Machine Learning Engineer Nanodegree

Capstone Project

CLASSIFICATION OF REVIEW RATINGS FOR AMAZON ALEXA

Prabhakar Mudliyar
July 20th, 2019

I. Definition

Project Overview

In the current market, companies are finding various ways to improve the product and increase the sale of their products. The main goal for a lot of companies is to study the customer feedback. Once the sale of product has started, it will all depend on the review of the end users. These review will help the company understand about the performance of the ongoing sale and future sales. Generally, people share their reviews online, mostly on the same website where the product was purchased. The main goal is collect all these reviews and find a way to automate the process of understanding the customer sentiments.

Amazon Alexa is one such product from Amazon for which the customers have shared many reviews and we will try to understand the customer sentiments using one of the Supervised Algorithms.

Why use Supervised Algorithms, as this will help to automate the process of reviewing the large datasets manually and also reduce human errors. And also in time how much we train our algorithm with data it will only increase performance.

Problem Statement

The main objective of this project will be understand the Amazon Alexa product reviews given by different type of people and different places. We will make use of one of Supervised Algorithm called Random Forest Classifier to understand the reviews which are in form of words.

Amazon has different type of Alexa products and they want to know the feedback from the customers for the products. The objective is to perform the analysis of customer sentiments and derive insights into consumer reviews.

The Kaggle data consist of more than 3000 customer reviews, star ratings, review date, variant and feedback of Amazon Alexa products. The objective is to discover insights into consumer reviews and perform sentiment analysis on the data.

Strategy designed to solve this problem are as follows:

- We will import the dataset using Pandas, explore and visualize the data to understand the statistics of what is required.
- Feature Extraction method will be used to solve the problem of review words into order to quantify them
- Then split the data into training and testing set to predict the labels, and evaluate the metric using Confusion Matrix to find out the accuracy.
- Once we have benchmark set for this, we can refine our model to achieve more accuracy.

Metrics

The evaluation metric for this problem will the Confusion Matrix or also called as the Classification Accuracy.

The metrics are percentages of correct predictions. The classification report will also show the results of the precision, recall, f1score. Based on these scores we will be to evaluate the performance of the Algorithm.

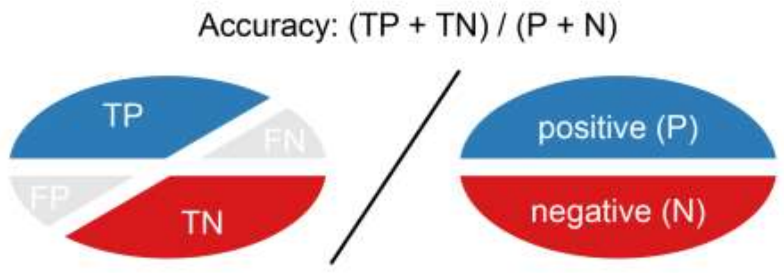
The Confusion Metric that can be used for this problem can be given as below:

A confusion matrix for the model can be shown as below.

Feedback	**Predicted Negative(0)**	**Predicted Positive (1)**
Actual Negative (0)	True Negative	False Positive
Actual Negative (1)	False Negative	True Positive

Accuracy

Accuracy is calculated as no of correct predictions divided by the total no of datasets.



Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

- $\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FN}+\mathrm{FP}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}}$

II. Analysis

Data Exploration

For this problem we will make use of dataset[amazon_alexa.tsv] which can be downloaded from the below link.

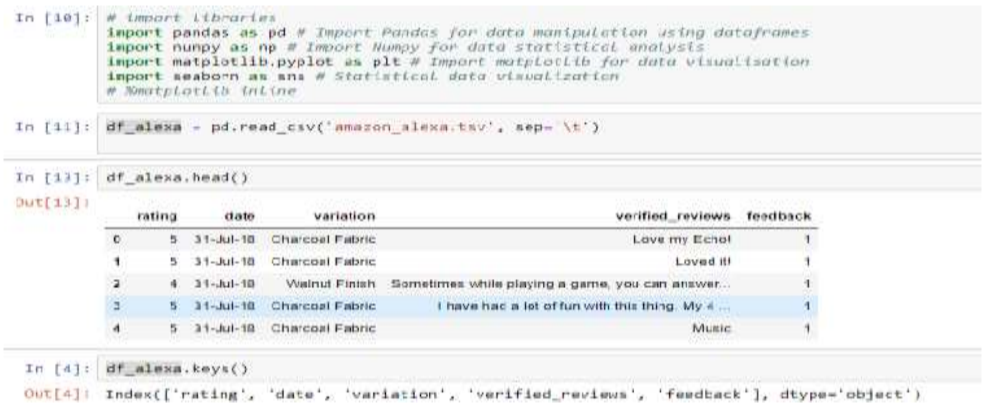
- Dataset: www.kaggle.com/sid321axn/amazon-alexa-reviews

The dataset contains 3150 reviews and 5 features which are explained below:

- Rating: Number ranging from 1 to 5 indicates likeness of the product by the customer.
- Date: On which date the review was recorded.
- Variation: Variants of Alexa Products ex Charcoal Fabric, Walnut Finish etc.
- Verified_reviews: The comment or the review given by the customer for the product.
- Feedback: This is the target variable which help to find the accuracy.

To understand the data, we will explore the datasets by loading the data (amazon_alexa.tsv) in the panda library and analyze the features of the dataset. The steps are given below.

1) Importing the data using the Panda library, we can see below that many other Python libraries are also imported. The below image shows first 5 reviews and also the features.

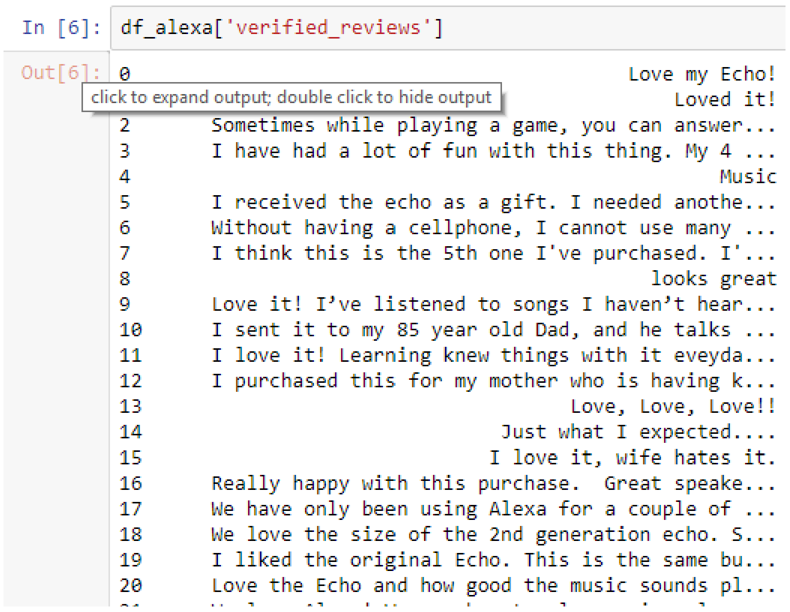


```
In [10]: # import libraries
         import pandas as pd # Import Pandas for data manipulation using dataframes
         import numpy as np # Import Numpy for data statistical analysis
         import matplotlib.pyplot as plt # Import matplotlib for data visualisation
         import seaborn as sns # Statistical data visualization
         # %matplotlib inline

In [11]: df_alexa = pd.read_csv('amazon_alexa.tsv', sep='\t')

In [13]: df_alexa.head()
Out[13]:
```

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1
4	5	31-Jul-18	Charcoal Fabric	Music	1

```
In [4]: df_alexa.keys()
Out[4]: Index(['rating', 'date', 'variation', 'verified_reviews', 'feedback'], dtype='object')
```

2) The below image shows all the reviews given by the customer, as you can see the reviews are not common and are in form of words.

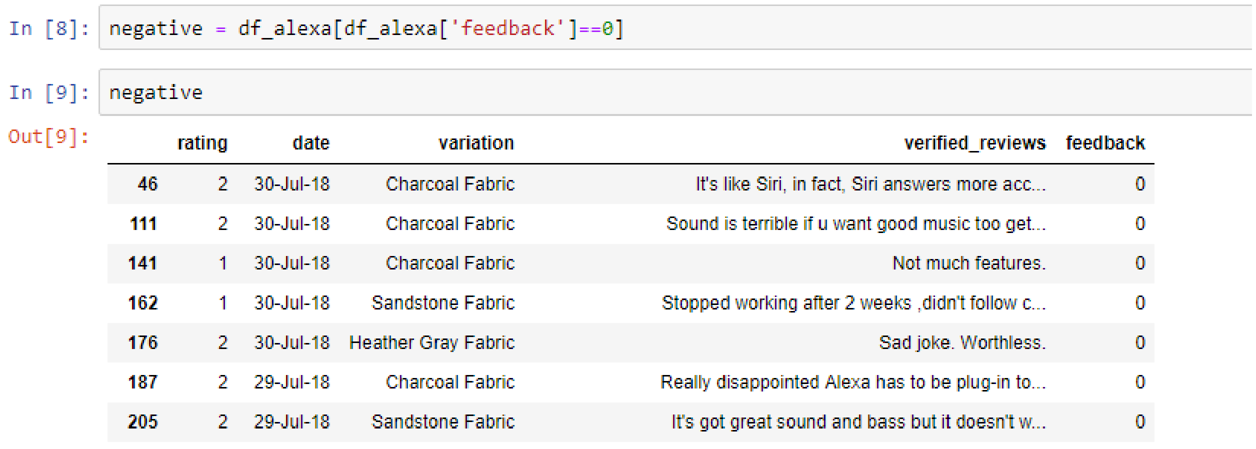


```
In [6]: df_alexa['verified_reviews']

Out[6]: 0                                          Love my Echo!
        click to expand output; double click to hide output   Loved it!
        2       Sometimes while playing a game, you can answer...
        3       I have had a lot of fun with this thing. My 4 ...
        4                                                   Music
        5       I received the echo as a gift. I needed anothe...
        6       Without having a cellphone, I cannot use many ...
        7       I think this is the 5th one I've purchased. I'...
        8                                             looks great
        9       Love it! I've listened to songs I haven't hear...
        10      I sent it to my 85 year old Dad, and he talks ...
        11      I love it! Learning knew things with it eveyda...
        12      I purchased this for my mother who is having k...
        13                                      Love, Love, Love!!
        14                                Just what I expected....
        15                               I love it, wife hates it.
        16      Really happy with this purchase.  Great speake...
        17      We have only been using Alexa for a couple of ...
        18      We love the size of the 2nd generation echo. S...
        19      I liked the original Echo. This is the same bu...
        20      Love the Echo and how good the music sounds pl...
```

Exploratory Visualization

To explore and better understand the data we explore the same by visualization. The visualization will help us understand the statistics of the dataset better.

- The feedback feature from the dataset has only two values i.e 0 or 1. They can also be represented as positive or negative feedback based on the value. The following the image shows all the reviews for which the customer has given negative feedback.

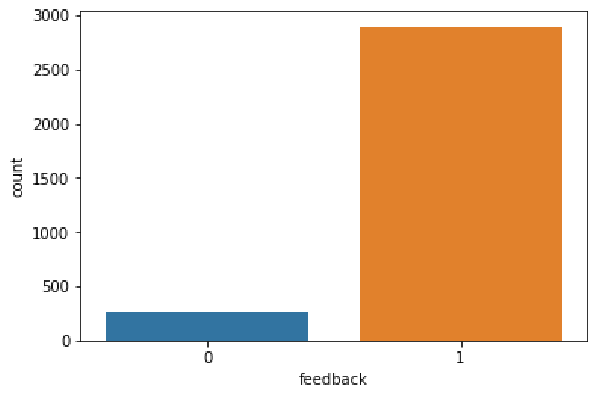


```
In [8]: negative = df_alexa[df_alexa['feedback']==0]

In [9]: negative

Out[9]:
```

	rating	date	variation	verified_reviews	feedback
46	2	30-Jul-18	Charcoal Fabric	It's like Siri, in fact, Siri answers more acc...	0
111	2	30-Jul-18	Charcoal Fabric	Sound is terrible if u want good music too get...	0
141	1	30-Jul-18	Charcoal Fabric	Not much features.	0
162	1	30-Jul-18	Sandstone Fabric	Stopped working after 2 weeks ,didn't follow c...	0
176	2	30-Jul-18	Heather Gray Fabric	Sad joke. Worthless.	0
187	2	29-Jul-18	Charcoal Fabric	Really disappointed Alexa has to be plug-in to...	0
205	2	29-Jul-18	Sandstone Fabric	It's got great sound and bass but it doesn't w...	0

- The below image is the visual representation for the count of positive and negative feedbacks. The visualization clearly shows the Amazon Alexa has more positive feedbacks.

```
In [10]: sns.countplot(df_alexa['feedback'], label = "Count")
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2a25f9e8>
```

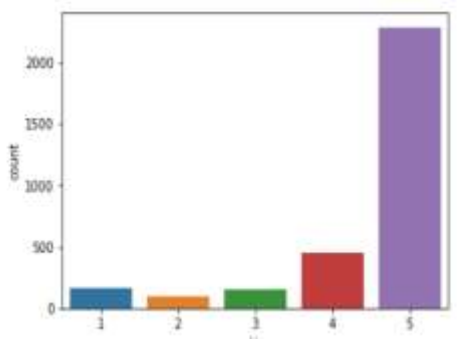


- The percentages for positive and negative feedback can be seen in the below image.

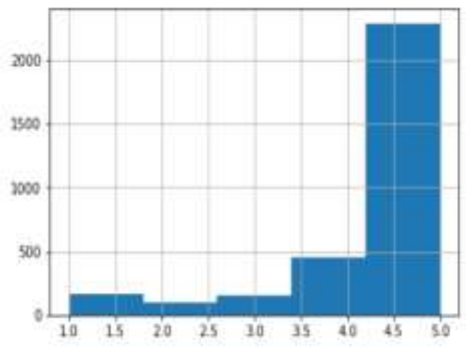
```
In [27]: print("Percentage of negative reviews: ", (len(df_alexa[df_alexa['feedback'] == 0]) * 100)/len(df_alexa))
         print("Percentage of Positive reviews: ", (len(df_alexa[df_alexa['feedback'] == 1]) * 100)/len(df_alexa))

         Percentage of negative reviews:  8.158730158730158
         Percentage of Positive reviews:  91.84126984126983
```

- The below visualization clearly shows that customer has given Alexa a 5 rating. As the count of rating 5 shows higher than the other ratings.

```
In [11]: sns.countplot(x = 'rating', data = df_alexa)
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2a398c50>
```

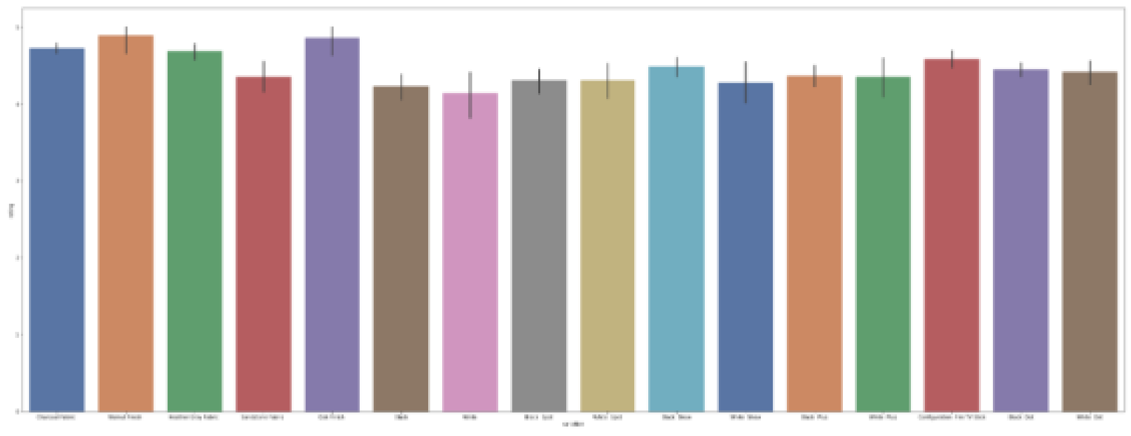


```
In [12]: df_alexa['rating'].hist(bins = 5)
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2a40ac88>
```

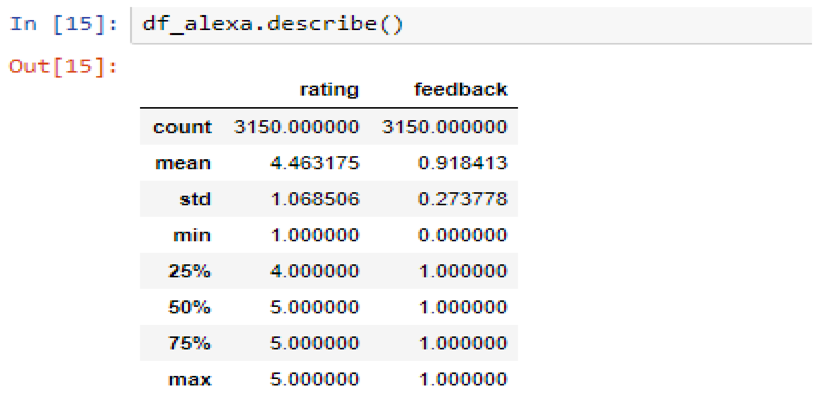


- The below visualization shows the average rating given to all the variants of Alexa.
- **Observations:** It clearly shows that Walnut Finish is the highest average rating among the variants but the count of Walnut Finish is the lowest as shown in the above barchart.

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2a33f2b0>
```

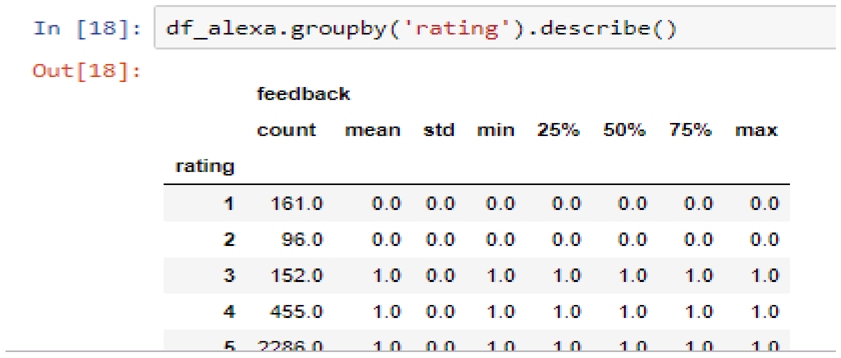


- Descriptive stats for this problem can be seen in the below bar chart.



```
In [15]: df_alexa.describe()
Out[15]:
```

	rating	feedback
count	3150.000000	3150.000000
mean	4.463175	0.918413
std	1.068506	0.273778
min	1.000000	0.000000
25%	4.000000	1.000000
50%	5.000000	1.000000
75%	5.000000	1.000000
max	5.000000	1.000000



```
In [18]: df_alexa.groupby('rating').describe()
Out[18]:
```

	feedback							
	count	mean	std	min	25%	50%	75%	max
rating								
1	161.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	96.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	152.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0
4	455.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0
5	2286.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0

Algorithms and Techniques

To predict the sentiments analysis of the Amazon Alexa products, first will start the with loading and visualizing of different feature to understand our data. After that, we can remove the features which will not helpful in the algorithm to reduce the overhead.

To solve our problem in this project and to find out solution for our problem we will make use of the below Supervised algorithms.

Decision Trees are supervised algorithms where data is split according to a certain condition. DT consist of nodes and leaves where leaves are the decisions or the final outcome whereas the decision nodes is where the data is split based on a certain attribute.

Random Forest Classifier is a type of ensemble algorithms, it creates a set of DT tress from randomly selected subsets of training data and then it combines the votes from the different DT tree to decide final class of test object.

Decision Trees has a limitation that it tends to over fit the data, where the algorithm tries to memorize that data and then predict the target value. To overcome the problem of generalization we will use Random Forest Classifier, which creates of random sets of data to train the algorithm and then find the vote count for the sets and return the result as average.

Before implementing the data, we need to perform some feature extraction and feature encoding for some features. The features like rating and review_date are removed from the dataset as they are not useful.

For feature variant we perform feature encoding where we remove the dummy data which are duplicates and are concatenated to the dataframe with encoded values in the form of number.

The feature review which consist of lots of words need to feature encoded, which can be done using CountVectorizer. CountVectorizer creates a bag of unique words from the reviews and encodes a number to the words.

These bag of words are now concatenated with the features, for which each row will now contain the count of words occurring in the review.

After the step of feature engineering, we will train our model with the Random Forest Classifier. The feedback feature will be removed from the dataframe and will be used as label to predict the values.

After training of data, we test our model using the test data and evaluate our model using the confusion matrix.

Confusion matric consist of scores for Accuracy, Precision, Recall and F1-Score to evaluate the model.

Benchmark

As the Random Forest Classifier is one of the suitable algorithms we will first the train our model using Random Forest Classifier and find out the accuracy.

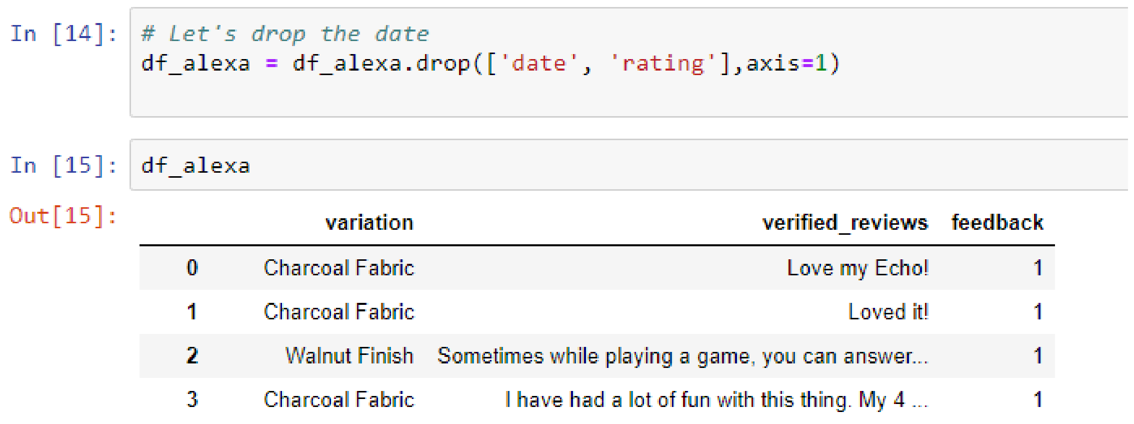
The accuracy will then be used as the benchmark to be crossed while improving the model in the later stages.

III. Methodology

Data Preprocessing

The dataset consists of 5 features which are rating, review_date, review, variant and the feedback. The review feature contains lot of vocabulary words used to describe customer sentiments towards different Alexa variants.

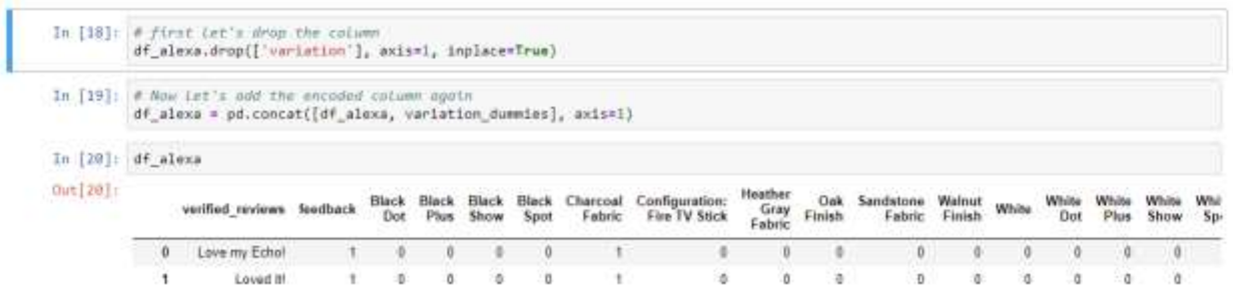
- Features such as rating and review_date cannot be used to predict the customer sentiments we remove them from the datasets.



```
In [14]: # Let's drop the date
         df_alexa = df_alexa.drop(['date', 'rating'],axis=1)

In [15]: df_alexa
Out[15]:
```

	variation	verified_reviews	feedback
0	Charcoal Fabric	Love my Echo!	1
1	Charcoal Fabric	Loved it!	1
2	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1

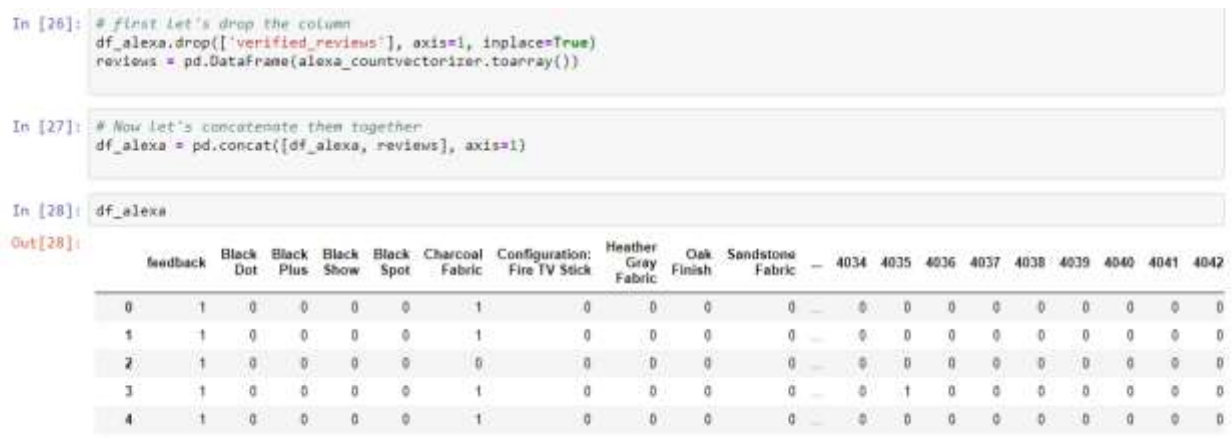
- Feature variant has lot of quality variables we need to quantify them. This can be achieved using Pandas get_dummies function which converts categorical variable into dummy/indicator variables (0 or 1) which makes them a lot easier to quantify and compare. We will use the axis = 1 parameter in the get_dummies function to concatenate the new variables as new features in our original dataframe.



```
In [18]: # first let's drop the column
         df_alexa.drop(['variation'], axis=1, inplace=True)

In [19]: # Now let's add the encoded column again
         df_alexa = pd.concat([df_alexa, variation_dummies], axis=1)

In [20]: df_alexa
Out[20]:
```

	verified_reviews	feedback	Black Dot	Black Plus	Black Show	Black Spot	Charcoal Fabric	Configuration: Fire TV Stick	Heather Gray Fabric	Oak Finish	Sandstone Fabric	Walnut Finish	White	White Dot	White Plus	White Show	Whi Sp
0	Love my Echo!	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
1	Loved it!	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	

- Review Feature has lot of words which are quality variables but we need to encode them to make them quantifiable. For this we will use the Bag of Words model, which can be implemented using feature extraction function CountVectorizer. The CountVectorizer encodes each unique words in the review feature to a number which can then be easily quantifiable for analyzing the reviews. The steps performed to achieve these is given below.



```
In [26]: # first let's drop the column
         df_alexa.drop(['verified_reviews'], axis=1, inplace=True)
         reviews = pd.DataFrame(alexa_countvectorizer.toarray())

In [27]: # Now let's concatenate them together
         df_alexa = pd.concat([df_alexa, reviews], axis=1)

In [28]: df_alexa
Out[28]:
```

	feedback	Black Dot	Black Plus	Black Show	Black Spot	Charcoal Fabric	Configuration: Fire TV Stick	Heather Gray Fabric	Oak Finish	Sandstone Fabric	...	4034	4035	4036	4037	4038	4039	4040	4041	4042
0	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0	0	...	0	1	0	0	0	0	0	0	0
4	1	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0

Now the data is completely processed to be trained.

Implementation

1. **Splitting Dataset**

- The previous step gave us a complete processed data which is quantifiable in nature which can be properly trained. To train our model with the dataset, we need to first split the data into training set and test set.

- The train_test_split function from the model_selection module from the sklearn library will be used to split the data. The function has parameters X(features), y (feedback), test_size (this indicates the size of test data to split from the original data), random_state).

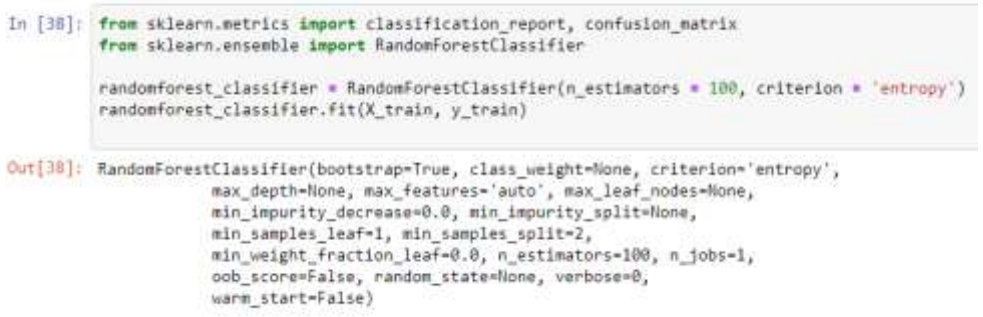
- The following image shows the steps of splitting the dataset.

```
In [33]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state=5)

In [34]: X_train.shape

Out[34]: (2520, 4059)

In [35]: X_test.shape

Out[35]: (630, 4059)

In [36]: y_train.shape

Out[36]: (2520,)

In [37]: y_test.shape

Out[37]: (630,)
```

- The X_train consists of features that needs to be trained and y_train consist of feedback feature which will be used to predict the model. X_test consist of features that will be used as training set and y_test will be label to predict the values.

2. Training the model

We will use the training set to implement our model, for this we will be using the RandomForestClassifier Algorithm. The implementation steps are given below.



```
In [38]: from sklearn.metrics import classification_report, confusion_matrix
         from sklearn.ensemble import RandomForestClassifier

         randomforest_classifier = RandomForestClassifier(n_estimators = 100, criterion = 'entropy')
         randomforest_classifier.fit(X_train, y_train)

Out[38]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)
```

For RandomForestClassifier function we will pass the following parameters

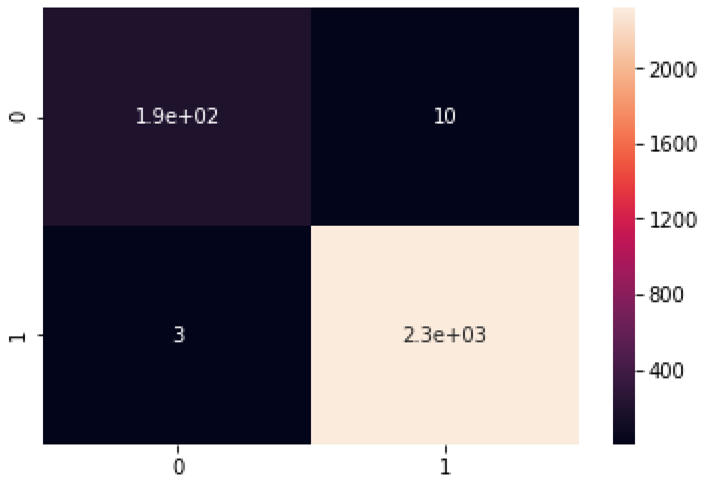
n_estimators is defined as the number of trees and entropy criteria will be used to train our model

3. Evaluating Model

To evaluate the model, we will use the confusion matrix to find the accuracy of our model which trained with training dataset.

Confusion matrix is a table that is used to describe the performance of classifier model on a set of test data for which the truth values are known. Detailed explanation will be given in the Results section.

```
In [39]: y_predict_train = randomforest_classifier.predict(X_train)
         cm = confusion_matrix(y_train, y_predict_train)

In [40]: sns.heatmap(cm, annot=True)

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2b9f8358>
```



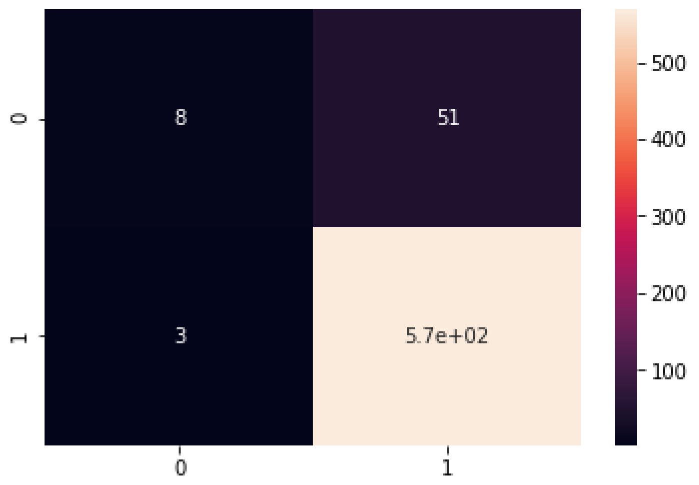
```
In [41]: print(classification_report(y_train, y_predict_train))

                      precision    recall  f1-score   support

                   0       0.98      0.95      0.97       198
                   1       1.00      1.00      1.00      2322
```

The randomforest_classifier.predict(X_train) function is used to predict the label for training set. Then the X_predict_train data is then passed to confustion_matrix function along with training label y_train. The image above clearly shows the model evaluates accuracy for the training model to 1.

Now the model is trained, it can be tested with our test dataset which can be seen in the below image:

```
In [42]: y_predict = randomforest_classifier.predict(X_test)
         cm = confusion_matrix(y_test, y_predict)
```

```
In [43]: sns.heatmap(cm, annot=True)

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2ba65e10>
```



```
In [44]: print(classification_report(y_test, y_predict))

              precision    recall  f1-score   support

           0       0.73      0.14      0.23        59
           1       0.92      0.99      0.95       571
```

Now the randomforest_classifier.predict function is passed the test data and is predicted using feedback label. Classification report shows the accuracy for the test data set.

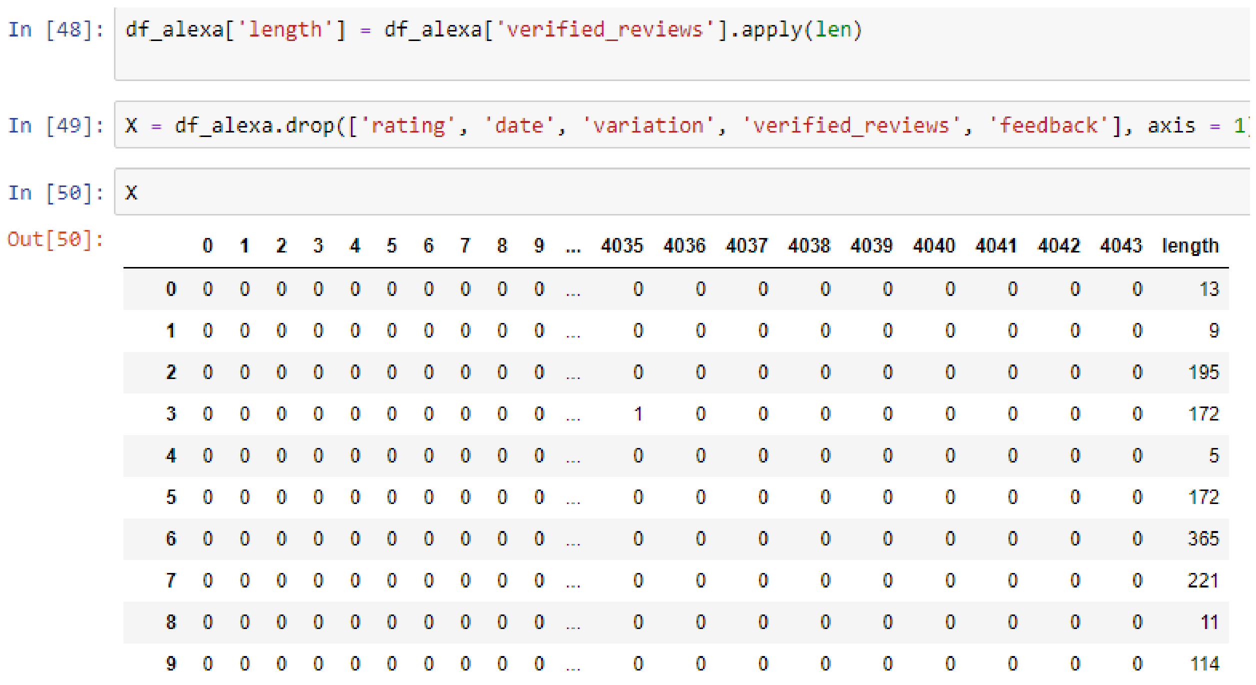
Accuracy for the model = 0.92

The above accuracy will now be considered as the benchmark. In the next step we will try to refine our model to achieve more accuracy than the benchmark.

Refinement

From the previous step, we trained our model to evaluate the accuracy, now in this will improve the model by following the below steps.

The length of the review feature from dataset can now be used as an additional feature to evaluate the sentiment analysis.



```
In [48]: df_alexa['length'] = df_alexa['verified_reviews'].apply(len)
```

```
In [49]: X = df_alexa.drop(['rating', 'date', 'variation', 'verified_reviews', 'feedback'], axis = 1
```

```
In [50]: X
```

Out[50]:

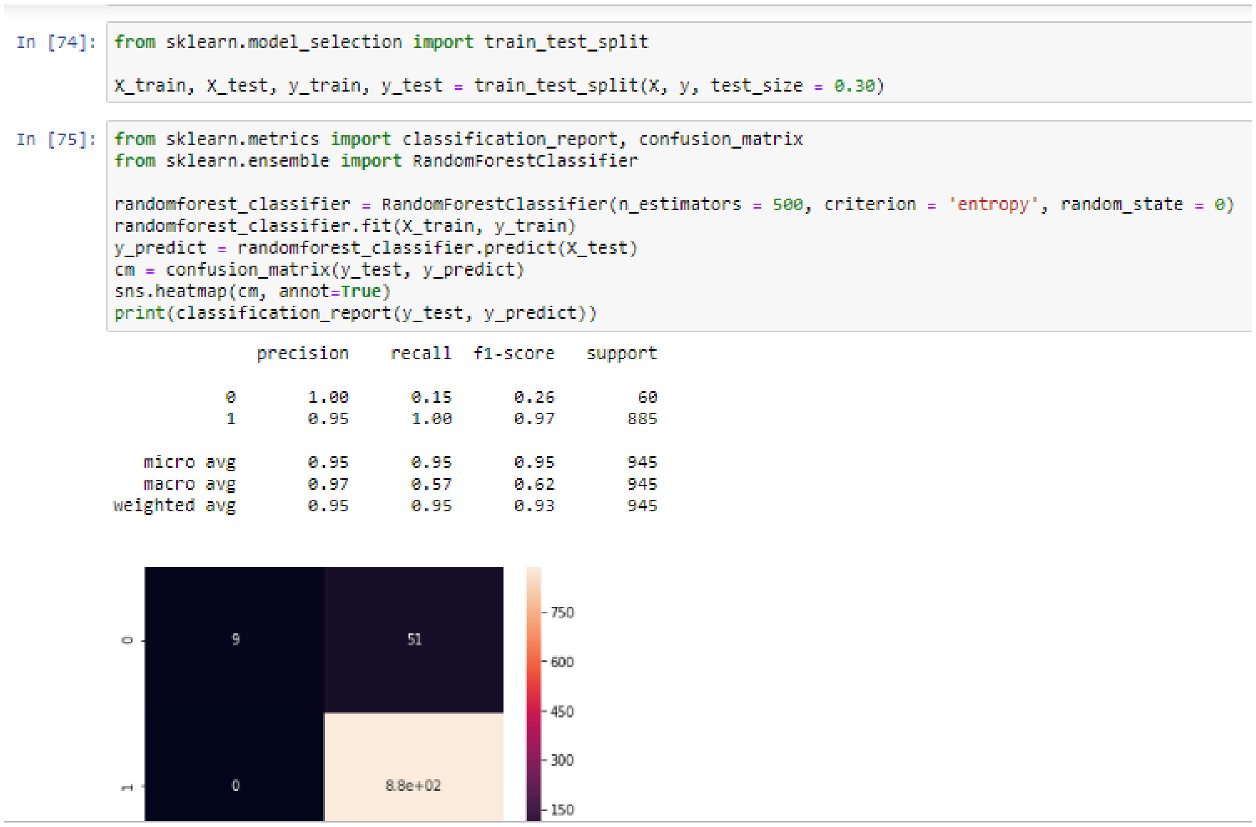
	0	1	2	3	4	5	6	7	8	9	...	4035	4036	4037	4038	4039	4040	4041	4042	4043	length
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	13
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	9
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	195
3	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	172
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	5
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	172
6	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	365
7	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	221
8	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	11
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	114

As shown above, an additional feature 'length' has been added to the dataset. Now we use the new dataset to train our model with the training set and then predict labels for it. Once the model is trained then we used this to predict labels for our test dataset. Once done we can evaluate the accuracy for the refined model using the confusion matrix. The steps performed are shown below.

Parameters Tuning

After including the review length feature to the dataset, we can update our parameters to obtain the highest accuracy for the model and also to overcome the problem of generalization.

- Dataset Splitting: test_size of the dataset was updated to 0.30.
- Estimators: n_estimators for the RandomForestClassifier was changed to 500 from 300 so that we can have more random sample sets for training our model.



```
In [74]: from sklearn.model_selection import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

```
In [75]: from sklearn.metrics import classification_report, confusion_matrix
         from sklearn.ensemble import RandomForestClassifier

         randomforest_classifier = RandomForestClassifier(n_estimators = 500, criterion = 'entropy', random_state = 0)
         randomforest_classifier.fit(X_train, y_train)
         y_predict = randomforest_classifier.predict(X_test)
         cm = confusion_matrix(y_test, y_predict)
         sns.heatmap(cm, annot=True)
         print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.15      0.26        60
           1       0.95      1.00      0.97       885

   micro avg       0.95      0.95      0.95       945
   macro avg       0.97      0.57      0.62       945
weighted avg       0.95      0.95      0.93       945
```

After tuning the parameters and updating the dataset with new feature we can see the accuracy achieved is much higher than the benchmark model

Accuracy for the refined model = 0.95

IV. Results

Model Evaluation and Validation

Model evaluation is done using the Confusion matrix.

“A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.”[2] source from google.

A confusion matrix for the model can be shown as below.

Feedback	**Predicted 0**	**Predicted 1**
Actual 0	True Negative	False Positive
Actual 1	False Negative	True Positive

The matrix above shows the rows which are Actual 0 and Actual 1 i.e for data the feedback is either actually negative or positive respectively. And the column Predicted 0 and Predicted 1 are the predicted by the model.

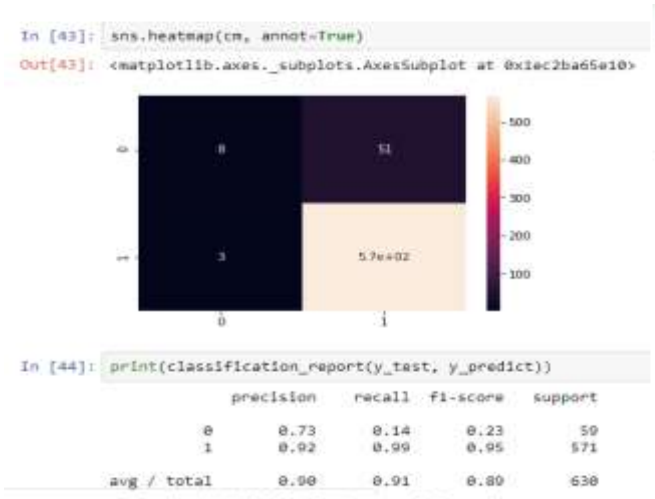
Let's now define our basic terms:

- True Negative: These are cases in which model predicted 0 (Negative feedback) and they are actual 0.
- True Positive: These are cases in which model predicted 1 (positive feedback) and they are actual 1.
- False Positive: These are cases in which model predicted 1 but they are actual 0
- False Negative: These are cased in which model predicted 0 but they are actual 1.

To find the accuracy of the model with respect to the above matrix, the following formula is used.

Accuracy = (True Positive + True Negative) / Total

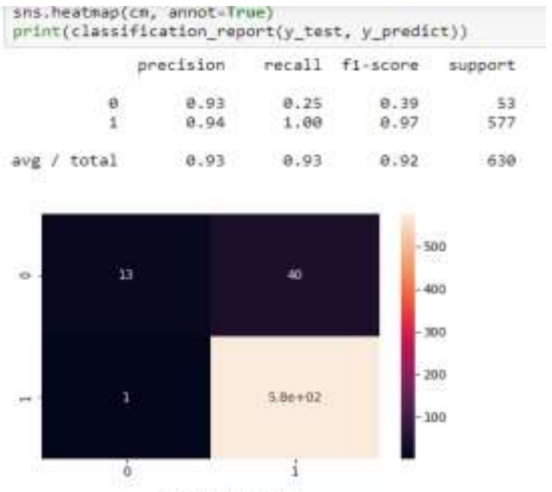
Confusion matrix for our benchmark model and the refined model are given below



```
In [43]: sns.heatmap(cm, annot=True)
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1ec2ba65e10>
```

```
In [44]: print(classification_report(y_test, y_predict))

             precision    recall  f1-score   support

          0       0.73      0.14      0.23        59
          1       0.92      0.99      0.95       571

avg / total       0.90      0.91      0.89       630
```

Benchmark Model (No Review Length Feature)



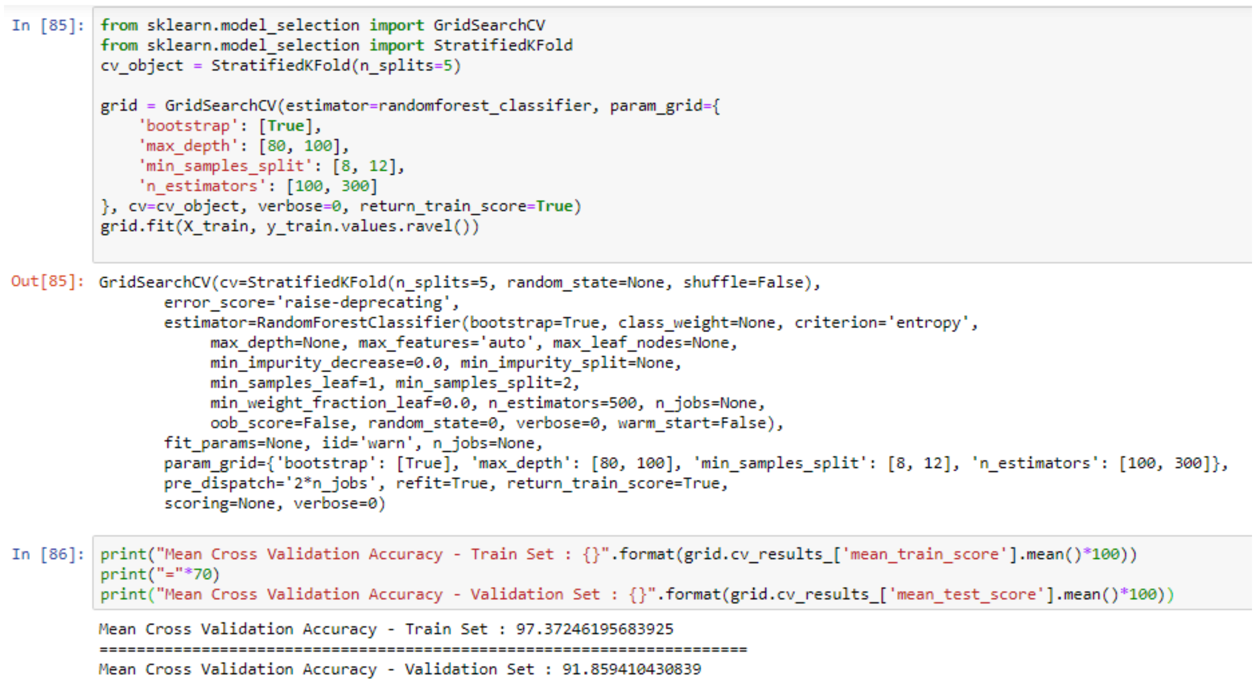
```
sns.heatmap(cm, annot=True)
print(classification_report(y_test, y_predict))

             precision    recall  f1-score   support

          0       0.93      0.25      0.39        53
          1       0.94      1.00      0.97       577

avg / total       0.93      0.93      0.92       630
```

Refinded Model

As shown in the above images the accuracy for the refined model (right image) is greater than the benchmark model(left).

Validation

The below image shows the validation accuracy for our model.



```
In [85]: from sklearn.model_selection import GridSearchCV
         from sklearn.model_selection import StratifiedKFold
         cv_object = StratifiedKFold(n_splits=5)

         grid = GridSearchCV(estimator=randomforest_classifier, param_grid={
             'bootstrap': [True],
             'max_depth': [80, 100],
             'min_samples_split': [8, 12],
             'n_estimators': [100, 300]
         }, cv=cv_object, verbose=0, return_train_score=True)
         grid.fit(X_train, y_train.values.ravel())

Out[85]: GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=None, shuffle=False),
                error_score='raise-deprecating',
                estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
                    oob_score=False, random_state=0, verbose=0, warm_start=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'bootstrap': [True], 'max_depth': [80, 100], 'min_samples_split': [8, 12], 'n_estimators': [100, 300]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
                scoring=None, verbose=0)

In [86]: print("Mean Cross Validation Accuracy - Train Set : {}".format(grid.cv_results_['mean_train_score'].mean()*100))
         print("="*70)
         print("Mean Cross Validation Accuracy - Validation Set : {}".format(grid.cv_results_['mean_test_score'].mean()*100))

         Mean Cross Validation Accuracy - Train Set : 97.37246195683925
         ======================================================================
         Mean Cross Validation Accuracy - Validation Set : 91.859410430839
```

Cross Validation

Normally in a machine learning problem, dataset is divided into training and test sets; the training set is then used to train the model and the test set is used to evaluate the performance of a model. However, this approach may lead to variance problems. In simpler words, a variance problem refers to the scenario where our accuracy obtained on one test is very different to accuracy obtained on another test set using the same algorithm.

The solution to this problem is to use K-Fold Cross-Validation for performance evaluation where K is any number. The process of K-Fold Cross-Validation is straightforward. You divide the data into K folds. Out of the K folds, K-1 sets are used for training while the remaining set is used for testing. The algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training. Finally, the result of the K-Fold Cross-Validation is the average of the results obtained on each set.

The parameters to the CV are as follows:

```
params = {
    'bootstrap': [True],
    'max_depth': [80, 100],
    'min_samples_split': [8, 12],
    'n_estimators': [100, 300]
}
```

The accuracy of the Validation Set is 91.85% which validates our RandomForestClassifier Accuracy.

Justification

Validation set accuracy obtained justifies our model based on the accuracy. The accuracy of our model and validation set is 95% and 91.85% respectively. The difference in the accuracy of the refine model and benchmark model (without parameter tuning) shows a significant increase in the accuracy.

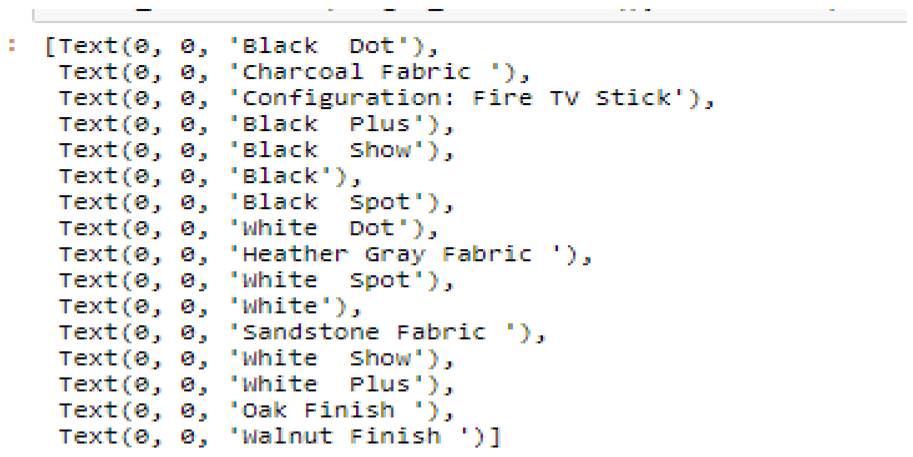
Thus, accuracy obtained using the RandomForestClassifier is significant to say that the model can be used to predict sentiments of the customer reviews.

V. Conclusion

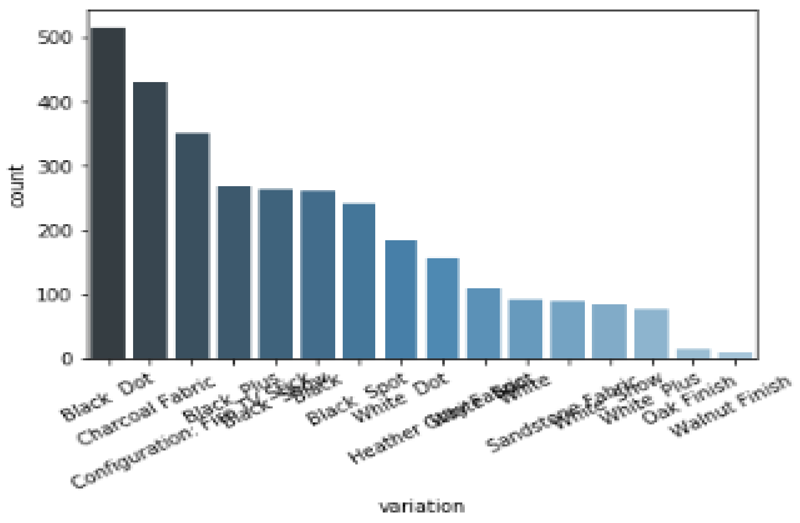
Free-Form Visualization

One aspect which I would like to note is the variation, the average rating for a variation depends on the count of variation.

Here is the count representation for the variations:



```
: [Text(0, 0, 'Black  Dot'),
   Text(0, 0, 'Charcoal Fabric '),
   Text(0, 0, 'Configuration: Fire TV Stick'),
   Text(0, 0, 'Black  Plus'),
   Text(0, 0, 'Black  Show'),
   Text(0, 0, 'Black'),
   Text(0, 0, 'Black  Spot'),
   Text(0, 0, 'White  Dot'),
   Text(0, 0, 'Heather Gray Fabric '),
   Text(0, 0, 'White  Spot'),
   Text(0, 0, 'White'),
   Text(0, 0, 'Sandstone Fabric '),
   Text(0, 0, 'White  Show'),
   Text(0, 0, 'White  Plus'),
   Text(0, 0, 'Oak Finish '),
   Text(0, 0, 'Walnut Finish ')]
```



Now the Walnut Finish Variation has the least count but the average rating is higher for the Walnut Finish. So this may affect the accuracy of the overall model.

Reflection

The steps involved in this model can be stated as:

1. Load the model with dataset using pandas library.
2. Visualize the data and features to understand the data.
3. Preprocess data and perform feature encoding
4. Train the model with train data and predict labels for the training set
5. Test the model with the testing set and evaluate the accuracy using the confusion matrix.
6. Implement a refined model for the above model using addition feature and perform the same steps to get higher accuracy than the previous one.

To reflect back, the feature encoding steps was the hardest for me. But after some research I was able to perform it.

Improvement

In the future, if the scale of the dataset increases or the vocabulary of words in the customer reviews are internationalized, or the words get complex we can do the following steps to improve the model.

1) Implement the model using different ensemble methods for classification like XgBoost classifier.

2) In the Data Pre Processing stage we can make use of Natural Language Toolkit stopwords module to remove stopwords and also the stem module of NLTK to stem words. This will have more efficiency in the results and also the memory size problem can be reduced.

References

[1[Kaggle Amazon Alexa Reviews: https://www.kaggle.com/sid321axn/amazon-alexa-reviews

[2] Confusion Matrix: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[3] Cross Validation: https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/

[4] Evaluation Metrix: https://classeval.wordpress.com/introduction/basic-evaluation-measures/