## Experiments

Date: 06-03-2020

### Problem - To create features for our model from the extracted reviews using UI Path.

**Github** - <https://github.com/veeravignesh1/Capstone/blob/master/Iphone7.ipynb>

**Experiment No. 1** – Used NLTK library to create features such as Length of review, Number of sentences, Number of emoticons used, Number of exclamation marks, Number of Question marks etc.

**Findings** – Most of the reviews were of length between 1-70 words. The average length of the reviews was about 52 words per review. The average number of sentences was 4. We found that there were very a small number of reviews that used exclamation or question marks. Also, not many reviewers have used Proper Case in their reviews. The maximum number of words per review that we got after web scraping was 500. This was because there were reviews that had READ MORE section in them which UI Path was not able to handle.

**Experiment No. 2** – Manually copying the reviews that had READ MORE section in them and performing the steps again as done in Experiment No.1.

**Findings** – After manually copying the reviews that had READ MORE section in them, the average length of the reviews came to be 58 words per review. The maximum number of words per review jumped to 633. All the features were extracted in a csv file named ‘withfeatures.csv’.

Date: 13-03-2020

### Problem – Filtering stop words, punctuations and emoticons. Feature engineering using NER tagging, POS tagging, TF-IDF etc. Applying SVR model on the thus created data.

**Github** - <https://github.com/veeravignesh1/Capstone/blob/master/Modelling-Iphone.ipynb>

**Experiment No. 1** – Removing stop words, punctuations, emoticons and spell check. Also, we removed those reviews that had less than 2-3 words in it.

**Findings** – Using the ‘withfeatures.csv’ file that we created in Experiment No. 2 in Problem 1 , we removed the punctuations (question marks, exclamation marks, full stop etc), stop words and emoticons. For spell correction we experimented using autocorrect package of python and textblob package. Since textblob gave us better result we went ahead with textblob.

**Experiment No. 2** – Removed the stop words and applied POS and NER tagging. Used TF-IDF Vectorizer to create a TF-IDF matrix. Created the target variable named ‘h’ which is used to rank the reviews and is calculated as below:

h = Upvote/(Upvote + Downvote)

**Findings** – The TF-IDF matrix thus created consisted of all the unique words found in all the reviews as columns plus the columns we created during feature engineering stage. For iphone7, for a total of 264 reviews, we had a total of 1610 unique words. This dataframe was then exported to a csv file named ‘Unigram\_Feature.csv’.

**Experiment No. 3** – We split the data in ‘Unigram\_Features.csv’ thus created in Experiment No. 2 into Train-Validation-Test. We then used the SVR (Support Vector Regression) model to fit our data.

**Findings** – After fitting the SVR model on training data we predicted the test data and calculated the R^2 score which came out to be just 13%.

Date: 15-03-2020

### Problem – Fitting Random Forest Regressor and XGBoost Regressor models on the data.

**Github** -

1. <https://github.com/veeravignesh1/Capstone/blob/master/Modelling_Iphone_RandomForest.ipynb>
2. <https://github.com/veeravignesh1/Capstone/blob/master/Modelling_Iphone_XGBoost.ipynb>

**Experiment No. 1** – We fitted the Random Forest Regressor model on the Unigram\_Features data and calculated the MAPE score. We also found the most important features by using the feature\_importances\_ function provided by the Random Forest Regressor model.

**Findings** – We found that Upvote and Downvote are the most important features found by our model to predict the outcome. Also, we got a MAPE score of 17.28% means the accuracy of our model was around 82.7%.

**Experiment No. 2** – We fitted the XGBoost Regressor model on the Unigram\_Features data and calculated the MAPE score.

**Findings** – We got a MAPE score of 16.5% means the accuracy of our model was around 83.5%.

Date: 16-03-2020

### Problem – Remove Upvote and Downvote columns and observe the MAPE score.

**Github** ­- <https://github.com/veeravignesh1/Capstone/blob/master/Modelling_Exclude_Upvote_Downvote.ipynb>

**Experiment No. 1** – We removed the Upvote and Downvote columns which our model ranked as the most important features for predicting the outcome and then observed the MAPE score using both Random Forest Regressor and XGBoost Regressor.  
**Findings** – When we drop Upvote and Downvote columns, fit our model on train data and test on test data, we found that MAPE score increased to 29% for Random Forest Regressor and 30% for XGBoost Regessor.

Date: 22-03-2020

### Problem – Getting the review sentiment.

**Github** - <https://github.com/veeravignesh1/Capstone/blob/master/Review_Title_Sentiments.ipynb>

**Experiment No. 1** – We used the Vader Sentiment Analysis for getting the review sentiments such as positive, negative or neutral.

**Findings** – We found that a few of the users who have rated the product very low (less than 2 ratings) have positive sentiments. On checking the reviews of these users manually we found that the reviews were actually positive but the ratings given were less. This can be misleading.

Date: 27-03-2020

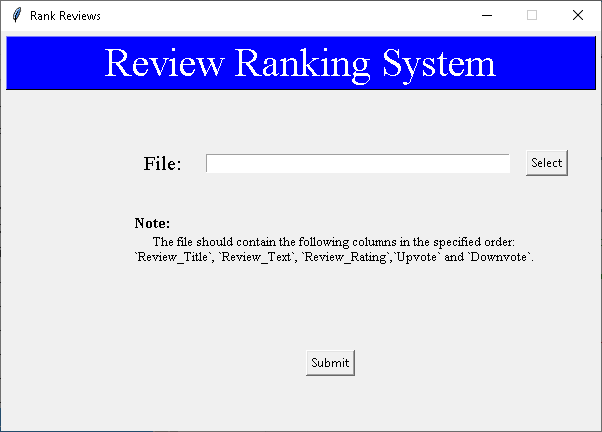
### Problem – Create 1st version of GUI for Review Ranking System.

**Github** –

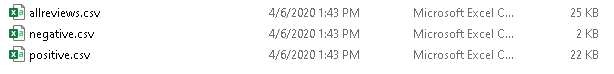
1. <https://github.com/veeravignesh1/Capstone/blob/master/Review_GUI_v1.ipynb>
2. <https://github.com/veeravignesh1/Capstone/blob/master/Review_GUI_v1.py>

**Experiment No. 1 –**

* Used ‘Tkinter’ library to create GUI for our project. It is designed such that it takes the extracted reviews file from the user as input with columns in order of: Review\_Title, Review\_Text, Review\_Rating, Upvote and Downvote. Converted the ipython notebook (.ipynb) to python script (.py)



* This system outputs 3 csv files namely: allreviews, negative and positive each containing reviews which are ranked.



Date: 04-04-2020

### Problem – Add warning message if user clicks the Submit button without selecting any file.

**Experiment No. 1** – Added a warning message that pops up when a user tries to click on Submit button without selecting any file.

**Findings** – The system throwed the warning message even if the file was selected by the user.

**Experiment No. 2** – Tweaked the program to rectify the defect encountered in Experiment No. 1 to give warning message only when file is not selected.

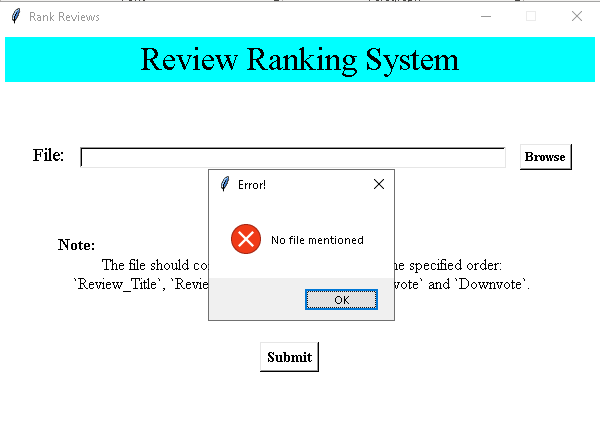
Date: 05-04-2020

### Problem – Change warning message to error message and some UI changes. GUI version 2.

**Github** - <https://github.com/veeravignesh1/Capstone/blob/master/Review_GUI_v2.ipynb>

**Experiment No. 1** – Changed the warning message to an error message when user does not select any file. Changed the look and feel of the UI.

**Findings** – The system showed the error message accurately when user intends to click the submit button without selecting any file.



Date 15-04-2020

### Problem – Web scraping the reviews data using Selenium Webdriver.

**Github** -

**Experiment No. 1** – In order to rank the reviews using just the URL of the reviews page of the product for the new version of the GUI, we scraped the reviews data by using Selenium Webdriver.

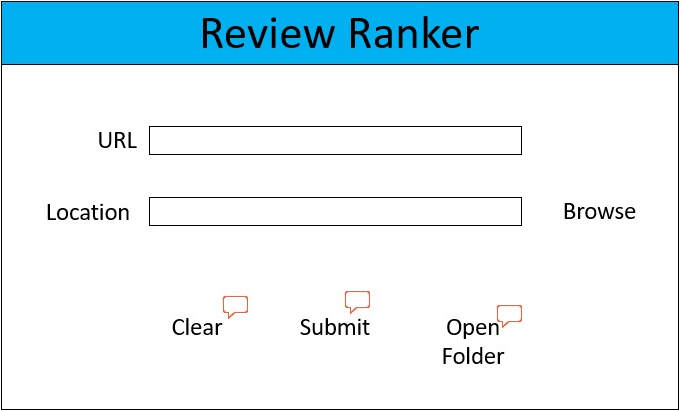
**Findings** – We created a function that would get the reviews just by passing the URL of the reviews page of the product.

Date 16-04-2020

### Problem – To create a GUI to accept the URL of the reviews page of the product and rank them. GUI version 3.

**Github** –

**Experiment No. 1** –Created a GUI that will accept the URL of the reviews page of the product and a folder location where you want to place the files in. On click of Submit button, the ranked reviews (all, positive and negative) will be stored in the specified folder.



Date 18-04-2020

### Problem – To select the best model from Random Forest or XG Boost or SVR to rank the reviews of products in different categories such as Mobile Phones, Laptops and Cameras.

**Github** –

**Experiment No. 1** –Scraped the reviews for a mobile phone, passed it through the get\_features() function containing TF-IDF vectorizer, created different combinations of feature sets, fitted the Random Forest Regressor Model on all the three products and calculated the MAPE score.

**Experiment No. 2** –Scraped the reviews for a mobile phone, passed it through the get\_features() function containing TF-IDF vectorizer, created different combinations of feature sets, fitted the XG Boost Regressor Model on all the three products and calculated the MAPE score.

**Experiment No. 3** –Scraped the reviews for a mobile phone, passed it through the get\_features() function containing TF-IDF vectorizer, created different combinations of feature sets, fitted the Support Vector Machine Regressor Model on all the three products and calculated the MAPE score.

**Findings** – Out of all the three experiments, we found that SVR model gave least MAPE scores, so we went ahead with SVR as our final model.