Import libraries

In [70]:

```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
from bs4 import BeautifulSoup
import os
import contractions
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('omw-1.4')

from sklearn.metrics import precision_score, recall_score, f1_score, classification_report

import re
import pickle
from emot.emo_unicode import UNICODE_EMOJI # For emojis
from emot.emo_unicode import EMOTICONS_EMO # For EMOTICONS
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/venkatasaisumanthsadu/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/venkatasaisumanthsadu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/venkatasaisumanthsadu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/venkatasaisumanthsadu/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [71]:

```
!pip install bs4 # in case you don't have it installed
!pip install emot
```

```
Requirement already satisfied: bs4 in /Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in /Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-packag
es (from bs4) (4.11.1)
Requirement already satisfied: soupsieve>1.2 in /Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-package
s (from beautifulsoup4->bs4) (2.3.1)
Requirement already satisfied: emot in /Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-packages (3.1)
```

Read Data

In [72]:

```
#to get the current working directory
directory = os.getcwd()
url = os.path.join(directory, "data.tsv")
df = pd.read_csv(url, sep='\t', header=0, on_bad_lines='skip')
```

```
/var/folders/dz/k6x51lvd2jv050r966v_h2br0000gn/T/ipykernel_6120/1309074324.py:4: DtypeWarning: Columns (8) have mixed
types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(url, sep='\t', header=0, on_bad_lines='skip')
```

Keep Reviews and Ratings

In [73]:

```
df = df[['review_body','star_rating']]
```

We form three classes and select 20000 reviews randomly from each class.

In [74]:

```
df = df[df['star_rating'].eq(1) | df['star_rating'].eq(2) | df['star_rating'].eq(3) | df['star_rating'].eq(4) | df['sta
df['class'] = df['star_rating'].apply(lambda x: 1 if x in [1, 2] else 2 if x == 3 else 3)
df.head(2)
```

Out[74]:

	review_body	star_rating	class
32768	I have a bunch of these color-changing gel pol...	5	3
32769	this product smells like it's been dipped in f...	1	1

In [75]:

```
# there are few nan values in the dataframe, also removing duplicate rows
df = df.dropna()
df = df.drop_duplicates()
df = df.reset_index()
```

In [76]:

```
class1 = df[df['class']==1].sample(n=20000, random_state=42)
class2 = df[df['class']==2].sample(n=20000, random_state=42)
class3 = df[df['class']==3].sample(n=20000, random_state=42)
df = pd.concat([class1, class2, class3])
```

In [77]:

```
# average length before data cleaning:
print('Average length of the reviews before data cleaning :', (df['review_body'].str.len()).mean())
```

```
Average length of the reviews before data cleaning : 289.2713
```

Data Cleaning

In [78]:

```
def remove_urls(text):
    text = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', text, flags=re.MULTILINE)
    return (text)

def remove_contractions(text) :
    expanded_words = []
    for word in text.split():
        expanded_words.append(contractions.fix(word))
        expanded_text = ' '.join(expanded_words)
    return expanded_text


remove_non_english = lambda s: re.sub(r'[^a-zA-z]', ' ', s)
remove_spaces = lambda s: re.sub(' +',' ', s)
```

In [79]:

```
def cleaning(text):
    #remove urls
    text = remove_urls(text)
    #remove html tags
    text = BeautifulSoup(text, "lxml").text
    #remove contractions
    text = remove_contractions (text)
    #remove non-alphabetic chars
    text = remove_non_english(text)
    #lowercase
    text = text.lower( )
    #remove extra spaces
    text = remove_spaces(text)

    return text
```

In [80]:

```
df['cleaned_text_reviews'] = list(map(cleaning, df.review_body))
```

```
/Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-packages/bs4/__init__.py:435: MarkupResemblesLocatorWar
ning: The input looks more like a filename than markup. You may want to open this file and pass the filehandle into B
eautiful Soup.
  warnings.warn(
```

In [81]:

```
def convert_emojis(text):
    for emot in UNICODE_EMOJI:
        text = text.replace(emot, "_".join(UNICODE_EMOJI[emot].replace(",","").replace(":","").split()))
    return text

df['cleaned_text_reviews'] = df['cleaned_text_reviews'].apply(lambda row: convert_emojis(str(row)))
```

In [82]:

```
# average length after data cleaning:
print('Average length of the reviews after data cleaning :', (df['cleaned_text_reviews'].str.len()).mean())
```

```
Average length of the reviews after data cleaning : 279.3924166666667
```

Pre-processing

In [83]:

```
# average length before pre-processing:
print('Average length of the reviews before pre-processing :', (df['cleaned_text_reviews'].str.len()).mean())
```

```
Average length of the reviews before pre-processing : 279.3924166666667
```

In [84]:

```
df2 = df.copy()
```

remove the stop words

```
In [85]: from nltk.corpus import stopwords

         to_remove = ['not']
         new_stopwords = set(stopwords.words('english')).difference(to_remove)

         df2['cleaned_text_reviews'] = df2['cleaned_text_reviews'].apply(lambda x: " ".join(x for x in x.split() if x not in new
```

perform lemmatization

```
In [86]: from nltk.stem import WordNetLemmatizer
         lemmatizer = WordNetLemmatizer()

         df2['cleaned_text_reviews'] = df2['cleaned_text_reviews'].apply(lambda x: " ".join([lemmatizer.lemmatize(word) for word
```

```
In [87]: # average length after pre-processing:
         print('Average length of the reviews after pre-processing :', (df2['cleaned_text_reviews'].str.len()).mean())

         Average length of the reviews after pre-processing : 172.00831666666667
```

TF-IDF Feature Extraction

```
In [88]: from sklearn.model_selection import train_test_split
         x_train,x_valid,y_t,y_v = train_test_split(df2['cleaned_text_reviews'],df2['class'],test_size=0.2, stratify = df2['clas
```

```
In [89]: from sklearn.feature_extraction.text import TfidfVectorizer

         tfidf = TfidfVectorizer(ngram_range=(1, 3))
         tfidf.fit(df2['cleaned_text_reviews'])

Out[89]: TfidfVectorizer(ngram_range=(1, 3))
```

```
In [90]: x_t = tfidf.transform(x_train)
         x_v = tfidf.transform(x_valid)
```

Perceptron

```
In [91]: from sklearn.linear_model import Perceptron
         p = Perceptron()
         p.fit(x_t, y_t)

Out[91]: Perceptron()
```

```
In [92]: report = classification_report(y_v, p.predict(x_v), output_dict=True )
```

```
In [93]: print('Perceptron:')
         print('Class 1: Precision - ', report['1']['precision'], ', Recall - ', report['1']['recall'], ', F1-score - ', report[
         print('Class 2: Precision - ', report['2']['precision'], ', Recall - ', report['2']['recall'], ', F1-score - ', report[
         print('Class 3: Precision - ', report['3']['precision'], ', Recall - ', report['3']['recall'], ', F1-score - ', report[
         print('Average: Precision - ', precision_score(y_v, p.predict(x_v), average='micro'), ', Recall - ', recall_score(y_v,

         Perceptron:
         Class 1: Precision -  0.673538740371545 , Recall -  0.74325 , F1-score -  0.7066793439505585
         Class 2: Precision -  0.6193907631837536 , Recall -  0.47275 , F1-score -  0.5362257195519636
         Class 3: Precision -  0.7182881094198103 , Recall -  0.814 , F1-score -  0.7631548107347942
         Average: Precision -  0.6766666666666666 , Recall -  0.6766666666666666 , F1-score -  0.6766666666666666
```

SVM

```
In [94]: from sklearn.svm import LinearSVC
         sv = LinearSVC()
         sv.fit(x_t, y_t)

Out[94]: LinearSVC()
```

In [95]:
```
report_sv = classification_report(y_v, sv.predict(x_v), output_dict=True )
```

In [96]:
```
print('SVM:')
print('Class 1: Precision - ', report_sv['1']['precision'], ', Recall - ', report_sv['1']['recall'], ', F1-score - ', r
print('Class 2: Precision - ', report_sv['2']['precision'], ', Recall - ', report_sv['2']['recall'], ', F1-score - ', r
print('Class 3: Precision - ', report_sv['3']['precision'], ', Recall - ', report_sv['3']['recall'], ', F1-score - ', r
print('Average: Precision - ', precision_score(y_v, sv.predict(x_v), average='micro'), ', Recall - ', recall_score(y_v,
```

```
SVM:
Class 1: Precision -  0.7066477407144547 , Recall -  0.74675 , F1-score -  0.7261456180867875
Class 2: Precision -  0.6310975609756098 , Recall -  0.56925 , F1-score -  0.5985804416403786
Class 3: Precision -  0.7771908763505402 , Recall -  0.80925 , F1-score -  0.792896509491733
Average: Precision -  0.7084166666666667 , Recall -  0.7084166666666667 , F1-score -  0.7084166666666667
```

Logistic Regression

In [97]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_t, y_t)
```

```
/Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/mo
dules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[97]:
```
LogisticRegression()
```

In [98]:
```
report_lr = classification_report(y_v, lr.predict(x_v), output_dict=True )
```

In [99]:
```
print('Logistic Regression:')
print('Class 1: Precision - ', report_lr['1']['precision'], ', Recall - ', report_lr['1']['recall'], ', F1-score - ', r
print('Class 2: Precision - ', report_lr['2']['precision'], ', Recall - ', report_lr['2']['recall'], ', F1-score - ', r
print('Class 3: Precision - ', report_lr['3']['precision'], ', Recall - ', report_lr['3']['recall'], ', F1-score - ', r
print('Average: Precision - ', precision_score(y_v, lr.predict(x_v), average='micro'), ', Recall - ', recall_score(y_v,
```

```
Logistic Regression:
Class 1: Precision -  0.7166585246702492 , Recall -  0.7335 , F1-score -  0.7249814677538918
Class 2: Precision -  0.6244518957957184 , Recall -  0.60525 , F1-score -  0.6147010283102704
Class 3: Precision -  0.7803425167535368 , Recall -  0.786 , F1-score -  0.7831610412255572
Average: Precision -  0.70825 , Recall -  0.70825 , F1-score -  0.70825
```

Naive Bayes

In [100]:
```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(x_t, y_t)
```

Out[100]:
```
MultinomialNB()
```

In [101]:
```
report_nb = classification_report(y_v, nb.predict(x_v), output_dict=True )
```

In [102]:
```
print('Naive Bayes:')
print('Class 1: Precision - ', report_nb['1']['precision'], ', Recall - ', report_nb['1']['recall'], ', F1-score - ', r
print('Class 2: Precision - ', report_nb['2']['precision'], ', Recall - ', report_nb['2']['recall'], ', F1-score - ', r
print('Class 3: Precision - ', report_nb['3']['precision'], ', Recall - ', report_nb['3']['recall'], ', F1-score - ', r
print('Average: Precision - ', precision_score(y_v, nb.predict(x_v), average='micro'), ', Recall - ', recall_score(y_v,
```

```
Naive Bayes:
Class 1: Precision -  0.7185148018063221 , Recall -  0.716 , F1-score -  0.7172551965940395
Class 2: Precision -  0.6042411246128186 , Recall -  0.634 , F1-score -  0.61876296205929
Class 3: Precision -  0.7982708933717579 , Recall -  0.76175 , F1-score -  0.7795829602149162
Average: Precision -  0.7039166666666666 , Recall -  0.7039166666666666 , F1-score -  0.7039166666666666
```

In [ ]:

In [ ]:

```

```

Model training without doing stopwords preprocessing

perform lemmatization

In [103]:

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

df['cleaned_text_reviews'] = df['cleaned_text_reviews'].apply(lambda x: " ".join([lemmatizer.lemmatize(word) for word ir
```

TF-IDF Feature Extraction

In [104]:

```
from sklearn.model_selection import train_test_split
x_train2 ,x_valid2 ,y_t2 ,y_v2 = train_test_split(df['cleaned_text_reviews'],df['class'],test_size=0.2, stratify = df['

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(ngram_range=(1, 3))
tfidf.fit(df['cleaned_text_reviews'])
x_t2 = tfidf.transform(x_train2)
x_v2 = tfidf.transform(x_valid2)
```

Perceptron

In [105]:

```
from sklearn.linear_model import Perceptron
p = Perceptron()
p.fit(x_t2, y_t2)
report = classification_report(y_v2, p.predict(x_v2), output_dict=True )
```

In [106]:

```
print('Perceptron')
print('Class 1: Precision, Recall, F1-score - ', report['1']['precision'],report['1']['recall'],report['1']['f1-score']
print('Class 2: Precision, Recall, F1-score - ', report['2']['precision'],report['2']['recall'],report['2']['f1-score']
print('Class 3: Precision, Recall, F1-score - ', report['3']['precision'],report['3']['recall'],report['3']['f1-score']
print('Average: Precision, Recall, F1-score - ', precision_score(y_v2, p.predict(x_v2), average='micro'), recall_score(
```

```
Perceptron
Class 1: Precision, Recall, F1-score -  0.7090085795996187 0.74375 0.7259638848218644
Class 2: Precision, Recall, F1-score -  0.648235294117647 0.551 0.5956756756756757
Class 3: Precision, Recall, F1-score -  0.7606721162579473 0.8375 0.7972394098048549
Average: Precision, Recall, F1-score -  0.71075 0.71075 0.71075
```

SVM

In [107]:

```
from sklearn.svm import LinearSVC
sv = LinearSVC()
sv.fit(x_t2, y_t2)
report_sv = classification_report(y_v2, sv.predict(x_v2), output_dict=True )
```

In [108]:

```
print('SVM:')
print('Class 1: Precision, Recall, F1-score - ', report_sv['1']['precision'],report_sv['1']['recall'],report_sv['1']['f
print('Class 2: Precision, Recall, F1-score - ', report_sv['2']['precision'],report_sv['2']['recall'],report_sv['2']['f
print('Class 3: Precision, Recall, F1-score - ', report_sv['3']['precision'],report_sv['3']['recall'],report_sv['3']['f
print('Average: Precision, Recall, F1-score - ', precision_score(y_v2, sv.predict(x_v2), average='micro'), recall_score
```

```
SVM:
Class 1: Precision, Recall, F1-score -  0.739343459088682 0.7545 0.746844840386043
Class 2: Precision, Recall, F1-score -  0.6593085106382979 0.61975 0.6389175257731958
Class 3: Precision, Recall, F1-score -  0.8027898027898028 0.8345 0.8183378278989948
Average: Precision, Recall, F1-score -  0.73625 0.73625 0.73625
```

Logistic Regression

In [109]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_t2, y_t2)
report_lr = classification_report(y_v2, lr.predict(x_v2), output_dict=True )
```

```
/Users/venkatasaisumanthsadu/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/mo
dules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [110]:

```
print('Logistic Regression:')
print('Class 1: Precision, Recall, F1-score - ', report_lr['1']['precision'],report_lr['1']['recall'],report_lr['1']['f
print('Class 2: Precision, Recall, F1-score - ', report_lr['2']['precision'],report_lr['2']['recall'],report_lr['2']['f
print('Class 3: Precision, Recall, F1-score - ', report_lr['3']['precision'],report_lr['3']['recall'],report_lr['3']['f
print('Average: Precision, Recall, F1-score - ', precision_score(y_v2, lr.predict(x_v2), average='micro'), recall_score
```

```
Logistic Regression:
Class 1: Precision, Recall, F1-score -  0.7415338645418327 0.7445 0.7430139720558881
Class 2: Precision, Recall, F1-score -  0.6545500762582613 0.64375 0.6491051172170407
Class 3: Precision, Recall, F1-score -  0.8037037037037037 0.81375 0.8086956521739131
Average: Precision, Recall, F1-score -  0.734 0.734 0.734
```

Naive Bayes

In [111]:

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(x_t2, y_t2)
report_nb = classification_report(y_v2, nb.predict(x_v2), output_dict=True )
```

In [112]:

```
print('Naive Bayes:')
print('Class 1: Precision, Recall, F1-score - ', report_nb['1']['precision'],report_nb['1']['recall'],report_nb['1']['f
print('Class 2: Precision, Recall, F1-score - ', report_nb['2']['precision'],report_nb['2']['recall'],report_nb['2']['f
print('Class 3: Precision, Recall, F1-score - ', report_nb['3']['precision'],report_nb['3']['recall'],report_nb['3']['f
print('Average: Precision, Recall, F1-score - ', precision_score(y_v2, nb.predict(x_v2), average='micro'), recall_score
```

```
Naive Bayes:
Class 1: Precision, Recall, F1-score -  0.7529777317452098 0.727 0.7397608750953956
Class 2: Precision, Recall, F1-score -  0.6170583115752829 0.709 0.6598417868776175
Class 3: Precision, Recall, F1-score -  0.8565782044042913 0.7585 0.8045611243701936
Average: Precision, Recall, F1-score -  0.7315 0.7315 0.7315
```