EMOTION CLASSIFIER MODEL DOCUMENTATION

Aim: The aim of the project is to create a model which will understand the mode of speech either it is happy or sad or fear.

Tools used: Python.

Importing all the required libraries:

```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,MinMaxScaler,PowerTransformer,FunctionTransformer
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
import nltk
from nltk.tokenize import word_tokenize,sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,LancasterStemmer,SnowballStemmer,wordnet
import re
from sklearn.naive_bayes import BernoulliNB,MultinomialNB,CategoricalNB
from sklearn.metrics import accuracy_score
from nltk.stem import WordNetLemmatizer
import pickle
```

- Imported the required libraries to do all the steps in the project.
- Used the libraries sklearn, nltk, re, pickle.

Importing the dataset:

```
data = pd.read_csv(r"C:\Users\srava\Downloads\Emotion_classify_Data.csv")
```

```
data
```

	Comment	Emotion
0	i seriously hate one subject to death but now ...	fear
1	im so full of life i feel appalled	anger
2	i sit here to write i start to dig out my feel...	fear
3	ive been really angry with r and i feel like a...	joy
4	i feel suspicious if there is no one outside l...	fear
...	...	...
5932	i begun to feel distressed for you	fear
5933	i left feeling annoyed and angry thinking that...	anger
5934	i were to ever get married i d have everything...	joy
5935	i feel reluctant in applying there because i w...	fear
5936	i just wanted to apologize to you because i fe...	anger

5937 rows × 2 columns

- The data has 5937 data points and two columns.
- The columns are Comments and Emotion where the Comment column has text and the emotion column has the mode of text.

Dividing the Feature variable and class variable

```
fv = data.iloc[:,0]
cv = data.iloc[:,1]
```

```
# Splitting the data into train and test
```

```
x_train,x_test,y_train,y_test = train_test_split(fv,cv,test_size=0.2,random_state=42)
```

- Divided the data into train feature and class variable where the feature is comment column and class variable is emotion column.

- Splatted the data into train and test to perform training and testing.

Text preprocessing

```
# checking if there are any uppercase letters, html tags, urls and unwanted characters
```

```
def edat(data,name):
    case=" ".join(data[name]).islower()
    html_=data[name].apply(lambda x:True if re.search("<.+?>",x) else False).sum()
    url_=data[name].apply(lambda x:True if re.search("http[s]?://.+? +",x) else False).sum()
    unwanted_=data[name].apply(lambda x:True if re.search("[]()*\-.,@#$%^&0-9]",x) else False).sum()
    if case==False:
        print("not in lower case")
    if html_>0:
        print("have html tags")
    if url_>0:
        print("you are having urls")
    if unwanted_>0:
        print("you are having unwanted characters")
    else:
        print('The data is clean')
```

```
edat(data,'Comment')
```

```
The data is clean
```

- Checked whether the data has any lowercase letters, html tags, url's, unwanted characters.
- Our data is cleaned and don't have any unwanted characters.

Removing stop words

```
# removing stop words except 'not'
```

```
custom_stopwords = set(stopwords.words('english')) - {'not'}
```

```
def remove_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in custom_stopwords]
    return ' '.join(filtered_words)
```

```
x_train = x_train.apply(remove_stopwords)
```

- Removed stop words in the data.

Lemmatizing the Text

```
lemmatizer = WordNetLemmatizer()
```

```
def lemmatize_text(text):
    words = word_tokenize(text)
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(lemmatized_words)
```

```
x_train=x_train.apply(lemmatize_text)
```

- WordNet Lemmatizer is used to lemmatize each word in the sentence. The 'lemmatize' method takes into account the part of speech (POS) of each word, so it's often used in combination with POS tagging for better lemmatization results.
- Applied the lemmatization on x_train.

Converting text to Binary bag of words

```
ct = CountVectorizer(binary=True)
```

```
bbow = ct.fit_transform(x_train)
```

```
bbow
```

```
<4749x7039 sparse matrix of type '<class 'numpy.int64'>'
        with 43482 stored elements in Compressed Sparse Row format>
```

- Converting text to a binary bag of words representation is a way to represent text data in a binary format indicating the presence or absence of words. In the binary bag of words model, each document is represented by a binary vector where each element corresponds to the presence (1) or absence (0) of a particular word from the vocabulary.
- Converted the x_train to binary bag of words.

Preparing model

```
br = BernoulliNB()
```

```
final_pred = br.fit(bbow,y_train)
```

```
final_pred
```

```
▾ BernoulliNB
BernoulliNB()
```

```
cv2 = ct.transform(x_test)
```

```
y_pred = br.predict(cv2)
```

```
accuracy_score(y_test,y_pred)
```

```
0.9057239057239057
```

- Used Bernoulli naive bayes for the model.
- Performed the steps as shown in the image and got accuracy of 90 percent.

Deployement

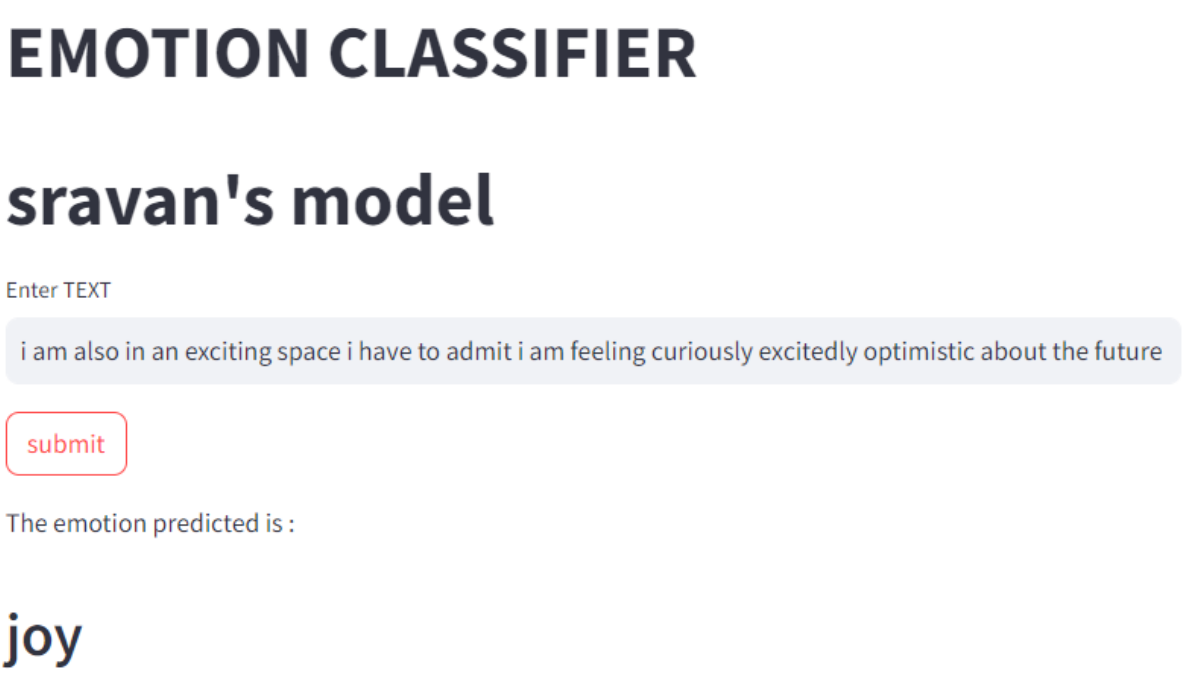
```
final_model = pickle.dump(final_pred,open(r'C:\Users\srava\Downloads\Emotion_model.pkl','wb'))
```

```
model = pickle.load(open(r"C:\Users\srava\Downloads\Emotion_model.pkl",'rb'))
```

```
model.predict(ct.transform(x_test.iloc[[76]]))
```

```
array(['joy'], dtype='<U5')
```

- Deployed the model and loaded the model using pickle.
- Model is ready to use.
- Used visual studio for web application visualization.



- The above shown image is the web application output of the model I have created using visual studio.