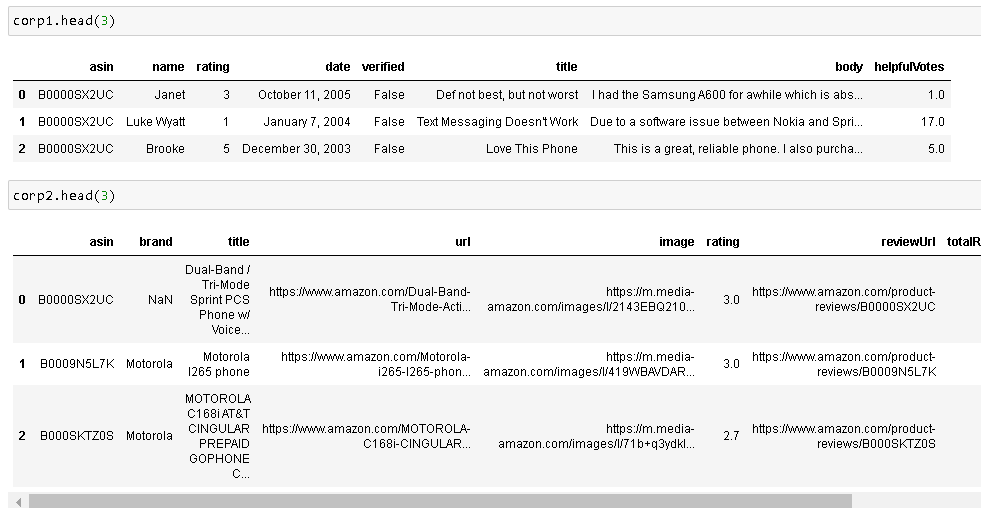
**Q1. Basic Data Understanding**

**1.1 Read data from ‘reviews.csv’ and ‘products.csv’**

****

****

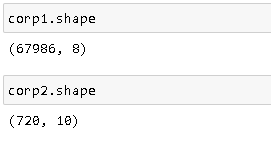
Approach used:

* We use pandas function read\_csv to import a csv file from the given location

Insights:

* Asin is the primary key for a product
* Reviews dataframe contains review, review title, number of people who found the review useful, date of posting and the customer’s name, rating provided for the device and weather or not the customer was valid
* Product\_data contains product key(asin), brand, device name(title), URLs of Review page and Image, and average rating for the product
* Our main focus is in reading the review (body in reviews dataset) and coming up with the sentiment of that review (and by extension, general perception of the product)

**1.2 Print shape of both the dataframes**



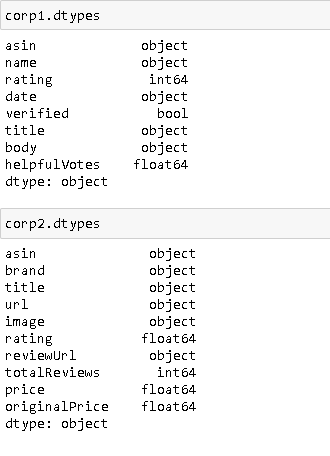
Approach used:

* Use df.shape, which prints rows and columns of a dataset

Insights:

* Corp2 (Product\_data) essentially contains the name of all the models and their respective brands.
* corp1(reviews) data contains reviews for the models.

**1.3 Print types of columns for both the dataframes**

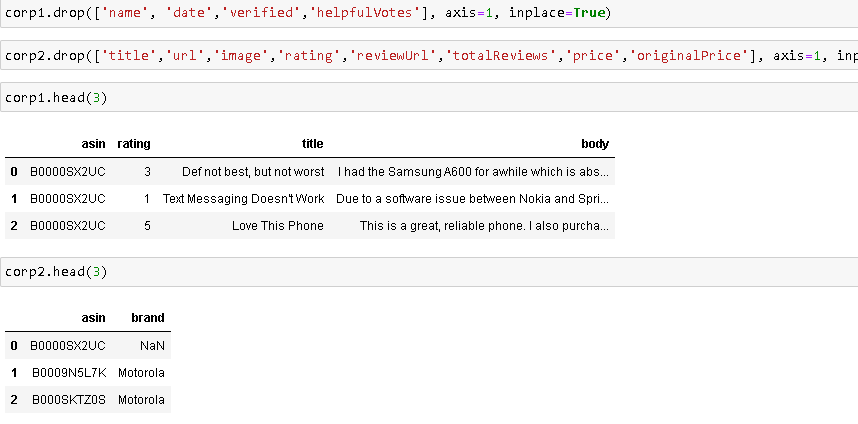
****

Approach used:

* Use df.dtypes to get the data types of all the columns

**Q2. Basic Data Analysis [20 Marks]**

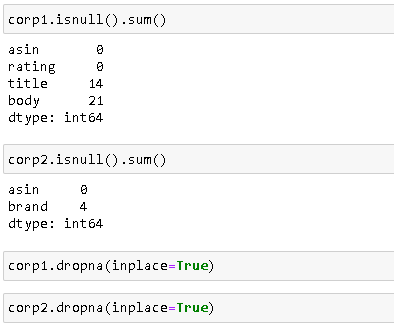
**2.1 Drop all columns except: asin, rating, title, body from reviews And Drop all columns except: asin, brand from products [3 Marks]**

****

Approach used:

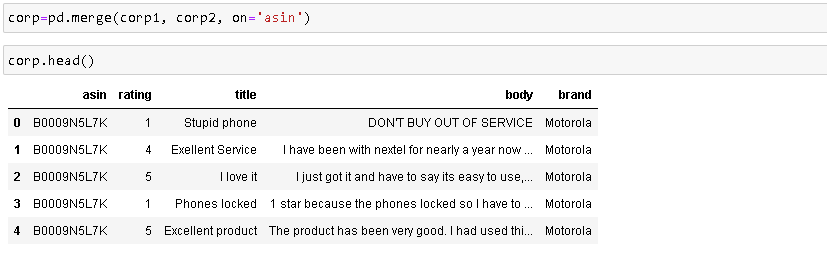
* We use drop() function with inplace=True to drop the said columns in the original dataframe

**2.2 Verify and drop all null values from both the tables [3 Marks]**

****

Approach used:

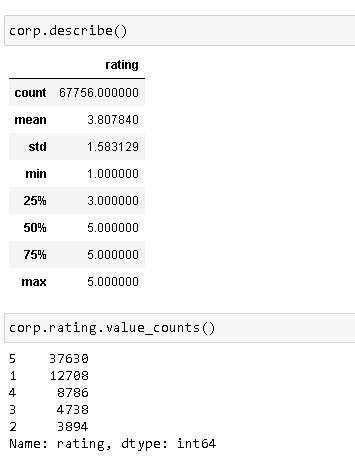
* isnull() function identifies all the null values. Appending sum() to isnull() gives the count of null values (because it perceives True as 1, sum() only counts null rows)
* To drop the null rows, we use dropna() with inplace=True to do it on the original dataframe

**2.3 Merge both the tables on key ‘asin’ [3 Marks] **

Approach used:

* Use pd.merge() function to merge two dataframes passed as parameters. The common key is asin

**2.3 Check the statistical summary of ‘rating’ and share insights [2 Marks]**



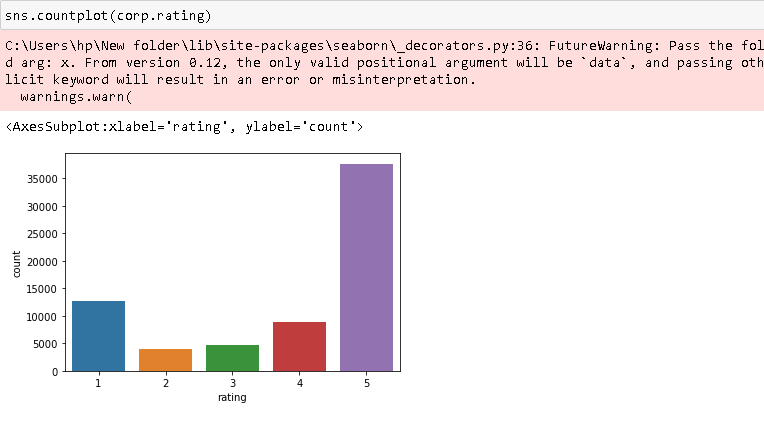
Approach used:

* Rating a numerical attribute (which is input as categorical by the customer).

Insights:

* We can see that most people tend to rate on extremes (either 1 or 5).

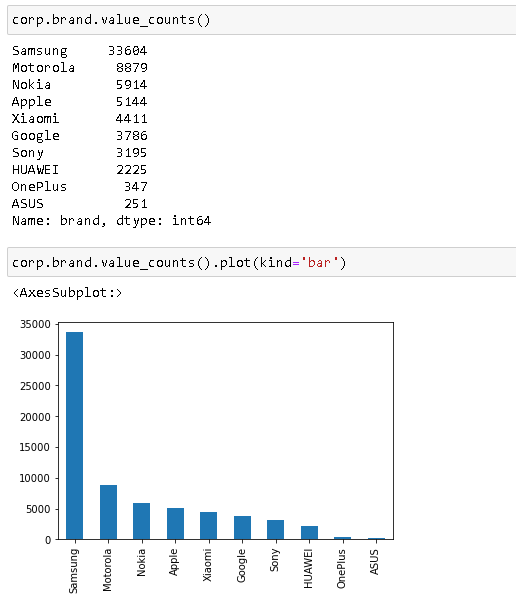
**2.4 Visualize a countplot for rating and share insights [3 Marks]**



Approach used:

* Use sns.countplot() to plot distribution of rating

**2.5 Visualize distribution of ‘brand’ and share insights [3 Marks]**

****

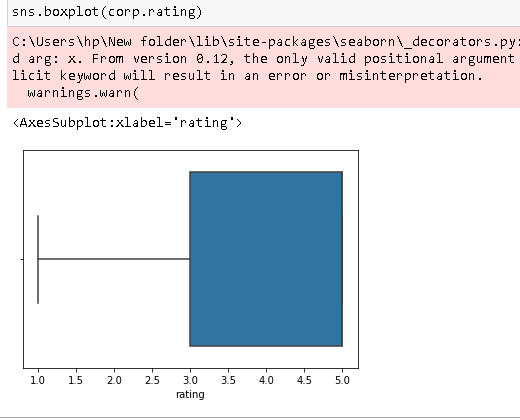
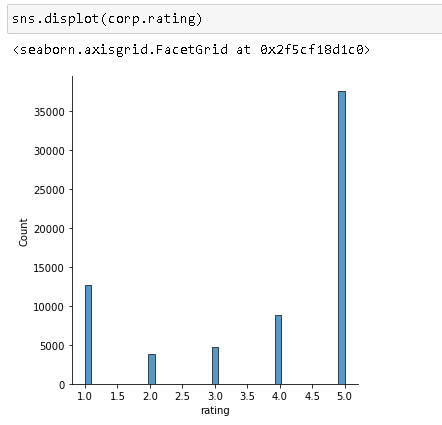
Approach used:

* We use value\_counts() to get number of rows pertaining to the specific brand
* Appending plot() to the previous result gives the result in graphical form

Insights:

* Though this distribution of Brands is taken from reviews posted, it can act as a good approximation
* Hence, we can safely say that Samsung is the largest handset seller, followed by Motorola, Nokia, etc.
* One more thing inferred from the distribution is that we have motorola and Nokia among top three, this implies that the data is very old (probably, some time after when Nokia’s downfall started). It is also superimposed by the absence of major chinese manufacturers like Oppo, Vivo, etc

**2.6 Visualize distribution of ‘rating’ and share insights [3 Marks]**

****

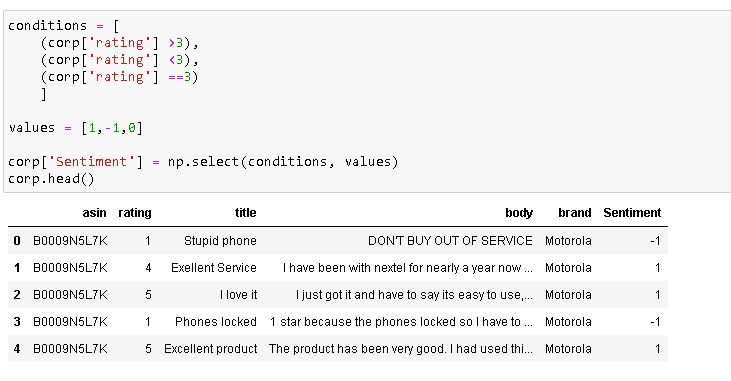
Approach used:

* We use seaborn distplot and boxplot to see the distribution of rating

Insights:

* Customers tend to vote on extremes (5 or 1) and the customers are generally happy with the product

**Q3. Data Preparation [5 Marks] Insert new Column in DataFrame ‘Sentiment’ and Assign Sentiment= 1 when Rating >3; Sentiment= -1 when Rating <3 and Sentiment= 0 when rating = 3. (This step is to create target class where 1 indicates positive sentiment, -1 indicates negative sentiment and 0 indicates neutral sentiment)**



Approach used:

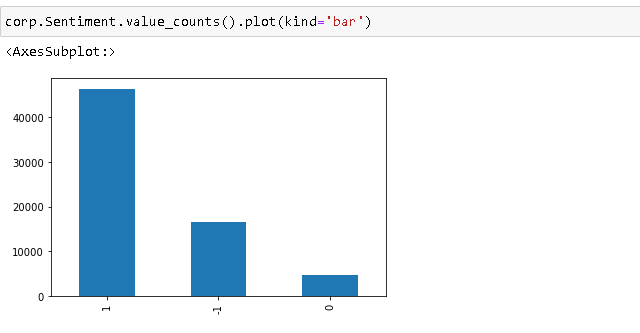
* We first make list of conditions and values pertaining to it
* We then use np.select() to apply it on the dataframe

Insights:

* We can see that sentiment is correctly filled with respect to the reviews

**Q4. Data Visualization [5 Marks]**

**4.1 Visualize distribution of ‘sentiment’ and share insights [3 Marks]**

****

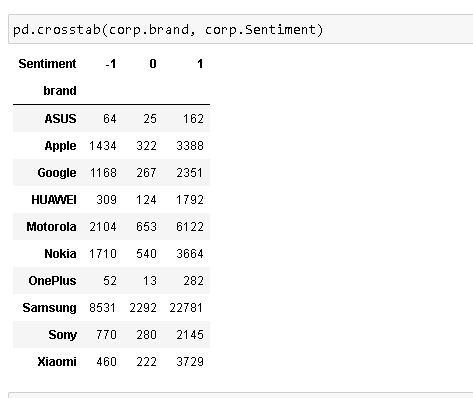
Approach used:

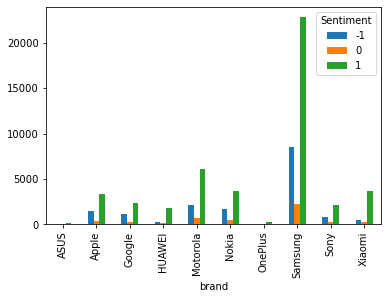
* We use value\_counts() to find number of respondents in each sentiment category (-1,0 & 1)
* We then append plot() to the result which plots the result

Insights:

* We can see that 1 (positive) sentiment is way more than negative and neutral sentiment.

**4.2 Visualize a crosstab of ‘brand’ with sentiment [2 Marks]**

****

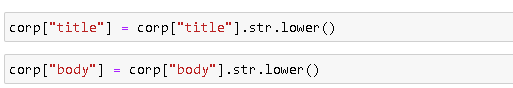
****

Approach used:

* We use pd.crosstab to get a crosstab between brands and their respective sentiment distribution

**Q5. Textual Data Pre-processing [15 Marks]**

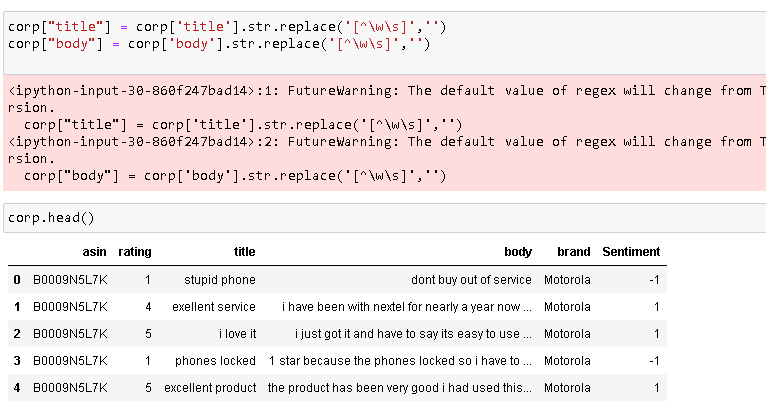
**5.1 Lowercase all data in columns ‘title’ and ‘body’ [3 Marks]**

****

Approach used:

* Use str.lower() function to convert all letters to lower case

**5.2 Remove all punctuations from columns ‘title’ and ‘body’ [3 Marks]**



Approach used:

* Use str.replace() to replace all punctuation marks with ‘’, i.e., removing

**5.3 Remove stopwords from columns ‘title’ and ‘body’ [3 Marks]**

****

Approach used:

* To remove the stopwords, we first need to tokenize all the rows
* We tokenize using re.findall(), which finds all empty spaces and tokenizes the words around it.
* Then define a function that returns the words after comparing their lowercase to the stopwords list we have. If the word is found in stopwords list, that word is not returned.( effectively deleting it)
* Use the function on required columns

**5.4 Transform ‘body’ using TF-IDF vectorizer [3 Marks]**

**5.5 Split data into X and Y. [3 Marks] (Vectorized data should be X and sentiment should be Y.)**

****

Approach used:

* We first lammetize the words using WordnetLemmatizer ()
* Then we join all the tokenized words in the respective columns.
* We use TfidfVectorizer() to calculate Tf-Idf vectors for each row and save it as x
* Convert x into a dataframe (TfidfVectorizer returns array) by using pd.Dataframe
* Now save the target column Sentiment to y

**Q6. Model Building [5 Marks]**

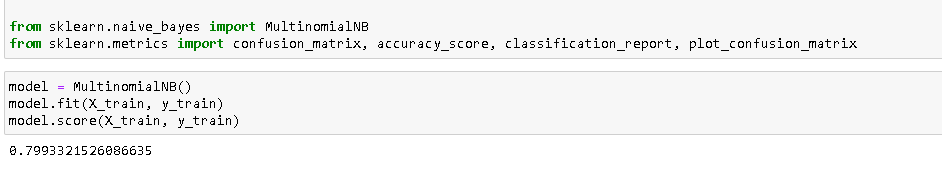
**6.1 Split data into 80% train and 20% test data [1 Marks]**

****

Approach used:

* Use test\_train\_split() to split the data frame into 80% train and 20% test.
* Use stratify to ensure that the splitting is done such that the distribution of target attribute is similar on both test and train set

**6.2 Build a Classification model to predict sentiment [2 Marks]**

****

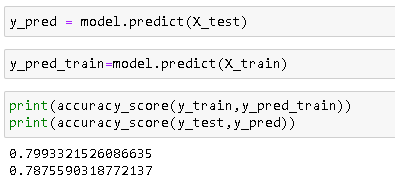
Approach used:

* Use MultinomialNB for classification
* We assign it to ‘model’ and train it on X\_train and y\_train
* Model.score prints the accuracy score on the train data set

Insights:

* The accuracy score of given model is approx 80%

**6.3 Print accuracy score for train and test data both [1 Marks]**

****

Approach used:

* We can directly use model.score(X\_test, y\_test) to get accuracy score
* here , however we are predicting the target attribute on the test set first and then directly print the accuracy score

Insights:

* Accuracy score for both the test and train sets are similar and close to 80%

**6.4 State whether model is underfitting / Overfitting / Good fit [1 Marks]**

* **Model would have been overfitting if accuracy of train was remarkably higher than that of test set**
* **Same is the case with underfitting**
* **Since, in this case, the accuracy score are similar for both test and train sets, we can say that the model is a good fit.**