In [1]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
import re
from bs4 import BeautifulSoup

import warnings
warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ajeya\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ajeya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\ajeya\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [2]:
```
! pip install bs4
! pip install pandas
! pip install numpy
! pip install nltk
! pip install contractions
! pip install sklearn
```

```
Requirement already satisfied: bs4 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from beautifulsoup4->bs4) (2.3.2.post1)
Requirement already satisfied: pandas in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (1.1.5)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.15.4 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from pandas) (1.19.5)
Requirement already satisfied: pytz>=2017.2 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from pandas) (2022.2.1)
Requirement already satisfied: six>=1.5 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
Requirement already satisfied: numpy in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (1.19.5)
Requirement already satisfied: nltk in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (3.7)
Requirement already satisfied: click in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: regex>=2021.8.3 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from nltk) (2022.8.17)
Requirement already satisfied: tqdm in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from nltk) (4.64.0)
Requirement already satisfied: joblib in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: colorama in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from click->nltk) (0.4.4)
Requirement already satisfied: importlib-metadata in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from click->nltk) (4.8.3)
Requirement already satisfied: importlib-resources in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from click->nltk) (5.2.3)
Requirement already satisfied: zipp>=3.5 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from importlib-metadata->click->nltk) (3.6.0)
Requirement already satisfied: typing-extensions>=3.6.4 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from importlib-metadata->click->nltk) (4.0.1)
Requirement already satisfied: contractions in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: anyascii in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from contractions) (0.3.1)
Requirement already satisfied: pyahocorasick in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: sklearn in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from sklearn) (0.24.2)
Requirement already satisfied: scipy>=0.19.1 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from scikit-learn->sklearn) (1.5.4)
Requirement already satisfied: numpy>=1.13.3 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from scikit-learn->sklearn) (1.19.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from scikit-learn->sklearn) (3.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\ajeya\appdata\local\programs\python\python36\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
```

Read Data

Read data as pandas dataframe providing local path for the dataset file

In [4]:
```
df = pd.read_csv("C:/Users/ajeya/OneDrive - University of Southern California/Desktop/
```

Keep Reviews and Ratings

Retaining only reviews and ratings field. Also filtering reviews which has empty review body.

In [5]:
```
data = df[['review_body', 'star_rating']]
data = data.data['review_body'].str.len() > 0 ]
```

We select 20000 reviews randomly from each rating class.

Since the dataset is big, we're taking a sample of 20000 records from each class and combining the data to get 100K records.

In [6]:
```
#Separating the data based on star_rating

rating_1 = data[data['star_rating']==1]

rating_2 = data[data['star_rating']==2]

rating_3 = data[data['star_rating']==3]

rating_4 = data[data['star_rating']==4]

rating_5 = data[data['star_rating']==5]

#Taking a sample of 20000 from each rating value

rating_1 = rating_1.sample(n=20000, random_state=2)
rating_2 = rating_2.sample(n=20000, random_state=2)
rating_3 = rating_3.sample(n=20000, random_state=2)
rating_4 = rating_4.sample(n=20000, random_state=2)
rating_5 = rating_5.sample(n=20000, random_state=2)
```

Data Cleaning

Data Cleaning includes following tasks: 1) Remove hyperlinks 2) Remove HTML 3) Convert review to lowercase 4) Fix Contractions 5) Remove non-alphabetical characters 6) Remove whitespaces

Pre-processing

Data preprocessing includes following tasks: 1) Remove stopwords 2) Perform lemmatization

In [7]:
```
import contractions

#Combining data from all types of ratings
review_data = pd.concat([rating_1, rating_2,rating_3, rating_4,rating_5])
review_data = review_data.sample(frac = 1)

#Converting review_body to String
review_data["review_body"] = review_data["review_body"].apply(str)
total_characters_init = 0
total_reviews = len(review_data)
for index, row in review_data.iterrows():
    total_characters_init = total_characters_init + len(row["review_body"])

#Calculating average review length before data cleaning
average_review_length = total_characters_init/total_reviews
#print("Average Characters per review before data cleaning: ",average_review_length)
print("Total reviews before data cleaning: ",total_reviews)

#Function for data cleaning
def clean_review(text):
    regex_alpha = '[^a-zA-Z]'
    regex_html = '<.*?>'
    regex_url = r'http[s]?://\S+'
    #Remove URL from reviews
    text = re.sub(r'http[s]?://\S+', "", text)
    #Remove HTML from reviews
    text = re.sub(regex_html, ' ', text)
    #Convert to lower letters
    text = text.lower()
    #Fix Contractions
    text = contractions.fix(text)
    #Remove non alphabetical characters
    text = re.sub(regex_alpha, ' ', text)
    text = ' '.join(text.split())
    return text


review_data["review_body"] = review_data["review_body"].apply(clean_review)

#Calculating average review length after data cleaning
total_characters = 0
for index, row in review_data.iterrows():
    total_characters = total_characters + len(row["review_body"])

average_review_length = total_characters/total_reviews
#print("Average Characters per review after data cleaning: ",average_review_length)
print("Total reviews_len_before,",",average_review_length )
```

```
188.25037 , 181.49296
```

remove the stop words

In [8]:
```
from nltk.corpus import stopwords
```

perform lemmatization

In [9]:
```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
stop_words = set(stopwords.words("english"))

#print("Average Characters per review before data preprocessing: ",average_review_len

#Function for data preprocessing
def preprocessing(text):
    word_list = word_tokenize(text)
    word_list = [w for w in word_list if not w.lower() in stop_words]
    word_list = map(lemmatizer.lemmatize, word_list)
    #stem_list = [stemmer.stem(word) for word in word_list]
    text = ' '.join(word_list)
    return text

review_data["review_body"] = review_data["review_body"].apply(preprocessing)

#Calculating average length of reviews after data preprocessing
total_characters_preprocess = 0
for index, row in review_data.iterrows():
    total_characters_preprocess = total_characters_preprocess + len(row["review_body

average_review_length_preprocess = total_characters_preprocess/total_reviews
#print("Average Characters per review after data preprocessing: ",average_review_leng
print(average_review_length,",", average_review_length_preprocess)
```

```
181.49296 , 107.18811
```

TF-IDF Feature Extraction

In [10]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

#Divide the set to reviews and labels
X = review_data.review_body
review_data['star_rating'] = review_data['star_rating'].apply(int)
Y = review_data.star_rating

#Divide data into train and test set, train set has 80% of data and test set has 20%
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2)

#TF-IDF Feature Extraction
tfidfvectorizer = TfidfVectorizer(analyzer='word',stop_words= 'english')
tfidfvectorizer.fit(X)

#print("No. of feature_words: ", len(tfidfvectorizer.get_feature_names()))

#Form features set from tfidf vectorizer
X_train_feature = tfidfvectorizer.transform(X_train)
X_test_feature  = tfidfvectorizer.transform(X_test)
```

Perceptron

In [18]:
```
from sklearn.linear_model import Perceptron
from sklearn import metrics

#Function to print results in required format
def print_result(scores):
    for i in range(1,6):
        class_value = str(i)
        precision = str(round(scores[class_value]['precision'],2))
        recall = str(round(scores[class_value]['recall'],2))
        f1 = str(round(scores[class_value]['f1-score'],2))
        print(precision,",",recall,",", f1)
    avg_precision =  str(round(scores['weighted avg']['precision'],2))
    avg_recall = str(round(scores['weighted avg']['recall'],2))
    avg_f1 = str(round(scores['weighted avg']['f1-score'],2))
    print(avg_precision,",", avg_recall,",", avg_f1)

#Fitting sklearn perceptron model and predicting the ratings
perceptron = Perceptron(
                max_iter=50,
                tol=0.001)
perceptron.fit(X_train_feature, Y_train)
prediction_perceptron = perceptron.predict(X_test_feature)
perceptron_results = metrics.classification_report(Y_test, prediction_perceptron, outp

print_result(perceptron_results)
#print(metrics.classification_report(Y_test, prediction_perceptron))
```

```
0.48 , 0.52 , 0.5
0.31 , 0.32 , 0.32
0.35 , 0.29 , 0.32
0.37 , 0.37 , 0.37
0.54 , 0.6 , 0.57
0.4 , 0.41 , 0.4
```

SVM

In [11]:
```
from sklearn.svm import LinearSVC

#Fitting SVM model and predicting the ratings
SVCmodel = LinearSVC(C = 0.5)
SVCmodel.fit(X_train_feature, Y_train)
prediction_svm = SVCmodel.predict(X_test_feature)

svm_results = metrics.classification_report(Y_test, prediction_svm,output_dict=True)
print_result(svm_results)
#print(metrics.classification_report(Y_test, prediction_svm))
```

```
0.55 , 0.64 , 0.59
0.38 , 0.34 , 0.36
0.39 , 0.35 , 0.36
0.44 , 0.4 , 0.42
0.6 , 0.72 , 0.65
0.47 , 0.49 , 0.48
```

Logistic Regression

In [13]:
```
from sklearn.linear_model import LogisticRegression

#Fitting sklearn logistic regression model and predicting the ratings
LogisticRegressionModel = LogisticRegression(C = 1, max_iter = 1000)
LogisticRegressionModel.fit(X_train_feature, Y_train)
prediction_logistic_regression = LogisticRegressionModel.predict(X_test_feature)

logistic_regression_results = metrics.classification_report(Y_test, prediction_logist
print_result(logistic_regression_results)
#print(metrics.classification_report(Y_test, prediction_logistic_regression))
```

```
0.58 , 0.63 , 0.6
0.39 , 0.38 , 0.38
0.41 , 0.38 , 0.39
0.46 , 0.43 , 0.44
0.63 , 0.67 , 0.65
0.49 , 0.5 , 0.5
```

Naive Bayes

In [14]:
```
from sklearn.naive_bayes import MultinomialNB

#Fitting sklearn Naive Bayes model and predicting the ratings
naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(X_train_feature, Y_train)

prediction = naive_bayes_classifier.predict(X_test_feature)

naive_bayes_results = metrics.classification_report(Y_test, prediction,output_dict=Tr
print_result(naive_bayes_results)
#print(metrics.classification_report(Y_test, prediction))
```

```
0.59 , 0.58 , 0.58
0.38 , 0.39 , 0.39
0.39 , 0.37 , 0.38
0.43 , 0.42 , 0.42
0.63 , 0.64 , 0.64
0.48 , 0.48 , 0.48
```