```
print("hello")
```

```
hello
```

```
# utilities
import re
import numpy as np
import pandas as pd
# plotting
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# nltk
from nltk.stem import WordNetLemmatizer
# sklearn
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report
```

```
DATASET_COLUMNS=['Target','Id','Date','Flag','User','Text']
DATASET_ENCODING = "ISO-8859-1"
df = pd.read_csv('twitter_new.csv', encoding=DATASET_ENCODING,
names=DATASET_COLUMNS)
df.sample(5)
```

```
         Target          Id                          Date      Flag  \
1566786       4  2187858770  Mon Jun 15 20:54:58 PDT 2009  NO_QUERY
2578          0  1468405683  Tue Apr 07 01:28:12 PDT 2009  NO_QUERY
112553        0  1825485684  Sun May 17 05:52:12 PDT 2009  NO_QUERY
207155        0  1973411047  Sat May 30 11:05:43 PDT 2009  NO_QUERY
713374        0  2258820631  Sat Jun 20 16:33:05 PDT 2009  NO_QUERY

                    User
Text
1566786       NeeliaROX  Just uploaded PICS in facebook, friendster,
my...
2578          Bartemans  @robertzalme Yes I do... Too much theory
getti...
112553          Jonloge  My mom DOES NOT know how to walk in the city!
...
207155      PaoloAlonso  @taylorswift13 I just read your blog, it was
h...
713374   ShaunaMilliken  I'm the worst mother in the whole world. My
ba...
```

```
df.shape
```

```
(1600000, 6)
```

```
df.head()
```

```
   Target          Id                          Date      Flag  \
0       0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1       0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
2       0  1467810917  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY
3       0  1467811184  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
4       0  1467811193  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY

              User                                                Text

0  _TheSpecialOne_  @switchfoot http://twitpic.com/2y1zl - Awww, t...

1    scotthamilton  is upset that he can't update his Facebook by ...

2         mattycus  @Kenichan I dived many times for the ball. Man...

3          ElleCTF    my whole body feels itchy and like its on fire

4           Karoli  @nationwideclass no, it's not behaving at all....
```

```
df.columns
```

```
Index(['Target', 'Id', 'Date', 'Flag', 'User', 'Text'],
dtype='object')
```

```
print(len(df))
```

```
1600000
```

```
df.info
```

```
<bound method DataFrame.info of          Target          Id
Date      Flag  \
0             0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1             0  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
2             0  1467810917  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY
3             0  1467811184  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
4             0  1467811193  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
...         ...         ...                           ...       ...
1599995       4  2193601966  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599996       4  2193601969  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599997       4  2193601991  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599998       4  2193602064  Tue Jun 16 08:40:49 PDT 2009  NO_QUERY
1599999       4  2193602129  Tue Jun 16 08:40:50 PDT 2009  NO_QUERY

                    User
Text
0        _TheSpecialOne_  @switchfoot http://twitpic.com/2y1zl - Awww,
t...
1          scotthamilton  is upset that he can't update his Facebook
```

```
by ...
2               mattycus  @Kenichan I dived many times for the ball.
Man...
3                ElleCTF    my whole body feels itchy and like its on
fire
4                 Karoli  @nationwideclass no, it's not behaving at
all....
...                  ...
...
1599995  AmandaMarie1028  Just woke up. Having no school is the best
fee...
1599996      TheWDBoards  TheWDB.com - Very cool to hear old Walt
interv...
1599997           bpbabe  Are you ready for your MoJo Makeover? Ask me
f...
1599998     tinydiamondz  Happy 38th Birthday to my boo of alll
time!!! ...
1599999   RyanTrevMorris  happy #charitytuesday @theNSPCC
@SparksCharity...

[1600000 rows x 6 columns]>
```

```
np.sum(df.isnull().any(axis=1))
```

```
0
```

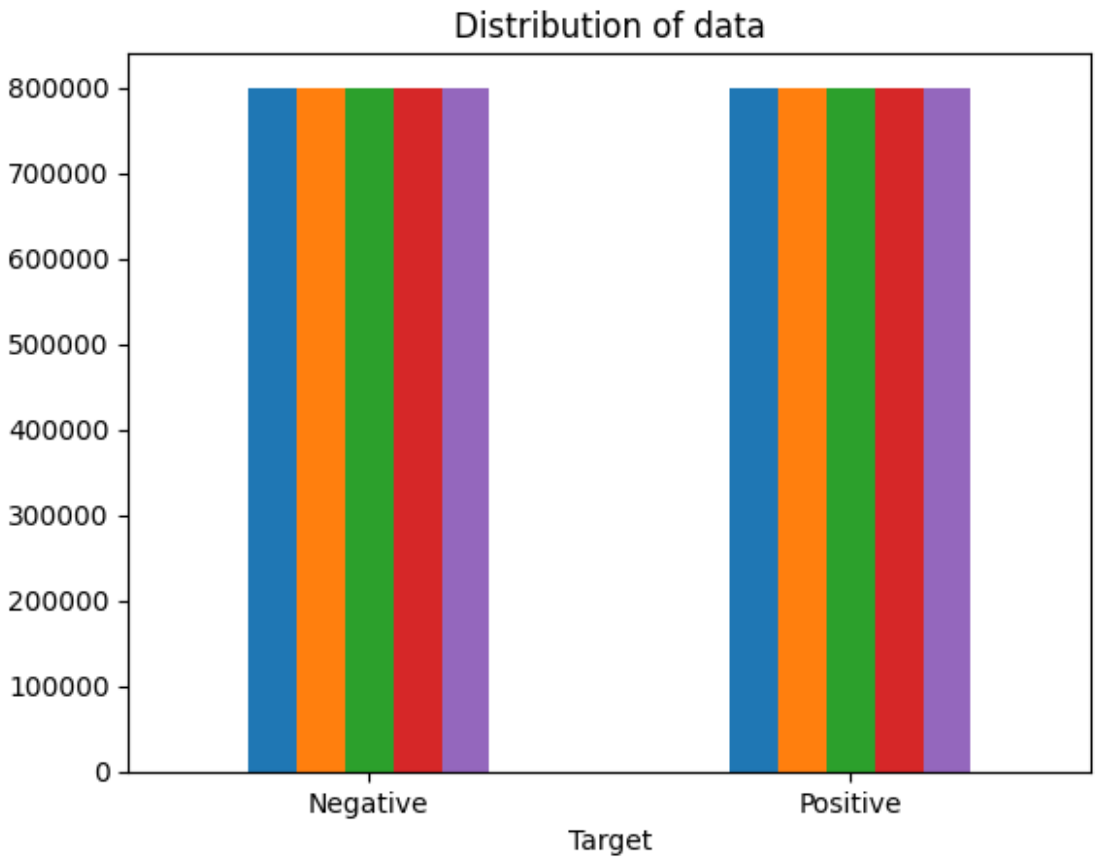
```
df['Target'].unique()
```

```
array([0, 4], dtype=int64)
```

```
df['Target'].nunique()
```

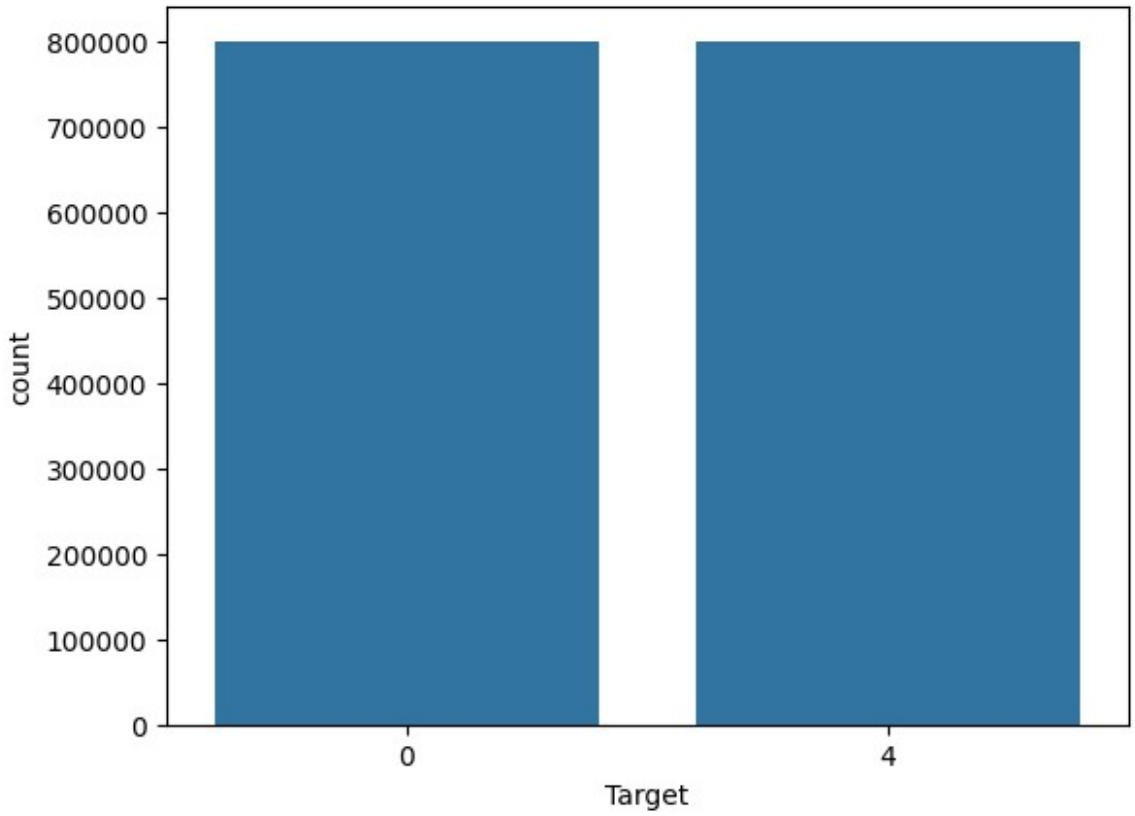
```
2
```

```
# Plotting the distribution for dataset.
ax = df.groupby('Target').count().plot(kind='bar', title='Distribution
of data',legend=False)
ax.set_xticklabels(['Negative','Positive'], rotation=0)
# Storing data in lists.
text, sentiment = list(df['Text']), list(df['Target'])
```




```
import seaborn as sns
sns.countplot(x='Target', data=df)
```

```
<AxesSubplot: xlabel='Target', ylabel='count'>
```




```
##Data Preprocessing

data=df[['Text','Target']]

data['Target'] = data['Target'].replace(4,1)

C:\Users\sangk\AppData\Local\Temp\ipykernel_11396\778994191.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  data['Target'] = data['Target'].replace(4,1)

data['Target'].unique()

array([0, 1], dtype=int64)

data_pos = data[data['Target'] == 1]
data_neg = data[data['Target'] == 0]

data_pos = data_pos.iloc[:int(20000)]
data_neg = data_neg.iloc[:int(20000)]
```

```
dataset = pd.concat([data_pos, data_neg])

dataset['Text']=dataset['Text'].str.lower()
dataset['Text'].tail()
```

```
19995    not much time off this weekend, work trip to m...
19996                              one more day of holidays
19997    feeling so down right now .. i hate you damn h...
19998    geez,i hv to read the whole book of personalit...
19999    i threw my sign at donnie and he bent over to ...
Name: Text, dtype: object
```

```
stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all',
'am', 'an',
             'and','any','are', 'as', 'at', 'be', 'because', 'been',
'before',
             'being', 'below', 'between','both', 'by', 'can', 'd',
'did', 'do',
             'does', 'doing', 'down', 'during', 'each','few', 'for',
'from',
             'further', 'had', 'has', 'have', 'having', 'he', 'her',
'here',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'i',
'if', 'in',
             'into','is', 'it', 'its', 'itself', 'just', 'll', 'm',
'ma',
             'me', 'more', 'most','my', 'myself', 'now', 'o', 'of',
'on', 'once',
             'only', 'or', 'other', 'our', 'ours','ourselves', 'out',
'own', 're','s', 'same', 'she', "shes", 'should', "shouldve",'so',
'some', 'such',
             't', 'than', 'that', "thatll", 'the', 'their', 'theirs',
'them',
             'themselves', 'then', 'there', 'these', 'they', 'this',
'those',
             'through', 'to', 'too','under', 'until', 'up', 've',
'very', 'was',
             'we', 'were', 'what', 'when', 'where','which','while',
'who', 'whom',
             'why', 'will', 'with', 'won', 'y', 'you', "youd","youll",
"youre",
             "youve", 'your', 'yours', 'yourself', 'yourselves']

# Cleaning and removing the above stop words list from the tweet text
STOPWORDS = set(stopwordlist)
def cleaning_stopwords(Text):
    return " ".join([word for word in str(Text).split() if word not in
STOPWORDS])
dataset['Text'] = dataset['Text'].apply(lambda text:
```

```
cleaning_stopwords(text))
dataset['Text'].head()
```

```
800000                love @health4uandpets u guys r best!!
800001    im meeting one besties tonight! cant wait!! - ...
800002    @darealsunisakim thanks twitter add, sunisa! g...
800003    sick really cheap hurts much eat real food plu...
800004                     @lovesbrooklyn2 effect everyone
Name: Text, dtype: object
```

```
# Cleaning and removing punctuations
import string
english_punctuations = string.punctuation
punctuations_list = english_punctuations
def cleaning_punctuations(text):
    translator = str.maketrans('', '', punctuations_list)
    return text.translate(translator)
dataset['Text']= dataset['Text'].apply(lambda x:
cleaning_punctuations(x))
dataset['Text'].tail()
```

```
19995    not much time off weekend work trip malmï¿½ fr...
19996                                   one day holidays
19997                    feeling right  hate damn humprey
19998    geezi hv read whole book personality types emb...
19999    threw sign donnie bent over get but thingee ma...
Name: Text, dtype: object
```

```
# Cleaning and removing repeating characters
def cleaning_repeating_char(text):
    return re.sub(r'(.)1+', r'1', text)
dataset['Text'] = dataset['Text'].apply(lambda x:
cleaning_repeating_char(x))
dataset['Text'].tail()
```

```
19995    not much time off weekend work trip malmï¿½ fr...
19996                                   one day holidays
19997                    feeling right  hate damn humprey
19998    geezi hv read whole book personality types emb...
19999    threw sign donnie bent over get but thingee ma...
Name: Text, dtype: object
```

```
# Cleaning and removing URLs

def cleaning_URLs(data):
    return re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data)
dataset['Text'] = dataset['Text'].apply(lambda x: cleaning_URLs(x))
dataset['Text'].tail()
```

```
19995    not much time off weekend work trip malmï¿½ fr...
19996                                   one day holidays
```

```
19997                      feeling right  hate damn humprey
19998    geezi hv read whole book personality types emb...
19999    threw sign donnie bent over get but thingee ma...
Name: Text, dtype: object
```

```
# Cleaning and removing numeric numbers

def cleaning_numbers(data):
    return re.sub('[0-9]+', '', data)
dataset['Text'] = dataset['Text'].apply(lambda x: cleaning_numbers(x))
dataset['Text'].tail()
```

```
19995    not much time off weekend work trip malmï¿½ fr...
19996                                      one day holidays
19997                      feeling right  hate damn humprey
19998    geezi hv read whole book personality types emb...
19999    threw sign donnie bent over get but thingee ma...
Name: Text, dtype: object
```

```
# Getting tokenization of tweet text

from nltk.tokenize import RegexpTokenizer
# Convert 'Text' column to strings
dataset['Text'] = dataset['Text'].astype(str)
tokenizer = RegexpTokenizer(r'\w+')
dataset['Text'] = dataset['Text'].apply(tokenizer.tokenize)
print(dataset['Text'].head())
```

```
800000           []
800001          [w]
800002    [w, w, w]
800003           []
800004           []
Name: Text, dtype: object
```

```
# Applying stemming

import nltk
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data
dataset['Text']= dataset['Text'].apply(lambda x: stemming_on_text(x))
dataset['Text'].head()
```

```
800000           []
800001          [w]
800002    [w, w, w]
800003           []
800004           []
Name: Text, dtype: object
```

```
import nltk
nltk.download('wordnet')
```

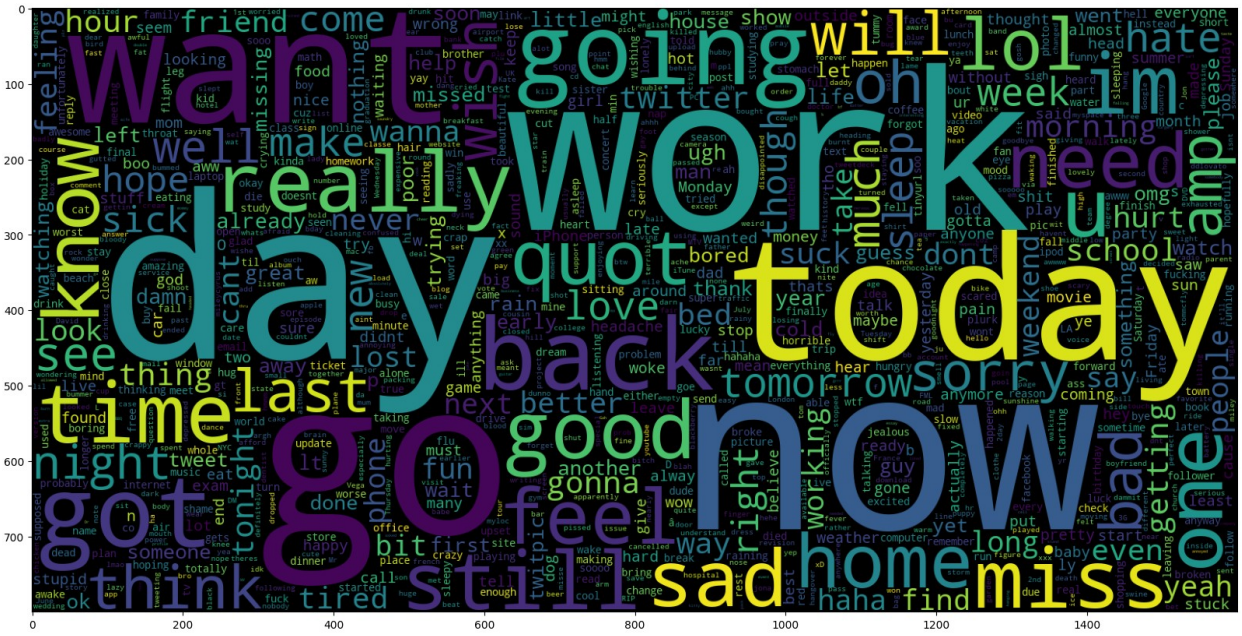
```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\sangk\AppData\Roaming\nltk_data...

True
```

```
# Applying lemmatizer

lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data
dataset['Text'] = dataset['Text'].apply(lambda x:
lemmatizer_on_text(x))
dataset['Text'].head()
```

```
800000           []
800001          [w]
800002    [w, w, w]
800003           []
800004           []
Name: Text, dtype: object
```

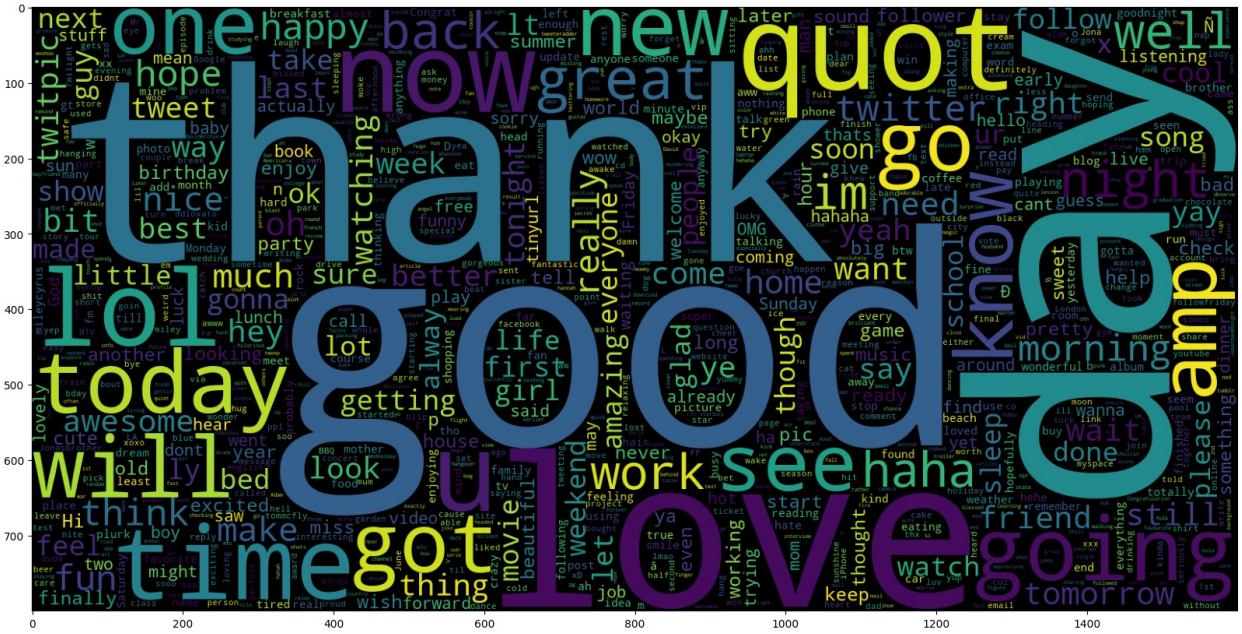
```
# Separating input feature and label

X=data.Text
y=data.Target

# Plot a cloud of words for negative tweets

data_neg = data['Text'][:800000]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_neg))
plt.imshow(wc)
```

```
<matplotlib.image.AxesImage at 0x219882f60e0>
```



```
# Plot a cloud of words for positive tweets

data_pos = data['Text'][800000:]
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_pos))
plt.figure(figsize = (20,20))
plt.imshow(wc)
```

```
<matplotlib.image.AxesImage at 0x219891b6e90>
```



```
# Splitting Our Data Into Train and Test Subsets

# Separating the 95% data for training data and 5% for testing data
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size =
0.05, random_state =26105111)

# Transforming the Dataset Using TF-IDF Vectorizer

vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
vectoriser.fit(X_train)
print('No. of feature_words: ', len(vectoriser.get_feature_names()))
```

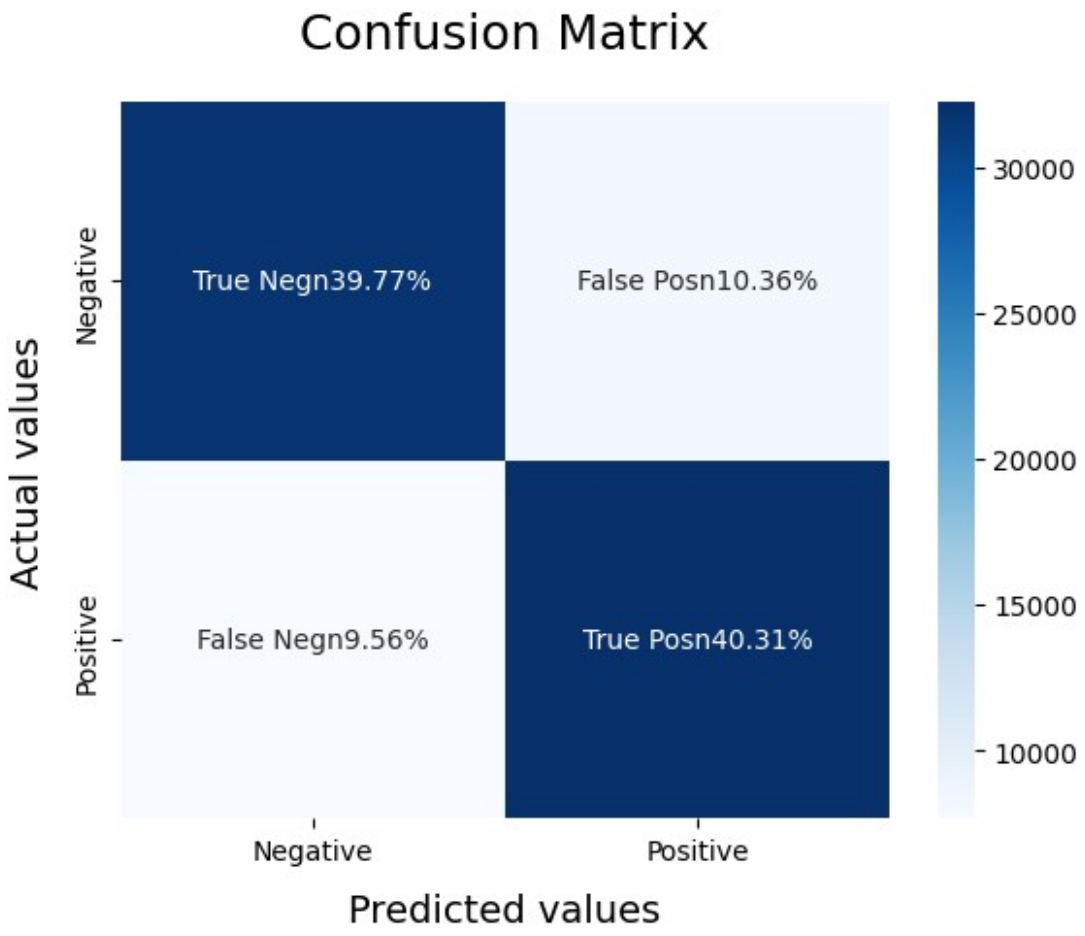
```
c:\Users\sangk\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\utils\deprecation.py:87: FutureWarning: Function
get_feature_names is deprecated; get_feature_names is deprecated in
1.0 and will be removed in 1.2. Please use get_feature_names_out
instead.
  warnings.warn(msg, category=FutureWarning)
```

```
No. of feature_words:  500000
```

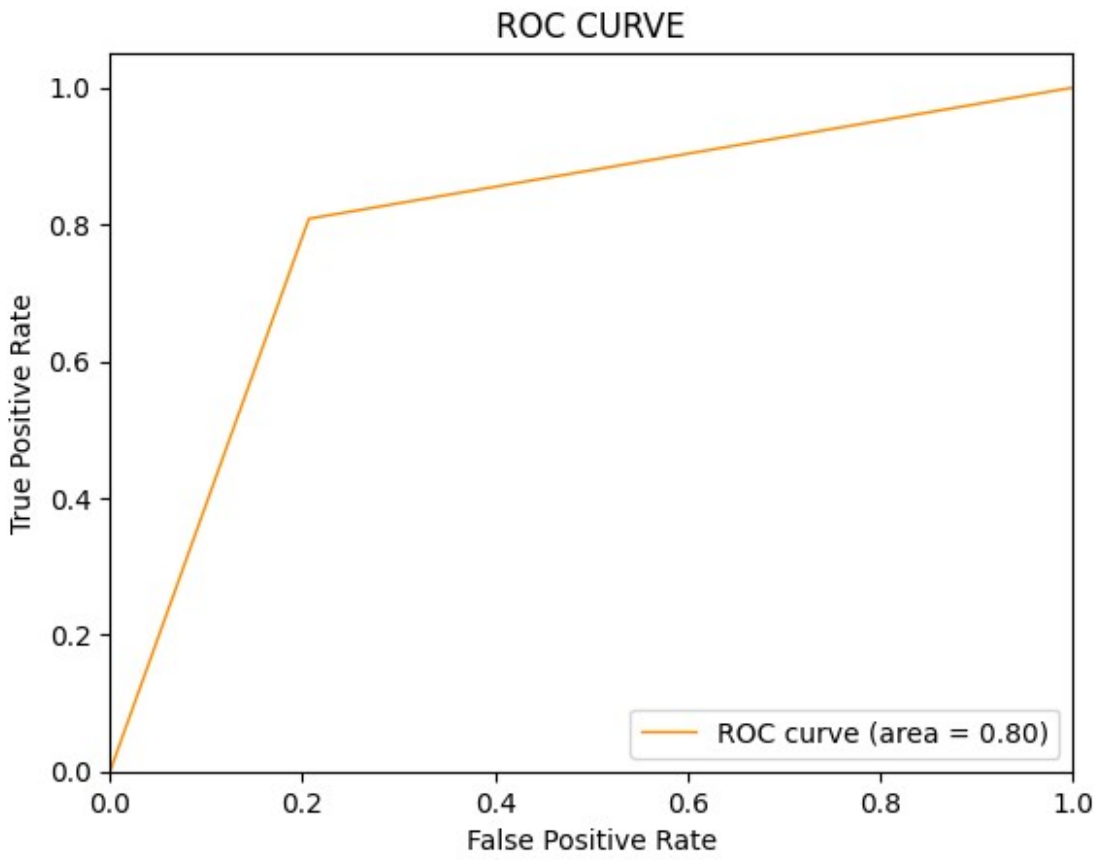
```
# Transform the data using TF-IDF Vectorizer

X_train = vectoriser.transform(X_train)
X_test  = vectoriser.transform(X_test)

# Function for Model Evaluation

# Accuracy Score
# Confusion Matrix with Plot
# ROC-AUC Curve

def model_Evaluate(model):
    # Predict values for Test dataset
    y_pred = model.predict(X_test)
    # Print the evaluation metrics for the dataset.
    print(classification_report(y_test, y_pred))
    # Compute and plot the Confusion matrix
    cf_matrix = confusion_matrix(y_test, y_pred)
    categories = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in
cf_matrix.flatten() / np.sum(cf_matrix)]
    labels = [f'{v1}n{v2}' for v1, v2 in
zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
    xticklabels = categories, yticklabels = categories)
    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad =
10)
    plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad =
```

```
10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)

# Model Building

# Model-1:
# 1.Bernoulli Naive Bayes Classifier
# 2.SVM (Support Vector Machine)
# 3.Logistic Regression

BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel)
y_pred1 = BNBmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

           0       0.81      0.79      0.80     40100
           1       0.80      0.81      0.80     39900

    accuracy                           0.80     80000
   macro avg       0.80      0.80      0.80     80000
weighted avg       0.80      0.80      0.80     80000
```

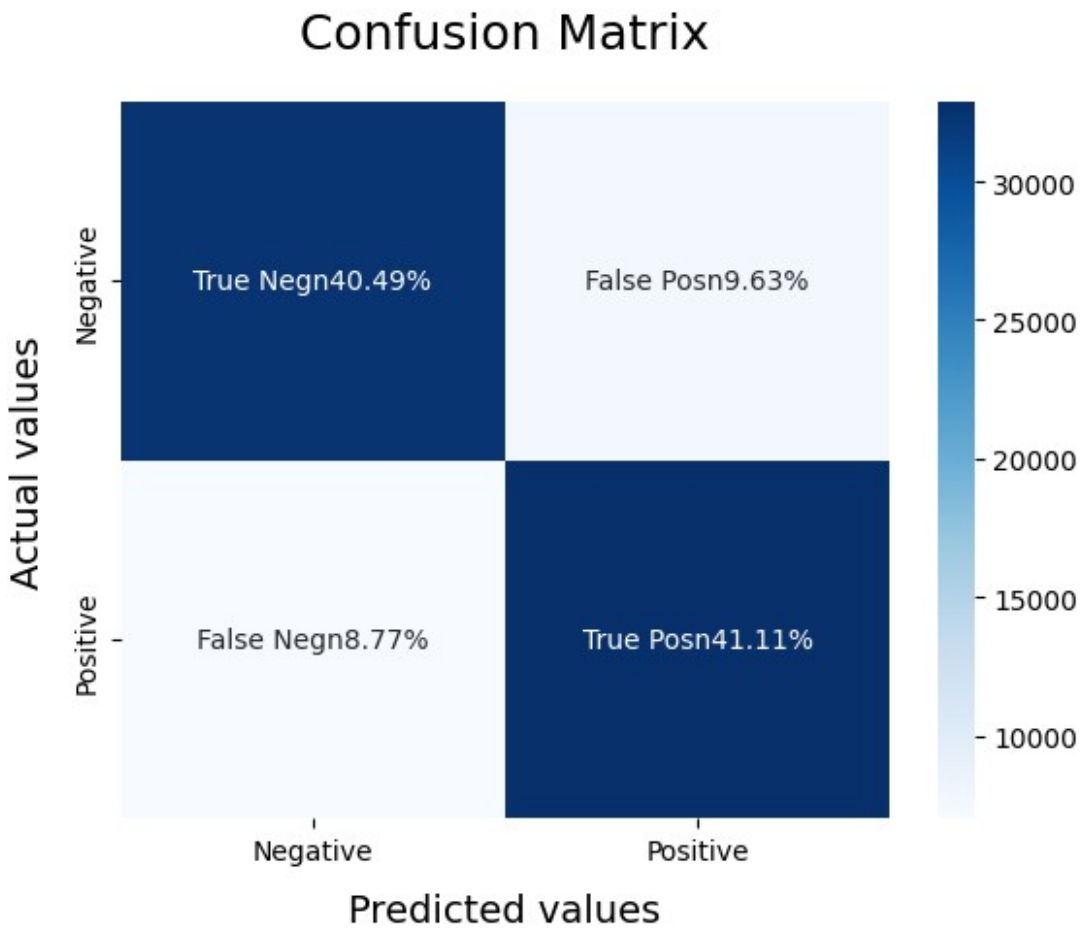



```
#Plot the ROC-AUC Curve for model-1
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area =
%0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

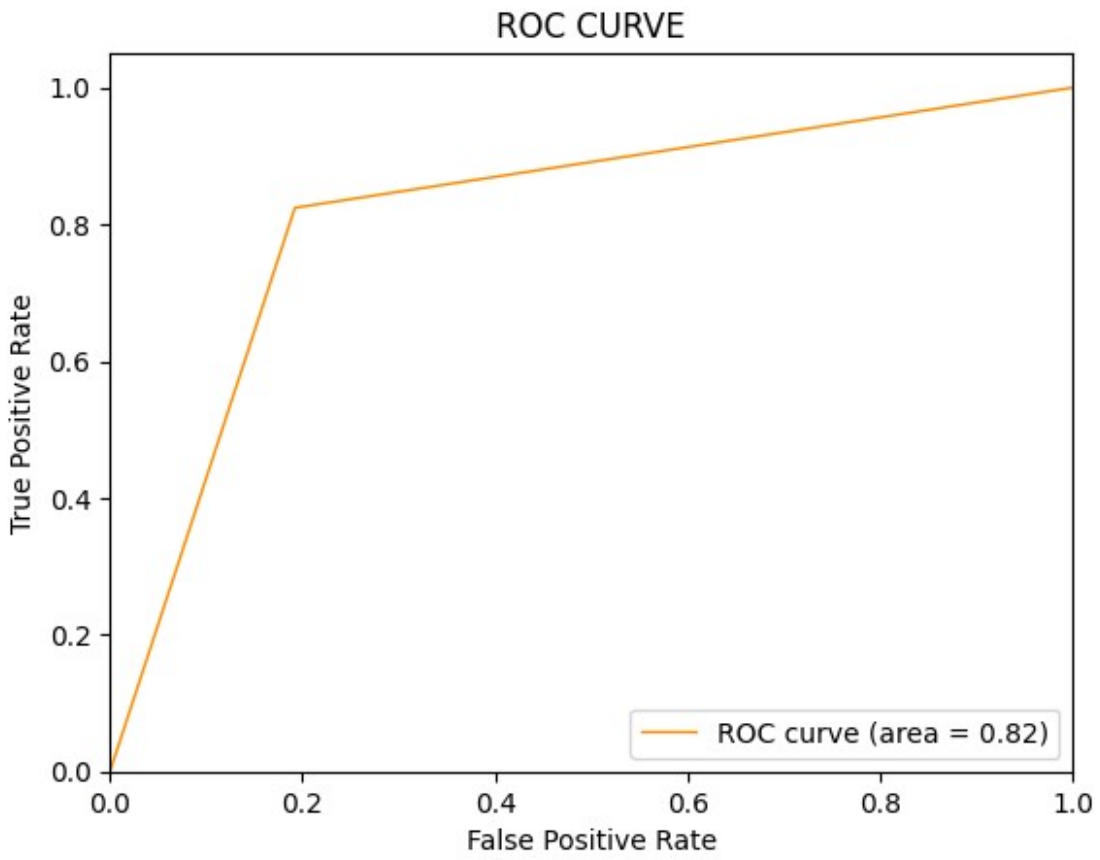



```
# Model-2

SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel)
y_pred2 = SVCmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

           0       0.82      0.81      0.81     40100
           1       0.81      0.82      0.82     39900

    accuracy                           0.82     80000
   macro avg       0.82      0.82      0.82     80000
weighted avg       0.82      0.82      0.82     80000
```

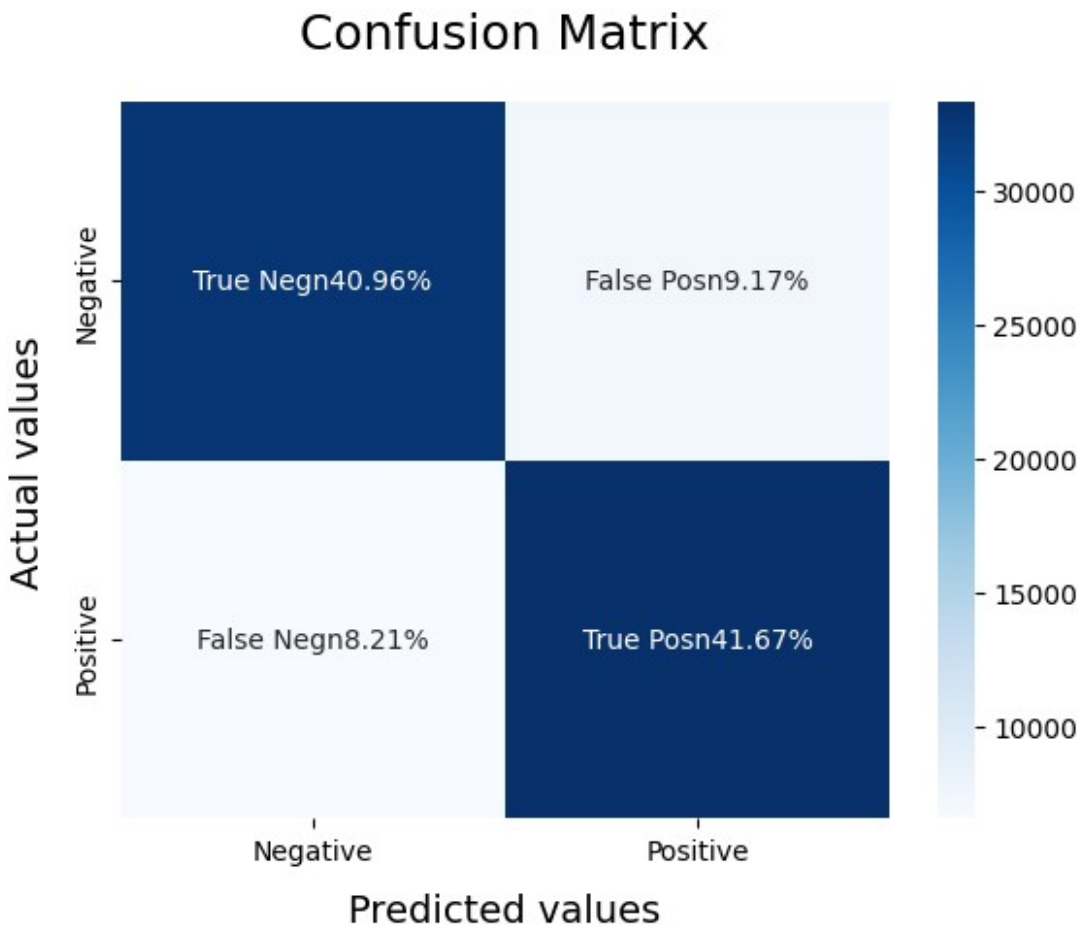



```
# Plot the ROC-AUC Curve for model-2

from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area =
%0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

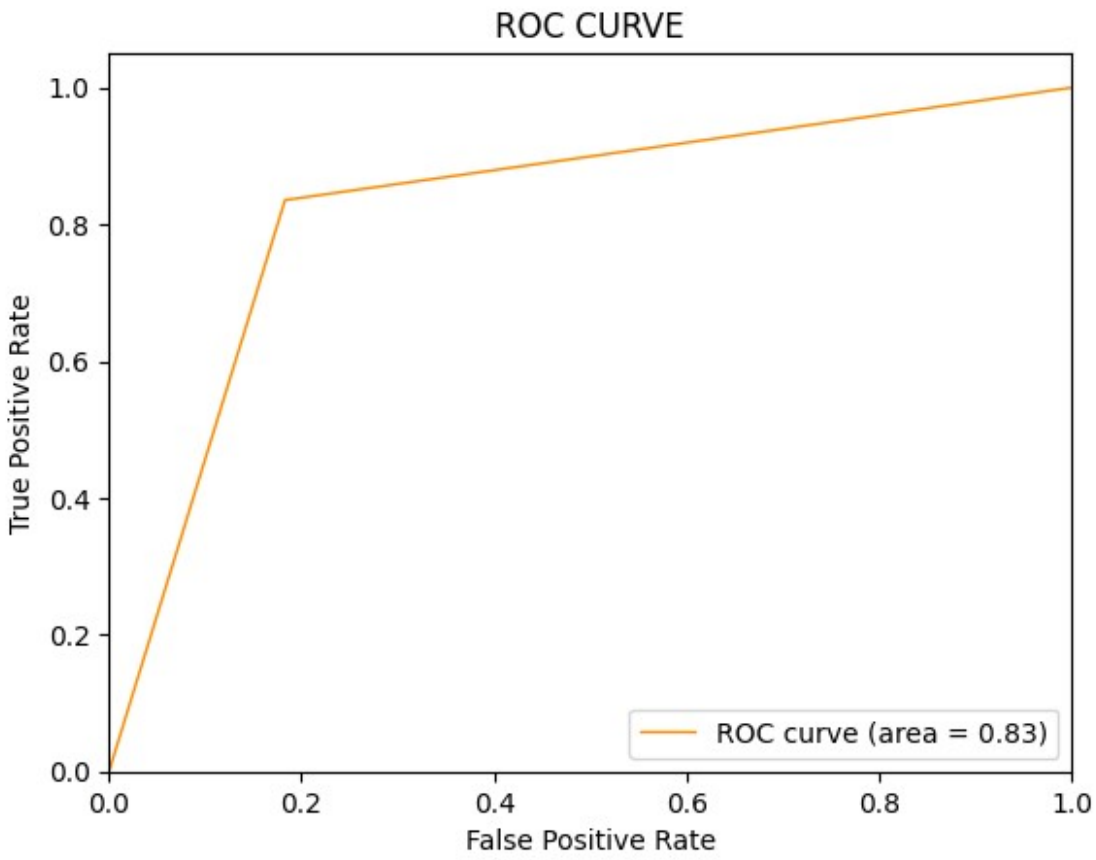



```
# Model-3

LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
y_pred3 = LRmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

           0       0.83      0.82      0.83     40100
           1       0.82      0.84      0.83     39900

    accuracy                           0.83     80000
   macro avg       0.83      0.83      0.83     80000
weighted avg       0.83      0.83      0.83     80000
```




```
# Plot the ROC-AUC Curve for model-3

from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred3)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area =
%0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```


ROC CURVE
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.83)