```
225229124
```

## EXERCISE-1

In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv("train.tsv",sep='\t')
df.head()
```

Out[2]:

| | PhraseId | SentenceId | Phrase | Sentiment |
|---|---|---|---|---|
| **0** | 1 | 1 | A series of escapades demonstrating the adage ... | 1 |
| **1** | 2 | 1 | A series of escapades demonstrating the adage ... | 2 |
| **2** | 3 | 1 | A series | 2 |
| **3** | 4 | 1 | A | 2 |
| **4** | 5 | 1 | series | 2 |

In [3]:

```
df.shape
```

Out[3]:

```
(156060, 4)
```

In [4]:

```
df.describe()
```

Out[4]:

| | PhraseId | SentenceId | Sentiment |
|---|---|---|---|
| **count** | 156060.000000 | 156060.000000 | 156060.000000 |
| **mean** | 78030.500000 | 4079.732744 | 2.063578 |
| **std** | 45050.785842 | 2502.764394 | 0.893832 |
| **min** | 1.000000 | 1.000000 | 0.000000 |
| **25%** | 39015.750000 | 1861.750000 | 2.000000 |
| **50%** | 78030.500000 | 4017.000000 | 2.000000 |
| **75%** | 117045.250000 | 6244.000000 | 3.000000 |
| **max** | 156060.000000 | 8544.000000 | 4.000000 |

In [5]:

```
df.columns
```

Out[5]:

```
Index(['PhraseId', 'SentenceId', 'Phrase', 'Sentiment'], dtype='object')
```

In [6]:

```
df.Sentiment.value_counts()
```

Out[6]:

```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

## EXERCISE-2

In [7]:

```
zero = df.loc[df.Sentiment == 0]
one = df.loc[df.Sentiment == 1]
two = df.loc[df.Sentiment == 2]
three = df.loc[df.Sentiment == 3]
four = df.loc[df.Sentiment == 4]
```

In [8]:

```
small_rotten_train = pd.concat([zero[:200],one[:200],two[:200],three[:200],four[:200]])
```

## EXERCISE-3

In [9]:

```
small_rotten_train.to_csv("small_rotten_train.csv")
```

In [10]:

```
X = small_rotten_train.Phrase
```

In [11]:

```
y = small_rotten_train.Sentiment
```

In [12]:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\8mpra\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\8mpra\AppData\Roaming\nltk_data...
```

Out[12]:

```
True
```

In [13]:

```
stop_words = set(stopwords.words('english'))
```

In [14]:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [15]:

```
def clean_review(review):
    tokens = review.lower().split()
    filtered_tokens = [lemmatizer.lemmatize(w)
        for w in tokens if w not in stop_words]
    return " ".join(filtered_tokens)
```

In [19]:

```
f=[]
for i in X.tolist():
    f.append(clean_review(i))
n = pd.Series(f)
```

In [17]:

```
import nltk
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\8mpra\AppData\Roaming\nltk_data...
```

Out[17]:

```
True
```

In [20]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(n,y,test_size=0.20,random_state=42)
```

In [22]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(min_df =3,max_features =None,ngram_range = (1,2), use_idf=1)
```

In [24]:

```
X_train_NB = tfidf.fit_transform(X_train)
X_test_NB = tfidf.transform(X_test)
```

In [25]:

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X_train_NB,y_train)
```

Out[25]:

```
MultinomialNB()
```

In [30]:

```
y_pred_NB= clf.predict(X_test_NB)
y_pred_NB
```

Out[30]:

```
array([2, 1, 1, 3, 2, 4, 2, 4, 4, 0, 4, 0, 1, 4, 4, 4, 1, 4, 3, 1, 3, 3,
       0, 2, 1, 2, 0, 4, 0, 0, 1, 0, 3, 2, 0, 0, 1, 1, 0, 0, 1, 3, 1, 4,
       3, 1, 0, 3, 2, 1, 2, 4, 1, 2, 1, 0, 2, 2, 0, 4, 0, 0, 0, 1, 3, 3,
       3, 4, 0, 2, 0, 2, 0, 1, 1, 3, 3, 0, 0, 1, 4, 0, 1, 0, 1, 2, 4, 3,
       3, 1, 1, 2, 2, 2, 3, 0, 3, 0, 0, 0, 3, 1, 4, 1, 2, 3, 0, 4, 0, 4,
       2, 3, 4, 4, 0, 2, 0, 0, 2, 1, 2, 2, 0, 0, 2, 2, 1, 2, 4, 3, 2, 3,
       1, 4, 2, 2, 0, 2, 1, 4, 0, 1, 3, 0, 2, 4, 4, 2, 3, 1, 4, 0, 0, 2,
       4, 2, 1, 3, 0, 4, 2, 3, 0, 3, 4, 3, 3, 3, 0, 1, 3, 2, 0, 4, 0, 1,
       1, 0, 1, 0, 2, 3, 1, 2, 4, 0, 1, 1, 4, 3, 4, 4, 3, 0, 2, 2, 1, 1,
       3, 0], dtype=int64)
```

In [31]:

```
from sklearn.metrics import accuracy_score,classification_report
accuracy_score(y_test,y_pred_NB)
```

Out[31]:

```
0.68
```

In [33]:

```
print(classification_report(y_test,y_pred_NB))
```

```
              precision    recall  f1-score   support

           0       0.52      0.79      0.63        33
           1       0.73      0.62      0.67        48
           2       0.68      0.73      0.70        37
           3       0.69      0.63      0.66        38
           4       0.85      0.66      0.74        44

    accuracy                           0.68       200
   macro avg       0.69      0.69      0.68       200
weighted avg       0.70      0.68      0.68       200
```

## EXERCISE-4

In [35]:

```
df1 = pd.read_csv("test.tsv",sep='\t')
df1.head()
```

Out[35]:

| | PhraseId | SentenceId | Phrase |
|---|---|---|---|
| **0** | 156061 | 8545 | An intermittently pleasing but mostly routine ... |
| **1** | 156062 | 8545 | An intermittently pleasing but mostly routine ... |
| **2** | 156063 | 8545 | An |
| **3** | 156064 | 8545 | intermittently pleasing but mostly routine effort |
| **4** | 156065 | 8545 | intermittently pleasing but mostly routine |

In [36]:

```
X2 = df1['Phrase']
```

In [37]:

```
X2 = X2.apply(lambda X2: clean_review(X2))
```

In [39]:

```
X2_test = tfidf.transform(X2)
```

In [41]:

```
y_pred_2 = clf.predict(X2_test)
y_pred_2
```

Out[41]:

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

In [ ]:

```
```