# NLP CLASSIFICATION MODEL: Emotion classification

In [2]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\Vaish\Desktop\NLP(AD)\emotion.csv")
df.head()
```

Out[2]:

| | text | label |
|---|---|---|
| **0** | i didnt feel humiliated | 0 |
| **1** | i can go from feeling so hopeless to so damned... | 0 |
| **2** | im grabbing a minute to post i feel greedy wrong | 3 |
| **3** | i am ever feeling nostalgic about the fireplac... | 2 |
| **4** | i am feeling grouchy | 3 |

In [6]:
```
df.shape
```

Out[6]:
```
(16000, 2)
```

In [8]:
```
df.info
```

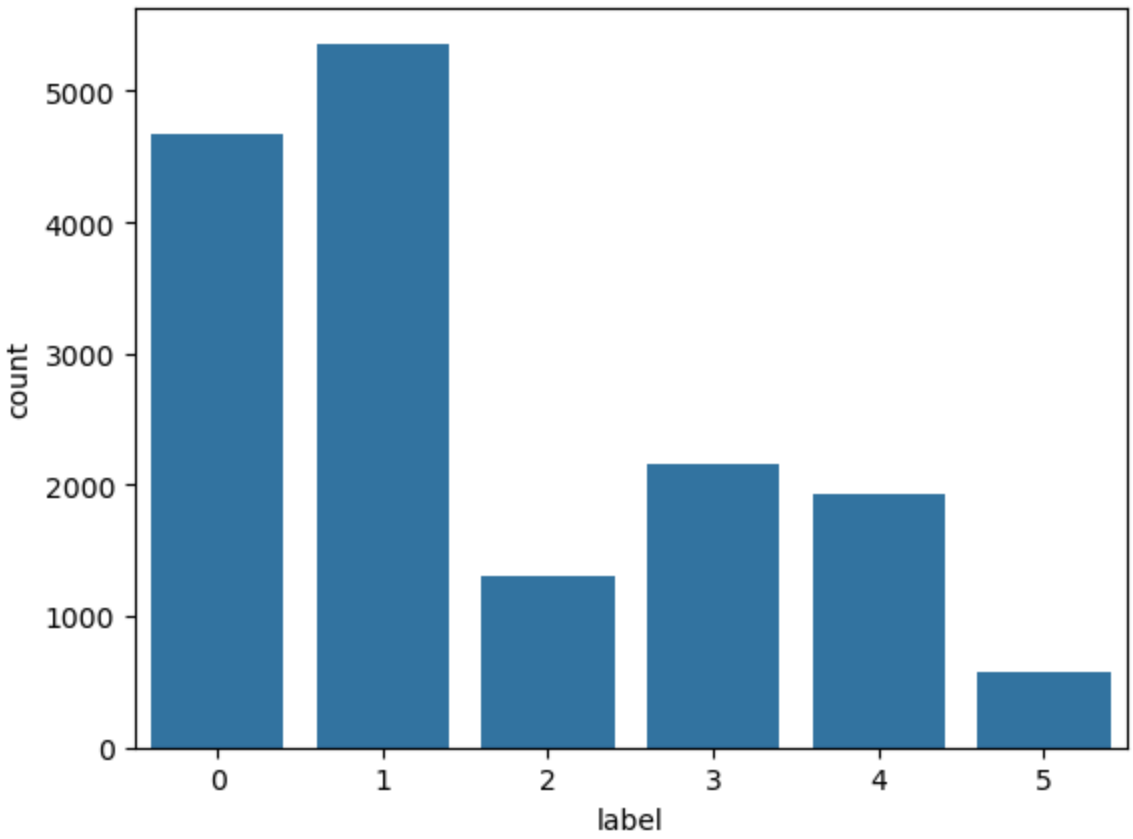
Out[8]:
```
<bound method DataFrame.info of
text  label
0                                i didnt feel humiliated      0
1      i can go from feeling so hopeless to so damned...      0
2       im grabbing a minute to post i feel greedy wrong      3
3      i am ever feeling nostalgic about the fireplac...      2
4                                  i am feeling grouchy      3
...                                                  ...    ...
15995  i just had a very brief time in the beanbag an...      0
15996  i am now turning and i feel pathetic that i am...      0
15997                    i feel strong and good overall      1
15998  i feel like this was such a rude comment and i...      3
15999  i know a lot but i feel so stupid because i ca...      0

[16000 rows x 2 columns]>
```

In [10]:
```
df.label.value_counts()
```

Out[10]:
```
label
1    5362
0    4666
3    2159
4    1937
2    1304
5     572
Name: count, dtype: int64
```

In [12]:
```
import seaborn as sns
sns.countplot(x = df.label)
```

Out[12]:
```
<Axes: xlabel='label', ylabel='count'>
```




In [15]:
```
df.isna().sum()
```

Out[15]:
```
text     0
label    0
dtype: int64
```

# Text processing

In [21]:
```
# convert text to lower case
df['text'] = df['text'].apply(lambda x: " ".join(x.lower() for x in x.split()))
```

In [23]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Vaish\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[23]:
```
True
```

In [25]:
```
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

```
df['text'] = df['text'].apply(lambda x:" ".join(x for x in  x.split() if x not in s
```

In [35]:
```
# Do Lemmatization
from nltk.stem import WordNetLemmatizer
from textblob import Word
df['text'] = df['text'].apply(lambda x: " ".join([Word(word).lemmatize() for word i
df['text'].head()
```

Out[35]:
```
0                                didnt feel humiliated
1    go feeling hopeless damned hopeful around some...
2            im grabbing minute post feel greedy wrong
3    ever feeling nostalgic fireplace know still pr...
4                                      feeling grouchy
Name: text, dtype: object
```

In [42]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(df['text'])
X=X.toarray()
y=df.label.values
```

In [46]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =0.2, random_state=0
```

# Model Building

In [54]:
```
from sklearn.naive_bayes import GaussianNB
#Initialize GaussianNB classifier
model = GaussianNB()
#Fit the model on the train dataset
model = model.fit(X_train, y_train)
#Make predictions on the test dataset
pred = model.predict(X_test)
```

In [55]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [56]:
```
print(confusion_matrix(y_test, pred))
```

```
[[293 116 117 144 171  64]
 [175 388 193  96 132  69]
 [ 48  60 103  15  32  13]
 [ 84  85  53 146  58  33]
 [ 84  61  38  37 147  30]
 [ 23  16  10   5  20  41]]
```

In [60]:
```
print(accuracy_score(y_test,pred))
```

```
0.349375
```

In [62]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.41      0.32      0.36       905
           1       0.53      0.37      0.44      1053
           2       0.20      0.38      0.26       271
           3       0.33      0.32      0.32       459
           4       0.26      0.37      0.31       397
           5       0.16      0.36      0.22       115

    accuracy                           0.35      3200
   macro avg       0.32      0.35      0.32      3200
weighted avg       0.40      0.35      0.36      3200
```

In [66]:
```
from sklearn.ensemble import RandomForestClassifier
clf_rf=RandomForestClassifier()
clf_rf.fit(X_train,y_train)
rf_pred = clf_rf.predict(X_test).astype(int)
```

In [67]:
```
print(confusion_matrix(y_test,rf_pred))
```

```
[[826  34   7  19  18   1]
 [ 25 979  25  10   9   5]
 [  3  63 201   2   2   0]
 [ 32  32   2 384   8   1]
 [ 10  18   3  18 334  14]
 [  4  14   0   0   9  88]]
```

In [68]:
```
print(accuracy_score(y_test,rf_pred))
```

```
0.87875
```

In [69]:
```
print(classification_report(y_test,rf_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.91      0.92       905
           1       0.86      0.93      0.89      1053
           2       0.84      0.74      0.79       271
           3       0.89      0.84      0.86       459
           4       0.88      0.84      0.86       397
           5       0.81      0.77      0.79       115

    accuracy                           0.88      3200
   macro avg       0.87      0.84      0.85      3200
weighted avg       0.88      0.88      0.88      3200
```

# Logistic Regression

In [75]:
```
from sklearn.linear_model import LogisticRegression
logreg= LogisticRegression()
logreg.fit(X_train,y_train)
lr_pred = logreg.predict(X_test)
```

In [77]:
```
print(confusion_matrix(y_test,lr_pred))
```

```
[[ 848   37    1   12    7    0]
 [  19 1015   12    3    2    2]
 [  18   90  155    5    3    0]
 [  51   55    2  348    3    0]
 [  26   49    2   26  287    7]
 [  12   27    0    0   13   63]]
```

In [79]:
```
print(classification_report(y_test,lr_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.90       905
           1       0.80      0.96      0.87      1053
           2       0.90      0.57      0.70       271
           3       0.88      0.76      0.82       459
           4       0.91      0.72      0.81       397
           5       0.88      0.55      0.67       115

    accuracy                           0.85      3200
   macro avg       0.87      0.75      0.80      3200
weighted avg       0.86      0.85      0.84      3200
```

In [81]:
```
print(accuracy_score(y_test,lr_pred))
```

```
0.84875
```

# Conclusion

In [85]:
```
Random forest is the best model with accuracy 88%
```

In [ ]:
```
```