DATA-602 FINAL PROJECT DOCUMENTATION

ABSTRACT: The project analysis investigates the reviews on mobile phones that are sold on Amazon.com with an objective of finding product and consumer insights based on price, ratings and reviews. We also try to analyze the trends based on the relationship between the three.

The concept of one click shop and the ease of shopping for various products from our homes through various electronic medium's like mobile phones and tablets has really changed the way a person looks at a product and its credibility. The product information and the product's reliability are easily available on various online portals for shopping and this amount of time spent on shopping is a critical factor for the decision-making process.

Amazon.com was chosen for the data because it has a fairly simple and easy method to use review system for thousands of products and the flexibility that website provides to the consumers to give a fair viewpoint of their experiences to their peers and also to the seller and the makers as well. The rating system of amazon.com is reliable as it even gives the peers to rate a review. The rating system is on a scale of 0-5 and the average of all the ratings is calculated for the final product rating.

Introduction to Dataset:

This dataset contains 82,815 reviews from Amazon about cell phones from 2004 until Sep 2019. Each review can be associated with an item and brand name and comes with a rating ranging from 1 to 5. This makes the dataset a perfect sample for text analytics and sentiment classification.

Project Topics:

1.Reading Datasets into JUPYTER Notebook.

2.Understanding Data and Cleaning both datasets.

3.Performing Visualization to understand trends in data.

4.Text Data Cleaning

A. Merging both datasets and cleaning,

B. Removing Numbers from review columns,

C. Remove punctuations,

D. Convert to lower case,

E. Remove Stop Words,

F. Word Tokenization.

5. Vectorization and Topic Modelling

A. Count Vectorizer,

B. TF-iDF Vectorizer,

C. PLSA,

D. LDA,

E. Visualizing topics in LDA.

6. Applying Different Classifiers

A. Multinomial NB,

B. Logistic Regression,

C. SVM,

D. Random Forest.

7. Building Sequential NN using Keras.

8. Conclusion

1.Reading Datasets into JUPYTER Notebook:

Initially I started with importing all the libraries required to perform all mentioned topics and import warnings to ignore depreciation warning. Reading items and reviews datasets in to JUPYTER notebook using pd.read_csv. Later for checking for null values, data types of each column and shape of the datasets is done. Items dataset have 792 rows and 9 columns whereas reviews dataset has 82815 rows and 8 columns.

Column names of both datasets-

```
items.columns
```

```
Index(['asin', 'brand', 'title', 'url', 'image', 'rating', 'reviewUrl',
       'totalReviews', 'prices'],
      dtype='object')
```

```
reviews.columns
```

```
Index(['asin', 'name', 'rating', 'date', 'verified', 'title', 'body',
       'helpfulVotes'],
      dtype='object')
```

2.Understanding Data and Cleaning both datasets:

To understand the dataset, I started with Observing prices column in items data frame which has more inconsistencies with Comma, Dollar sign and length of few elements greater than 8. In order to remove these inconsistencies following syntax is used which removes punctuations, identifies elements whose length is greater than 8 and removes those columns from data frame.

```
items['prices'] = items['prices'].replace({'\$': '', ',': ''}, regex=True)
```

```
index=items['prices'].str.len()
```

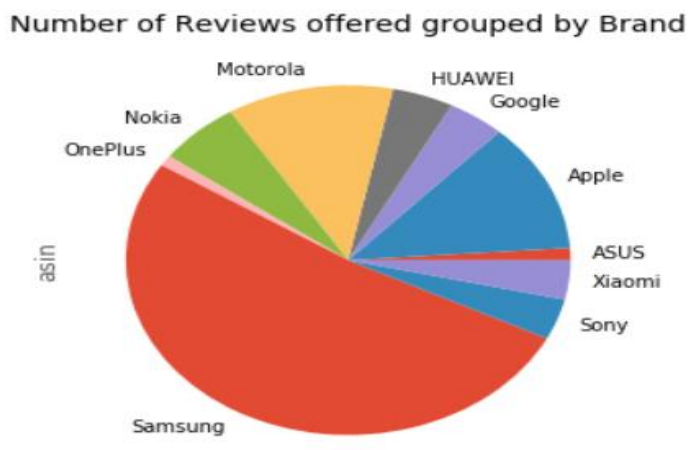
```
list_=[i for i,v in enumerate(index) if v > 8]
```

```
drop_list=pd.DataFrame(items.ix[list_])
```

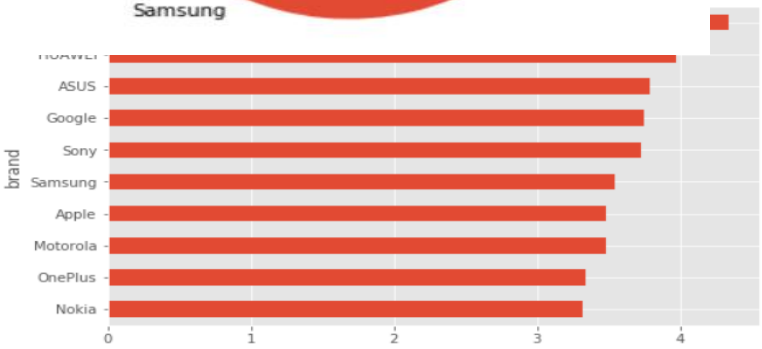
```
cond = items['prices'].isin(drop_list['prices']) == True
items.drop(items[cond].index, inplace = True)
```

Then re-indexing items data frame and null imputing the prices and other numerical columns in items data frame is done. A Column named hopeful votes in reviews dataset have more than 50% of null values, so dropping that column from reviews data frame.

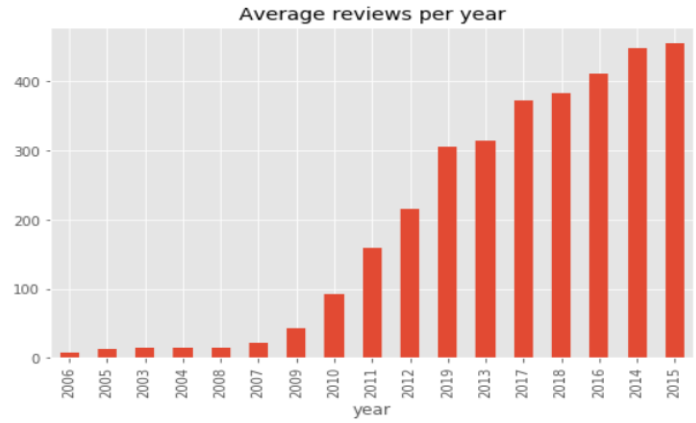
3.Performing Visualization to understand trends in data:



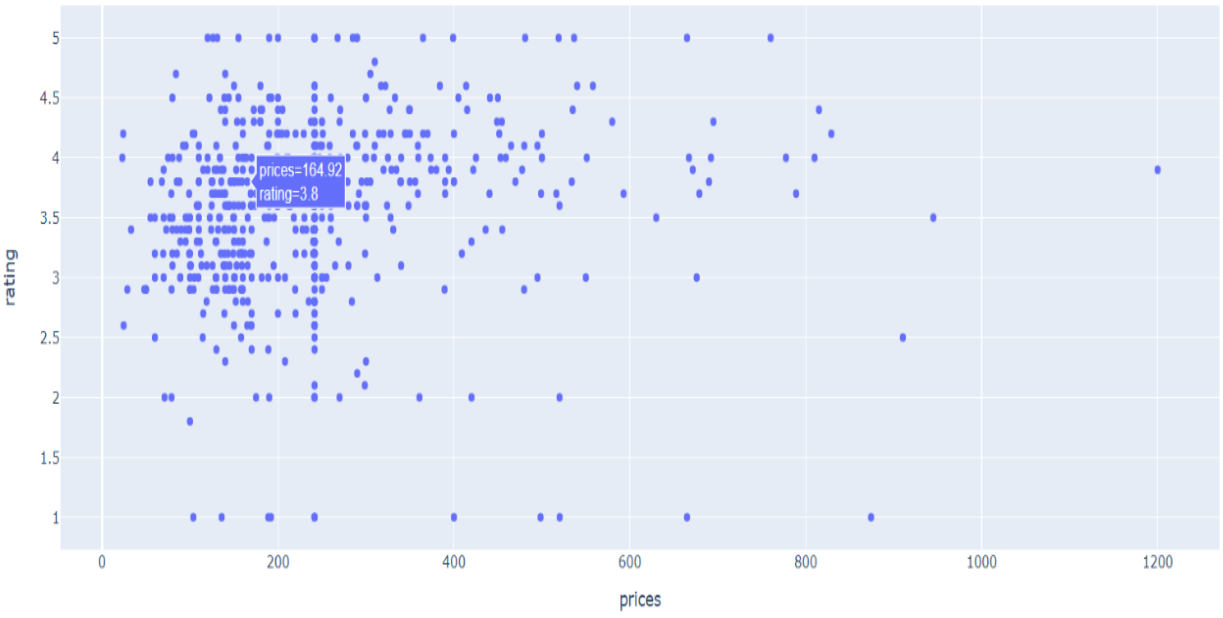
Visualizations help us in understanding the outcomes in a better way with pictorial representations. To generate visualizations groupby function is applied on brand and asin columns of items data frame which reveals that more than 50% of reviews are recorded under Samsung and then followed by Motorola and Apple.



Xiamoi tops the list of average rating per each brand. By applying groupby function on brand and rating in items data frame gives the above plot.



By breaking date column in reviews data frame and creating year and date columns. Taking asin, brand, prices, total reviews, year as columns and creating new dataframe DF then applying groupby on year and total reviews gives the following plot. Where we can observe ratings in 2015 and 2014 are high when compared to recent years.



Products whose price ranging from $50-$300 has highest number of reviews and most of the ratings range from 2.5 to 3.5. Used pyplot library to plot this scatter plot which gives the index of each point on the plot.

4.Text Data Cleaning:

To predict ratings based on title _y and body columns in the reviews data frame, text data in both the columns is cleaned after creating a new data frame called Final _DF by merging Items and reviews data frames with columns asin, rating _x, rating _y, title _y, prices, body, total _reviews. Imputing the null values in the new data frame and working on text columns of title _y and body.

```
Final_DF.title_y[10]
['works', 'great', 'dont', 'dropt']
```

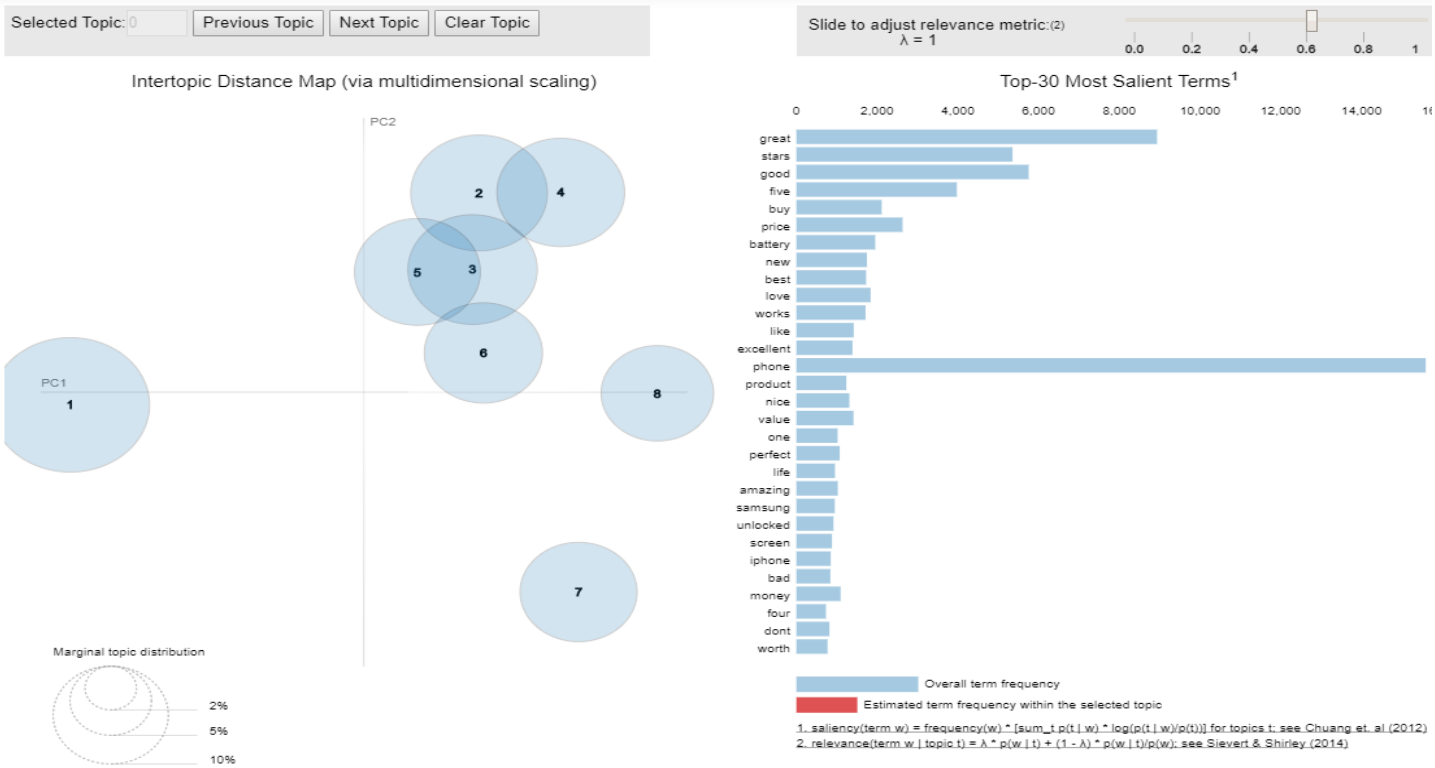
```
Final_DF.body[10]
[['ive',
  'phone',
  'year',
  'really',
  'like',
  'never',
  'partial',
  'flip',
  'phones',
  'appreciate',
  'nokias',
```

From title and body column removing punctuations, converting to lower case, removing stop words, removing numbers and word tokenizing gives you below output.

And now splitting the new dataset into training and testing datasets with title _y and ratings _y as columns.

5. Vectorization and Topic Modelling:

Applying Count vectorizer and Tf-iDF vectorizer techniques to give the metric value to every word in each document present in the column title _y. To classify each word in the text to a topic we are using LDA and PLSA topic modelling techniques after performing vectorization techniques. Up on performing PLSA on the Tfidf vectorized target variable I have created 30 topics. Calculating perplexity and score of LDA model resulted in: score= -1096800.05 and perplexity =317.71988Used gensim.models.LdaMulticore is used to build a lda model and tried visualization to view the topics-keywords distribution. A good topic model will have non-



overlapping, big sized blobs for each topic. This seems to be the case here. So, we are good.

6. Applying Different Classifiers:

Classifier	Accuracy
Multinomial NB	0.6960198103521169
Logistic Regression	0.7054418070906565
SVM	70.7374524370357
Random Forest	70.29655130760403

SVM gives the better prediction out of all the classifier models, I used to predict the ratings using review column.

7. Building Sequential NN using Keras:

I built a sequential neural network using keras tools with con1D, dense and other convolutional layers. This neural network model makes the best predictions of all the models. It's accuracy is around 85% for the 100 epochs.

```
66228/66228 [==============================] - 239s 4ms/step - loss: 0.6877 - accuracy: 0.7639 - val_loss: 1.8848 - val_accuracy: 0.6600
Epoch 59/100
66228/66228 [==============================] - 243s 4ms/step - loss: 0.6868 - accuracy: 0.7648 - val_loss: 2.0117 - val_accuracy: 0.6632
Epoch 60/100
66228/66228 [==============================] - 244s 4ms/step - loss: 0.6889 - accuracy: 0.7649 - val_loss: 2.0083 - val_accuracy: 0.6661
Epoch 61/100
66228/66228 [==============================] - 245s 4ms/step - loss: 0.6800 - accuracy: 0.7672 - val_loss: 2.2267 - val_accuracy: 0.6656
Epoch 62/100
66228/66228 [==============================] - 486s 7ms/step - loss: 0.6803 - accuracy: 0.7670 - val_loss: 1.9612 - val_accuracy: 0.6639
Epoch 63/100
66228/66228 [==============================] - 289s 4ms/step - loss: 0.6847 - accuracy: 0.7664 - val_loss: 2.2012 - val_accuracy: 0.6626
Epoch 64/100
66228/66228 [==============================] - 238s 4ms/step - loss: 0.6837 - accuracy: 0.7664 - val_loss: 2.0053 - val_accuracy: 0.6622
Epoch 65/100
66228/66228 [==============================] - 235s 4ms/step - loss: 0.6814 - accuracy: 0.7671 - val_loss: 2.1338 - val_accuracy: 0.6673
```