In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv('emotionClassifier.txt', sep=';', header=None, names=['senter
```

In [3]:
```
df.head()
```

Out[3]:

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

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16000 entries, 0 to 15999
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   sentence  16000 non-null  object
 1   emotion   16000 non-null  object
dtypes: object(2)
memory usage: 250.1+ KB
```

In [8]:
```
df['emotion'].unique()
```

Out[8]:
```
array(['sadness', 'anger', 'love', 'surprise', 'fear', 'joy'],
      dtype=object)
```

In [9]:
```
df["label"]=df["emotion"].map(
{
"sadness":0,
"anger":1,
"love":2,
"surprise":3,
"fear":4,
"joy":5
}
)
```

In [10]:
```
df.head()
```

Out[10]:

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

In [ ]:
```

```

In [12]:
```
#splitting the data in training test
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(df.sentence,df.label,test_size=0.20
```

In [13]:
```
xtrain.shape
```

Out[13]:
```
(12800,)
```

In [14]:
```
xtest.shape
```

Out[14]:
```
(3200,)
```

In [16]:
```
ytest.value_counts()
```

Out[16]:
```
5    1072
0     933
1     432
4     387
2     261
3     115
Name: label, dtype: int64
```

In [17]:
```
#build the pipeline
#1. vectorization
#2. model building
```

In [18]:
```
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [19]:
```
pipline=Pipeline(
[
("vectorization",TfidfVectorizer()),
("knn",KNeighborsClassifier())
]
)
```

In [20]:
```
pipline.fit(xtrain,ytrain)
ypred=pipline.predict(xtest)
```

In [21]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.64      0.86      0.74       933
           1       0.74      0.60      0.66       432
           2       0.64      0.44      0.53       261
           3       0.59      0.30      0.40       115
           4       0.72      0.60      0.66       387
           5       0.75      0.73      0.74      1072

    accuracy                           0.70      3200
   macro avg       0.68      0.59      0.62      3200
weighted avg       0.70      0.70      0.69      3200
```

In [22]:
```
xtest[:5]
```

Out[22]:
```
9785           i feel strong style color black line height
517                              i feel your innocent love
5519     i feel useless and gross and cant seem to find...
5607     i need to do everything i can to push away the...
11078    i refuse to allow my wonderful feeling to be d...
Name: sentence, dtype: object
```

In [23]:
```
ytest[:5]
```

Out[23]:
```
9785     5
517      5
5519     0
5607     0
11078    0
Name: label, dtype: int64
```

In [24]:
```
ypred[:5]
```

Out[24]:
```
array([0, 5, 0, 0, 0])
```

In [25]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [26]:
```
pipline=Pipeline(
[
("vectorization",TfidfVectorizer()),
("rf",RandomForestClassifier())
]
)
```

In [27]:
```
pipline.fit(xtrain,ytrain)
ypred=pipline.predict(xtest)
```

In [28]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.91      0.90       933
           1       0.88      0.79      0.83       432
           2       0.88      0.67      0.76       261
           3       0.79      0.66      0.72       115
           4       0.85      0.78      0.81       387
           5       0.82      0.94      0.88      1072

    accuracy                           0.86      3200
   macro avg       0.85      0.79      0.82      3200
weighted avg       0.86      0.86      0.86      3200
```

In [ ]:
```

```