CSCI 544 HW 1.

1. Data Preparation

a. Include 3 sample reviews in your report.

⇨ We use the following code to get this output. After selecting only the star_ratings and review_body columns, we use pandas head(3) method to get first 3 records.

```
rev_rat = amazon_reviews[['star_rating','review_body']]
rev_rat.head(3)
```

	star_rating	review_body
0	5.0	Beautiful. Looks great on counter.
1	5.0	I personally have 5 days sets and have also bo...
2	5.0	Fabulous and worth every penny. Used for clean...

b. Statistics of star ratings.

⇨ We can get the counts of each star by selecting the star_rating column and then apply pandas value_counts() method which returns count of unique values in sorted order.

```
# Statistics of ratings
rev_rat['star_rating'].value_counts()
```

```
5.0    3124759
4.0     731733
1.0     426900
3.0     349547
2.0     241948
Name: star_rating, dtype: int64
```

c. Statistics of the 3 classes.

⇨ I first dropped all the NaN type ratings so that it can help me in getting the correct classes. Then, I converted star_rating column to int to apply numpy where clause. After that I created a new column 'class' in which all ratings above 2 where class 1 and others as 0. Since we counted ratings 3 as class 1 then I changed its rating to class 3 meaning a neutral rating. After all this steps I dropped the class 3 as mentioned in the assignment.

```
rev_rat['star_rating']=rev_rat['star_rating'].astype(int) # convert values of star_rating to int so that we can use numpy where clause
rev_rat['class']=np.where(rev_rat['star_rating']<3,0,1) # set class based on the given requirements
rev_rat['class']=np.where(rev_rat['star_rating']==3,3,rev_rat['class']) # we will now change class of ratings 3 as on previous step we added it to class 1
```

```
# Statistics of ratings after classes
rev_rat['class'].value_counts()
```

```
1    3856296
0     668809
3     349539
Name: class, dtype: int64
```

2. Data Cleaning

a. Average length of characters in review before cleaning

⇨ I looped over all the reviews and measured the length of characters and saved it under char_len variable. After that I printed the mean of it.

```
#Average char length in review_body before data cleaning
from statistics import mean
char_len=[len(char) for char in rev_rat['review_body']]
print(mean(char_len))
```

```
323.796825
```

b. Average length of characters in review after cleaning

⇨ Used the same function as above.

```
#Average char length in review_body after data cleaning
from statistics import mean
char_len_after=[len(char) for char in rev_rat['review_body']]
print(mean(char_len_after))
```

```
309.058895
```

3. Preprocessing

a. 3 sample reviews before data Cleaning and Preprocessing.

⇨ I used rev_rat.head(3) to print the 3 sample reviews.

	star_rating	review_body	class
8	5	sharp and look great	1
27	5	I've been waiting my whole life for these!	1
64	5	Good water bottle. Water tastes so much bette...	1

b. Average length of reviews before data preprocessing.

⇨ It will be the same as avg length after data cleaning which is this.

```
309.058895
```

c. 3 sample reviews after data Cleaning and Preprocessing.

⇨ I used rev_rat.head(3) to print the 3 sample reviews.

	star_rating	review_body	class
8	5	sharp look great	1
27	5	I waiting whole life	1
64	5	good water bottle water taste much better old ...	1

d. Average length of reviews after data preprocessing.

⇨ The avg length has been reduced drastically after the preprocessing.

```
#Average char length in review_body after data preprocessing
from statistics import mean
char_len_af_pre=[len(char) for char in rev_rat['review_body']]
print(mean(char_len_af_pre))
✓ 0.5s
191.49201
```

4. Perceptron

a. Report Accuracy, Precision, Recall and F1 Score.

⇨ After training the model with 80% training data and trying different hyperparamteres I got an Accuracy of 85%.

```
...   Perceptron Model
      Accuracy of Test:  85.4275 %
      Precision of Test:  85.46517552113897 %
      Recall of Test:  87.06732216313836 %
      F1 Score of Test:  85.62196295108654 %
      Accuracy of Train:  93.16187500000001 %
      Precision of Train:  93.19302715779874 %
      Recall of Train:  94.4920440636475 %
      F1 Score of Train:  93.2568273005738 %
```

5. SVM
a. Report Accuracy, Precision, Recall and F1 Score
⇨ After training the model with 80% training data and trying different hyperparamteres I got an Accuracy of 89%.

```
...   SVM Model
      Accuracy of Test:  89.3475 %
      Precision of Test:  89.34906891308866 %
      Recall of Test:  89.03882813283836 %
      F1 Score of Test:  89.28292965114817 %
      Accuracy of Train:  94.03375 %
      Precision of Train:  94.0338741763167 %
      Recall of Train:  93.93375465241176 %
      F1 Score of Train:  94.03240729164062 %
```

6. Logistic Regression
a. Report Accuracy, Precision, Recall and F1 Score
⇨ After training the model with 80% training data and trying different hyperparamteres I got an Accuracy of 89.5%.

```
Logistic Regression Model
Accuracy of Test:  89.58500000000001 %
Precision of Test:  89.5887996961635 %
Recall of Test:  89.10905989766229 %
F1 Score of Test:  89.50418220296281 %
Accuracy of Train:  91.31937500000001 %
Precision of Train:  91.32108997598142 %
Recall of Train:  90.9874353658232 %
F1 Score of Train:  91.29701921811655 %
```

7. Multinomial Naïve Bayes

a. Report Accuracy, Precision, Recall and F1 Score

⇨ After training the model with 80% training data and trying different hyperparamteres I got an Accuracy of 86.4%.

```
Multinomial Naive Bayes Model
Accuracy of Test:  86.815 %
Precision of Test:  86.83457319324135 %
Recall of Test:  85.66268686665998 %
F1 Score of Test:  86.62304063308477 %
Accuracy of Train:  89.08625 %
Precision of Train:  89.1171864079181 %
Recall of Train:  87.67516798641121 %
F1 Score of Train:  88.93815961180302 %
```