5. Text Classification (Sentiment Analysis)

March 29, 2022

[1]:
```
import pandas as pd
data=pd.read_csv('sentiment_train.tsv', sep='\t')
data.head()
```

[1]:
```
   PhraseId  SentenceId                                             Phrase  \
0         1           1  A series of escapades demonstrating the adage …
1         2           1  A series of escapades demonstrating the adage …
2         3           1                                           A series
3         4           1                                                  A
4         5           1                                             series

   Sentiment
0          1
1          2
2          2
3          2
4          2
```

[2]:
```
data.info()
```

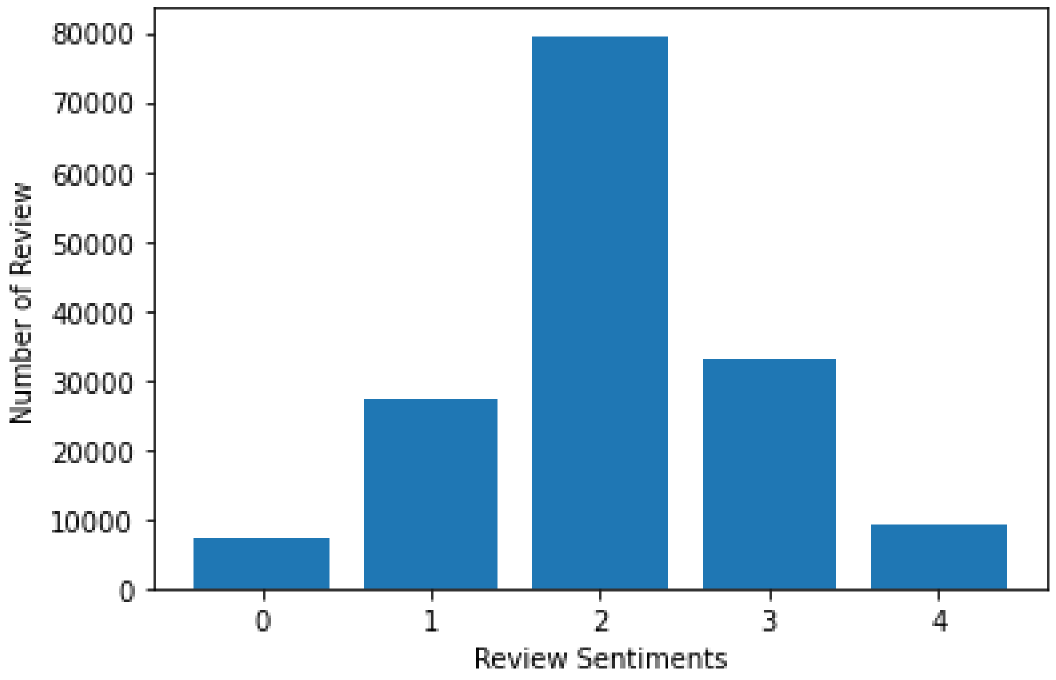
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156060 entries, 0 to 156059
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   PhraseId    156060 non-null  int64
 1   SentenceId  156060 non-null  int64
 2   Phrase      156060 non-null  object
 3   Sentiment   156060 non-null  int64
dtypes: int64(3), object(1)
memory usage: 4.8+ MB
```

[3]:
```
data.Sentiment.value_counts()
```

[3]:
```
2    79582
3    32927
1    27273
```

```
4     9206
0     7072
Name: Sentiment, dtype: int64
```

[4]:
```
import matplotlib.pyplot as plt
Sentiment_count=data.groupby('Sentiment').count()
plt.bar(Sentiment_count.index.values, Sentiment_count['Phrase'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```



[5]:
```
# Bag-of-Words feature extraction: frequency of every word in each document

from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
#tokenizer to remove unwanted elements from out data like symbols and numbers
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range =␣
 ↪(1,1),tokenizer = token.tokenize)
text_counts= cv.fit_transform(data['Phrase'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_counts,␣
 ↪data['Sentiment'], test_size=0.3, random_state=1)
```

```python
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
score=metrics.accuracy_score(y_test, predicted)
score
```

[5]:
```
0.6049169122986885
```

[6]:
```python
# TF-IDF feature extraction

from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
text_tf= tf.fit_transform(data['Phrase'])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(text_tf, data['Sentiment'],␣
 ↪test_size=0.3, random_state=123)

from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
metrics.accuracy_score(y_test, predicted)
score
```

[6]:
```
0.6049169122986885
```