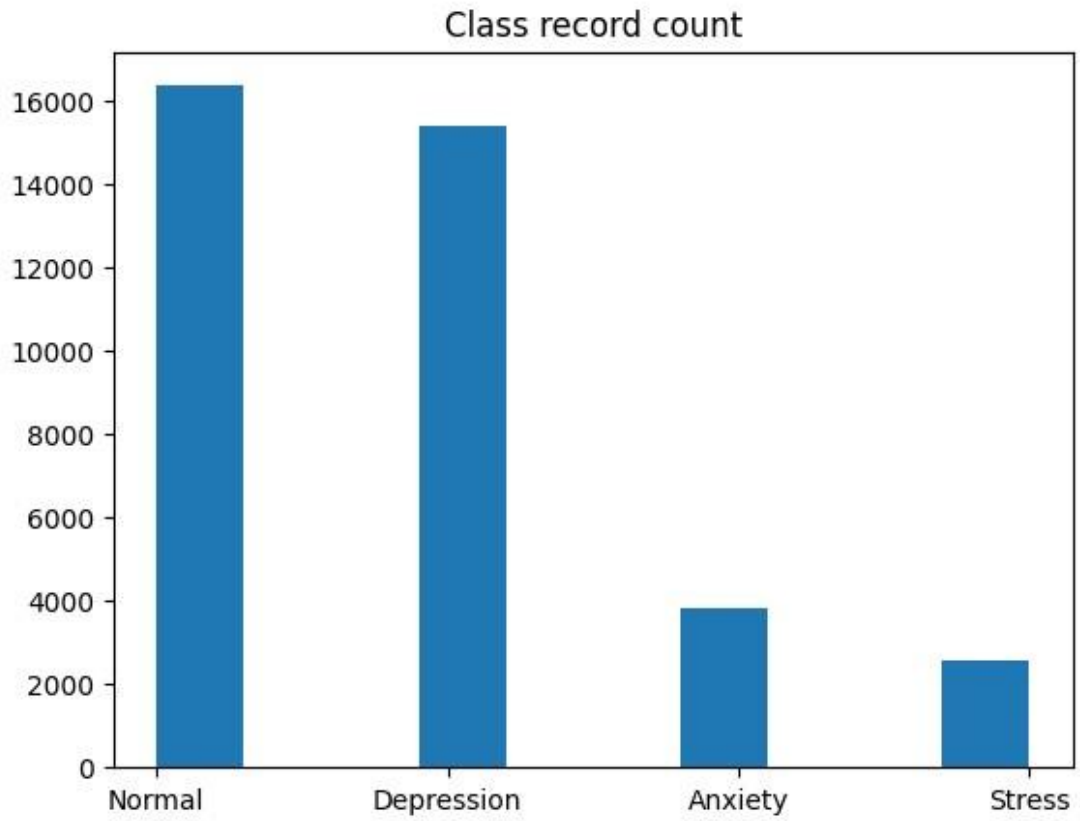
	0
Unnamed: 0	0
statement	137
status	0

dtype: int64

```
[7] df.dropna(inplace=True)
    df.reset_index(inplace=True,drop=True)
    df.isnull().sum()
```

	0
Unnamed: 0	0
statement	0

```
plt.title('Class record count')
plt.hist(df['status'])
plt.show()
```



Seperating text data and preprocessing it for model building

```
txt=df.iloc[:,1]
txt.head()
```

	statement
0	Gr gr dreaming of ex crush to be my game, God
1	wkwkwk what a joke
2	Leaves are also standby in front of the PC .....
3	Thank God even though it's just a ride through
4	wedding teaser concept using the song day6 - o...

dtype: object

Removing special characters

```
[11] txt=txt.str.replace('[^a-zA-Z0-9 ]','',regex=True)
     txt
```

Removing stopwords from text and converting text to lowercase

```
nltk.download('stopwords')
nltk.download('punkt')
sw=stopwords.words('english')
txt=txt.apply(lambda line:' '.join([word.lower() for word in nltk.word_tokenize(line) if word.lower() not in sw]))
txt.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

	statement
0	gr gr dreaming ex crush game god
1	wkwkwk joke
2	leaves also standby front pc office longer leave
3	thank god even though ride
4	wedding teaser concept using song day6 sounds ...

dtype: object

Converting each words to its root word using lemmatization

```
[13] nltk.download('wordnet')
     lemma=WordNetLemmatizer()
     txt=txt.apply(lambda line: ' '.join([lemma.lemmatize(word,pos='v') for word in nltk.word_tokenize(line)]))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

Initialising input and output variable

Converting text to vectors using TF-IDF Vectorizer

```
[14] vector=TfidfVectorizer()
     X=vector.fit_transform(txt)
     X
```

```
<38175x58022 sparse matrix of type '<class 'numpy.float64'>'
        with 1316347 stored elements in Compressed Sparse Row format>
```

```
[15] y=df['status'].map({'Anxiety':0,'Depression':1,'Stress':2,'Normal':3})
```

Handling class imabalance

```
[16] us=RandomUnderSampler()
     X,y=us.fit_resample(X,y)
```

```
y.value_counts()
```

status	count
0	2587
1	2587
2	2587
3	2587

dtype: int64

Splitting data for training and testing

```
x_train,x_test,y_train,y_test=train_test_split(X,y,random_state=1,test_size=0.3)
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

```
((7243, 58022), (7243,), (3105, 58022), (3105,))
```

```
----- MultinomialNB() -----
              precision    recall  f1-score   support

           0       0.71      0.85      0.77       786
           1       0.59      0.92      0.72       774
           2       0.68      0.67      0.67       796
           3       0.96      0.24      0.38       749

    accuracy                           0.67      3105
   macro avg       0.74      0.67      0.64      3105
weighted avg       0.73      0.67      0.64      3105
```

```
----- RandomForestClassifier() -----
              precision    recall  f1-score   support

           0       0.81      0.84      0.82       786
           1       0.82      0.79      0.81       774
           2       0.84      0.64      0.72       796
           3       0.76      0.94      0.84       749

    accuracy                           0.80      3105
   macro avg       0.80      0.80      0.80      3105
weighted avg       0.81      0.80      0.80      3105
```

```
----- DecisionTreeClassifier() -----
              precision    recall  f1-score   support

           0       0.72      0.69      0.71       786
           1       0.71      0.70      0.71       774
           2       0.63      0.63      0.63       796
           3       0.77      0.81      0.79       749

    accuracy                           0.71      3105
   macro avg       0.71      0.71      0.71      3105
weighted avg       0.71      0.71      0.71      3105
```

```
----- AdaBoostClassifier() -----
              precision    recall  f1-score   support

           0       0.76      0.75      0.76       786
           1       0.79      0.79      0.79       774
           2       0.70      0.57      0.63       796
           3       0.74      0.90      0.81       749

    accuracy                           0.75      3105
   macro avg       0.75      0.75      0.75      3105
weighted avg       0.75      0.75      0.75      3105
```

```
----- GradientBoostingClassifier() -----
              precision    recall  f1-score   support

           0       0.83      0.81      0.82       786
           1       0.85      0.80      0.83       774
           2       0.77      0.70      0.73       796
           3       0.77      0.92      0.84       749

    accuracy                           0.81      3105
   macro avg       0.81      0.81      0.81      3105
weighted avg       0.81      0.81      0.80      3105
```

```
----- XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...) -----
              precision    recall  f1-score   support

           0       0.85      0.84      0.84       786
           1       0.88      0.84      0.86       774
           2       0.79      0.79      0.79       796
           3       0.85      0.91      0.87       749

    accuracy                           0.84      3105
   macro avg       0.84      0.84      0.84      3105
weighted avg       0.84      0.84      0.84      3105
```

3. Building models

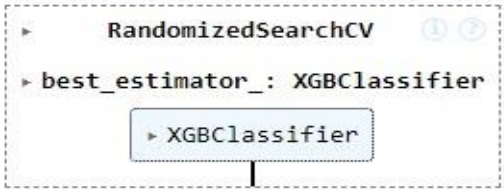
```
nb=MultinomialNB()
dtc=DecisionTreeClassifier()
rf=RandomForestClassifier()
gb=GradientBoostingClassifier()
ada=AdaBoostClassifier()
xgb=XGBClassifier()
```

```
models=[nb,dtc,rf,ada,gb,xgb]
for model in models:
  model.fit(x_train,y_train)
  y_pred=model.predict(x_test)
  print('-----',model,'-----')
  print(classification_report(y_test,y_pred))
```

4. Hyper Parameter Tuning

```
params = {
    'booster':['gbtree','gblinear'],
    'max_depth':[5,6,7],
    'n_estimators':[100,125,150]
}
```

```
clf=RandomizedSearchCV(xgb,params,cv=4,scoring='accuracy')
clf.fit(x_train,y_train)
```




```
pred=clf.predict(x_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.83      0.84       786
           1       0.87      0.87      0.87       774
           2       0.80      0.79      0.80       796
           3       0.88      0.90      0.89       749

    accuracy                           0.85      3105
   macro avg       0.85      0.85      0.85      3105
weighted avg       0.85      0.85      0.85      3105
```

```
import re
new_input='im scared.. i dont know what to do...i need help'
text=re.sub('[^a-zA-Z0-9 ]','',new_input)
txt = ' '.join([word.lower() for word in nltk.word_tokenize(text) if word.lower() not in sw])
txt = ' '.join([lemma.lemmatize(word, pos='v') for word in nltk.word_tokenize(txt)])
pred=clf.predict(vector.transform([txt]))
if pred:
    prediction=model.predict(vector.transform([txt]))
    if prediction==0:
        print('#Anxious')
    elif prediction==1:
        print('#Depressed')
    elif prediction==2:
        print('#Stressed')
    else:
        print('#Normal')
```

```
#Stressed
```

Saving model for GUI

```
import pickle
pickle.dump(clf,open('MHP.sav','wb'))
pickle.dump(vector,open('Vector.sav','wb'))
```

10. Future Enhancements:

- AI Chatbot Integration: An advanced enhancement would be to build a conversational AI chatbot that can engage users in a more dynamic and personalized way. By integrating a chatbot with natural language understanding (NLU) frameworks such as Rasa, Dialogflow, or GPT-based models, the interaction could go beyond pre-defined questions.
- Multilingual Support: Expanding the app to support multiple languages using NLP translation models would make it accessible to a broader audience. Fine-tuned multilingual models like mBERT or XLM-RoBERTa could help provide equally accurate predictions across various languages.
- Deployment as a Mobile Application: Deploying the app as a mobile-friendly tool, using React Native or Flutter, would make it more accessible to users. Mobile applications offer better usability and can also integrate features like push notifications to remind users to engage with the app for regular mental health check-ins.

11. Conclusion :

This project demonstrates the potential of NLP and machine learning in predicting mental health conditions through text-based conversations. By analyzing user inputs, the model is able to classify mental health states into categories such as Normal, Depression, and Suicidal, providing a valuable tool for early detection of mental health issues. The integration of this model into a user-friendly Streamlit application makes it accessible to non-technical users, facilitating interaction in a conversational manner.