Q1. Dataset Preparation

In data preparation I have first extracted the columns of importance that is star rating and the review body. To reduce the dataset size I have converted the star ratings 1 and 2 to form class 1, star rating 3 to class 2 and star ratings 4 and 5 to class 3. Further, I have extracted 20,000 rows of each class to form a dataset of 60,000 rows.

Q2. Dataset Cleaning

I have removed any extra spaces, punctuation, any other non-alphabetic letters, pointless html links, etc. It reduces the average length of the evaluations. Further, I have gotten rid of contractions (won't -> will not), which adds to the lengthening. Overall, the length falls, although not significantly.
Average length of the reviews in terms of character length before cleaning - 188.91358333333332
Average length of the reviews in terms of character length after cleaning - 182.34115

Q3. Data Preprocessing

In data preprocessing I removed the stop words(commin and repitative words like ‘a’,’the’,’so’) and performed lemmatization(converts running to run, eats to eat, etc.). using the nltk library.
Average length of the reviews in terms of character length before preprocessing- 188.91358333333332
Average length of the reviews in terms of character length after preprocessing - 182.34115

Q4. Feature Extraction

For feature extraction we have used TF-IDF(term frequency-inverse document)

Q5. Perceptron

```
0.6298258894776685, 0.5705142857142858, 0.5987047253538019
0.4970104633781764, 0.5160372478013451, 0.5063451776649746
0.6594581158339547, 0.7057728119180633, 0.6818298637882293
0.5954314895632665, 0.597441448477898, 0.5956265889356686
```

Q6. SVM

```
0.7009841029523088, 0.6820525411244782, 0.6913887506222
0.6018933731938216, 0.6038490377405649, 0.602869619463506
0.7529207059408402, 0.7715231788079471, 0.7621084413133727
0.6852660606956569, 0.6858082525576634, 0.685455603799693
```

Q7. Logistic regression

```
0.7158718142821096, 0.6851002173388071, 0.7001480750246792
0.593423019431988, 0.6177385892116183, 0.6053367217280813
0.7633606761123539, 0.7671746190357233, 0.7652628955893347
0.6908851699421504, 0.690004475195383, 0.6902492307806983
```

Q8. Multinomial Naïve Bayes

```
0.6507696189755235, 0.7092959295929593, 0.6787735228319516
0.6434977578475336, 0.5866454689984102, 0.6137578709754069
0.7434750186428039, 0.7551123453673315, 0.7492484969939879
0.679247465155287, 0.683684581319567, 0.6805932969337821
```

```
#Palak Umesh Mallawat
#Python 3
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
import re
import contractions
from bs4 import BeautifulSoup
```

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

Read Data

```
df=pd.read_table('amazon_reviews_us_Beauty_v1_00.tsv.gz', on_bad_lines = 'skip')
```

```
C:\Users\Palak\AppData\Local\Temp\ipykernel_26688\3934172176.py:1: DtypeWarning: Columns (7) have mixed t
  df=pd.read_table('amazon_reviews_us_Beauty_v1_00.tsv.gz', on_bad_lines = 'skip')
```

Keep Reviews and Ratings

```
df['review_body']=df['review_body'].apply(str)
dff = df.loc[:, ['star_rating','review_body']]
```

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

```
df1=dff.loc[df['star_rating'].isin(['1','2'])]
df1['star_rating']=1
df2=dff.loc[df['star_rating'] == '3']
df2['star_rating']=2
df3=dff.loc[df['star_rating'].isin(['4','5'])]
df3['star_rating']=3
df1=df1.sample(n=20000)
df2=df2.sample(n=20000)
df3=df3.sample(n=20000)
dfr=pd.concat([df1,df2,df3])
```

```
C:\Users\Palak\AppData\Local\Temp\ipykernel_26688\4135629575.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
  df1['star_rating']=1
C:\Users\Palak\AppData\Local\Temp\ipykernel_26688\4135629575.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
  df2['star_rating']=2
C:\Users\Palak\AppData\Local\Temp\ipykernel_26688\4135629575.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  df3['star_rating']=3
```

Data Cleaning

Pre-processing

```
avgCharLengthBeforeCleaning=dfr['review_body'].str.len().mean()
dfr['review_body'] = dfr['review_body'].str.lower()
dfr['review_body']=dfr['review_body'].apply(str)
 # strip html with BeautifulSoup
dfr['review_body'] = [BeautifulSoup(text).get_text() for text in dfr['review_body'] ]
# remove non alphabetic. keep spaces
dfr['review_body'] = dfr['review_body'].str.replace('[^a-zA-Z ]', '')
# strip leading and trailing spaces. strip extra white spaces
dfr['review_body'] = dfr['review_body'].str.strip()
# handle contractions
dfr['review_body'] = [contractions.fix(text) for text in dfr['review_body'] ]
# get average length of reviews after cleaning
avgCharLengthAfterCleaning=dfr['review_body'].str.len().mean()
print("Printing the average character count before and after cleaning  "+ str(avgCharLengthBeforeCleaning) + ",
```

```
C:\Users\Palak\anaconda3\lib\site-packages\bs4\__init__.py:435: MarkupResemblesLocatorWarning: The input l
  warnings.warn(
C:\Users\Palak\AppData\Local\Temp\ipykernel_26688\1062857879.py:7: FutureWarning: The default value of re
  dfr['review_body'] = dfr['review_body'].str.replace('[^a-zA-Z ]', '')
Printing the average character count before and after cleaning  188.72826666666666, 182.21608333333333
```

remove the stop words

```
from nltk.corpus import stopwords
nltk.download('omw-1.4')
nltk.download('stopwords')
stop = stopwords.words('english')
dfr['review_body'] = dfr['review_body'].apply(lambda x: ' '.join([word for word in x.split() if word not in (st
```

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

perform lemmatization

```
from nltk.stem import WordNetLemmatizer
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]
```

```
dfr['review_body'] = dfr['review_body'].apply(lemmatize_text)
dfr['review_body'] = dfr['review_body'].apply(str)
avgCharLengthAfterProcessing=dfr['review_body'].str.len().mean()
print("Printing the average character count before and after Preprocessing  "+ str(avgCharLengthAfterCleaning)
```

```
Printing the average character count before and after Preprocessing  182.21608333333333, 162.73295
```

TF-IDF Feature Extraction

```
from sklearn.feature_extraction.text import TfidfVectorizer
v = TfidfVectorizer()
x = v.fit_transform(dfr['review_body'])
# x.shape
```

function for printing values in the required format

```
def printValues(value):
    print(str(value['1']['precision']) + ", " + str(value['1']['recall']) + ", " + str(value['1']['f1-score']))
    print(str(value['2']['precision']) + ", " + str(value['2']['recall']) + ", " + str(value['2']['f1-score']))
    print(str(value['3']['precision']) + ", " + str(value['3']['recall']) + ", " + str(value['3']['f1-score']))
    print(str(value['macro avg']['precision']) + ", " + str(value['macro avg']['recall']) + ", " + str(value['m
```

spliting the data into train and test

```
from pandas.core.common import random_state
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, dfr['star_rating'], test_size = 0.2, random_state = 42)
y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

Perceptron

```
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report
p = Perceptron(n_jobs = -1, max_iter = 10000, random_state = 42)
p.fit(X_train, y_train)
printValues(classification_report(p.predict(X_test), y_test, output_dict=True))
```

```
0.6298258894776685, 0.5705142857142858, 0.5987047253538019
0.4970104633781764, 0.5160372478013451, 0.5063451776649746
0.6594581158339547, 0.7057728119180633, 0.6818298637882293
0.5954314895632665, 0.597441448477898, 0.5956265889356686
```

SVM

```
from sklearn import svm
```

```
from sklearn import metrics
sv = svm.SVC(kernel='linear')
sv.fit(X_train, y_train)
printValues(classification_report(sv.predict(X_test), y_test, output_dict=True))
```

```
     0.7009841029523088, 0.6820525411244782, 0.6913887506222
     0.6018933731938216, 0.6038490377405649, 0.602869619463506
     0.7529207059408402, 0.7715231788079471, 0.7621084413133727
     0.6852660606956569, 0.6858082525576634, 0.685455603799693
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
classification_report(lr.predict(X_test), y_test, output_dict=True)
printValues(classification_report(lr.predict(X_test), y_test, output_dict=True))
```

```
     0.7158718142821096, 0.6851002173388071, 0.7001480750246792
     0.593423019431988, 0.6177385892116183, 0.6053367217280813
     0.7633606761123539, 0.7671746190357233, 0.7652628955893347
     0.6908851699421504, 0.690004475195383, 0.6902492307806983
```

Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
mltnb = MultinomialNB()
mltnb.fit(X_train, y_train)
printValues(classification_report(mltnb.predict(X_test), y_test, output_dict=True))
```

```
     0.6507696189755235, 0.7092959295929593, 0.6787735228319516
     0.6434977578475336, 0.5866454689984102, 0.6137578709754069
     0.7434750186428039, 0.7551123453673315, 0.7492484969939879
     0.679247465155287, 0.683684581319567, 0.6805932969337821
```

Colab paid products - Cancel contracts here