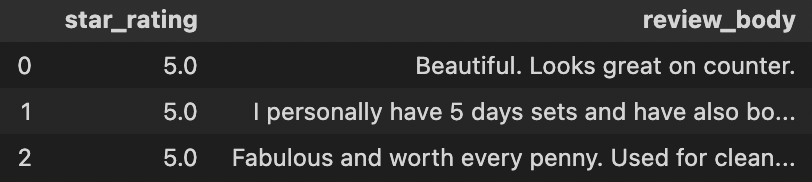
**CSCI 544 HW 1.**

1. **Data Preparation**
2. **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)



1. **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()

A picture containing table

Description automatically generated

1. **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()

Graphical user interface, text, application

Description automatically generated

1. **Data Cleaning**
2. **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))



1. **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))

Graphical user interface

Description automatically generated

1. **Preprocessing**
2. **3 sample reviews before data Cleaning and Preprocessing.**

* I used rev\_rat.head(3) to print the 3 sample reviews.

Graphical user interface, text

Description automatically generated

1. **Average length of reviews before data preprocessing.**

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

Graphical user interface

Description automatically generated

1. **3 sample reviews after data Cleaning and Preprocessing.**

* I used rev\_rat.head(3) to print the 3 sample reviews.

Graphical user interface, text, application

Description automatically generated

1. **Average length of reviews after data preprocessing.**

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

Graphical user interface, text

Description automatically generated

1. Perceptron
2. 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%.

Text

Description automatically generated

1. SVM
2. 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%.

Text

Description automatically generated

1. Logistic Regression
2. 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%.

Text

Description automatically generated

1. Multinomial Naïve Bayes
2. 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%.

Text

Description automatically generated