Nama : Rifqi Alfinnur C

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import wordcloud as wcd
pd.set_option('mode.chained_assignment', None)
palette=sns.color_palette("husqma")
sns.set(palette=palette)
```

```
#Reading the Data
train_df=pd.read_csv('Corona_NLP_train.csv', encoding='latin-1')
test_df=pd.read_csv('Corona_NLP_test.csv', encoding='latin-1')
```

```
train_df.head()
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	1799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV...	Extremely Negative

Exploring The Data

```
test_data.head()
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
3	4	44956	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative
4	5	44957	Melbourne, Victoria	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	Neutral

```
train_data.shape
```

```
(41157, 6)
```

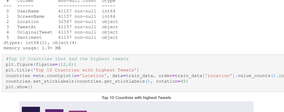
```
test_data.shape
```

```
(3798, 6)
```

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41157 entries, 0 to 41156
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   UserName       41157 non-null  int64
 1   ScreenName     41157 non-null  int64
 2   Location       32567 non-null  object
 3   TweetAt        41157 non-null  object
 4   OriginalTweet  41157 non-null  object
 5   Sentiment      41157 non-null  object
dtypes: int64(2), object(4)
memory usage: 1.9 MB
```

```
#Top 10 Countries that had the highest tweets
plt.figure(figsize=(12,6))
plt.title('Top 10 Countries with highest Tweets')
countries=mask.value_counts[lim:'Location', dataset=train_data, order=train_data['Location'].value_counts().index[:10],
countries.set_xticklabels(countries.get_xticklabels(), rotation=45)
plt.show()
```




```
#Pie Chart for the Sentiments Percentage
plt.figure(figsize=(15,6))
plt.pie(train_data['Sentiment'].value_counts(), labels=train_data['Sentiment'].unique(), autopct='%.1f%%', text
plt.legend(loc='upper right')
plt.axis('equal')
plt.show()
```

```
#Wordcloud for the Sentiments
from wordcloud import WordCloud
for label, cmap in zip(['Positive', 'Negative', 'Neutral', 'Extremely Positive', 'Extremely Negative'],
                       ['winter', 'Reds', 'Greys', 'summer', 'Greens', 'viridis', 'plasma']):
    text = train_data.query('Sentiment == {}labels[i]{}')['OriginalTweet'].str.cat(sep=' ')
    wc = WordCloud(width=500,
    wc = WordCloud(width=500, height=400, background_color='#FEFEFE', colormap=cmap)
    wc.generate_from_text(text)
    plt.imshow(wc)
    plt.axis("off"
    plt.title("Words Commonly Used in {}labels[i] Messages", size=20)
    plt.show()
```

Words Commonly Used in *Positive* Messages

Words Commonly Used in *Negative* Messages

Words Commonly Used in *Neutral* Messages

Words Commonly Used in *ExtremelyPositive* Messages

Words Commonly Used in *ExtremelyNegative* Messages

```
#Only interested in the OriginalTweet and Sentiment Columns
train_df= train_data[['OriginalTweet','Sentiment']]
test_df= test_data[['OriginalTweet','Sentiment']]
```

```
train_df.head()
```

	OriginalTweet	Sentiment
0	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/...	Neutral
1	advice Talk to your neighbours family to excha...	Positive
2	Coronavirus Australia: Woolworths to give elde...	Positive
3	My food stock is not the only one which is emp...	Positive
4	Me, ready to go at supermarket during the #COV...	Extremely Negative

Preprocessing

```
#Checking for null values
train_df.isnull().sum()
```

```
OriginalTweet    0
Sentiment        0
dtype: int64
```

```
test_df.isnull().sum()
```

```
OriginalTweet    0
Sentiment        0
dtype: int64
```

```
#Mapping the target column to (Negative/Neutral/Positive)
target_mapping={'Extremely Negative':5, 'Negative':4, 'Neutral':1,
                'Positive':2, 'Extremely Positive':3}
train_df['SentimentMapped']=train_df['Sentiment'].map(lambda x:target_mapping[x])
test_df['SentimentMapped']=test_df['Sentiment'].map(lambda x:target_mapping[x])
```

```
train_df.head()
```

	OriginalTweet	Sentiment	SentimentMapped
0	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/...	Neutral	1
1	advice Talk to your neighbours family to excha...	Positive	2
2	Coronavirus Australia: Woolworths to give elde...	Positive	2
3	My food stock is not the only one which is emp...	Positive	2
4	Me, ready to go at supermarket during the #COV...	Extremely Negative	0

```
#Cleaning the text
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
all_stopwords = stopwords.words('english')
all_stopwords.remove('not')

def preprocessing(text):
    corpus = []
    for i in range(len(data)):
        #remove urls
        text = re.sub(r'http[s?]', ' ', data['OriginalTweet'][i])
        #remove html tags
        text2 = re.sub(r'<.*?>', ' ', text1)
        #remove digits
        text3 = re.sub(r'\d+', ' ', text2)
        #remove punctuation
        text4 = re.sub(r'[^\w+]', ' ', text3)
        review = re.sub(r'[^a-zA-Z]', ' ', text4)
        review = review.lower()
        review = review.split()
        review = ' '.join(review)
        review = [ps.stem(word) for word in review if word not in all_stopwords]
        review = ' '.join(review)
        corpus.append(review)
    return corpus
```

```
#Transforming the OriginalTweet
text_df['OriginalTweet'].shape, (3798,)
corpus_testpreprocessed=text_df[
```

```
#Displaying the first 5 rows of corpus_train
corpus_train[0:5]
```

```
['morphin phil gahan chrisitv',
 'advic talk neighbour famili exchang phone number creat contact list phone number neighbour school employ chen
 len gp net milk nbc news nbcnews supermarkt replied meet not order',
 'coronavirus australia woolworth give elderli disabl dedicat shop hour amid covid outbreak',
 'my food stock not only one which empty pleas don panic there food enough for everyone take care each other',
 'ready go supermarket outbreak not paranoid food stock litterali empti melon thing pleas panic ceas shortag']
```

Bag Words of Models

```
#Splitting the Bag of words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2000) #for all pr 21750 columns if you dont specify max_features
X_train = cv.fit_transform(corpus_train).toarray()
y_train = train_df['SentimentMapped']
X_test = cv.transform(corpus_test).toarray()
y_test = test_df['SentimentMapped']
```

```
X_train.shape
```

```
(41157, 10000)
```

```
X_test.shape
```

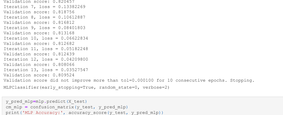
```
(3798, 10000)
```

Finally Model

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.neural_network import MLPClassifier
mlpCLASSIfier=MLPClassifier(random_state=1, early_stopping=True, verbose=2)
mlp.fit(X_train, y_train)
```

```
Iteration 1, loss = 0.72455349
Validation score: 0.808066
Iteration 2, loss = 0.43149657
Validation score: 0.827184
Iteration 3, loss = 0.28968849
Validation score: 0.821186
Iteration 4, loss = 0.20226486
Validation score: 0.822166
Iteration 5, loss = 0.14607548
Validation score: 0.820941
Iteration 6, loss = 0.10799368
Validation score: 0.817476
Iteration 7, loss = 0.13362284
Validation score: 0.818756
Iteration 8, loss = 0.13612887
Validation score: 0.814812
Iteration 9, loss = 0.09993183
Validation score: 0.813148
Iteration 10, loss = 0.06622834
Validation score: 0.812429
Iteration 11, loss = 0.05182248
Validation score: 0.812439
Iteration 12, loss = 0.04259960
Validation score: 0.808046
Iteration 13, loss = 0.03918547
Validation score: 0.809524
Validation score did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
```

```
MLPClassifier(early_stopping=True, random_state=1, verbose=2)
```

```
y_pred_mlp=mlp.predict(X_test)
cm_mlp = confusion_matrix(y_test, y_pred_mlp)
print(classification_report(y_test, y_pred_mlp))
print(classification_report(y_test, y_pred_mlp))
sns.heatmap(cm_mlp, annot=True, fmt='g', cbar=False)
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.title('MLP Confusion Matrix')
plt.show()
```

```
MLP Accuracy: 0.81161240381329
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1633
           1       0.72      0.74      0.74       619
           2       0.81      0.85      0.83      1546

    accuracy                           0.81      3798
   macro avg       0.79      0.80      0.80      3798
weighted avg       0.81      0.81      0.81      3798
```