Capstone Project 2 Milestone Report
Title: Customer Feedback Prediction

Introduction:
The primary goal of any company is to solve customer problems and fulfill their needs. Improving customer satisfaction through review of feedbacks is critical to the success of client business. The focus of this project is to analyze customer reviews and feedbacks of various Amazon Alexa products and to find opportunities for improvement in products and services. The objective is to discover insights and perform sentiment analysis on the data.

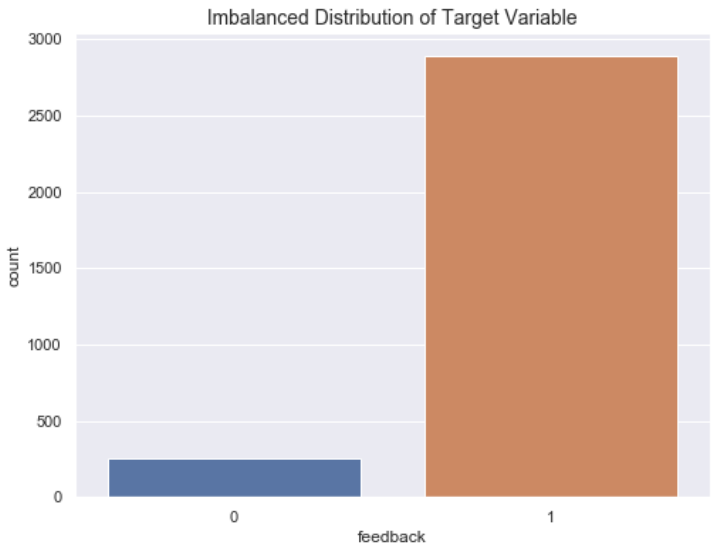
Dataset:
The dataset is taken from Kaggle's Amazon Alexa Review. This dataset consists of nearly 3150 Amazon customer reviews (input text), star ratings, date of review, variant and feedback of various amazon Alexa products like Alexa Echo, Echo dots, Alexa Firesticks etc. for learning how to train Machine for sentiment analysis.
One can use this data to analyze Amazon's Alexa product; discover insights into consumer reviews and assist with machine learning models. One can also train machine models for sentiment analysis and analyze customer positive and negative reviews.

Dataset Description:
The data column fields are described below:

- *Rating*: Ratings given by Alexa users on a scale of 1 to 5.
- *Date*: Date on which rating and review was posted.
- *Variation*: Product variation.
- *Verified Reviews*: The review text.
- *Feedback*: Positive or negative indicator for review. 1 means positive, 0 means negative.

The dataset does not have any null or missing values however, the target variable 'feedback' is quite imbalanced. The number of positive reviews (total 2893) weights significantly higher than the negative class (total 257).

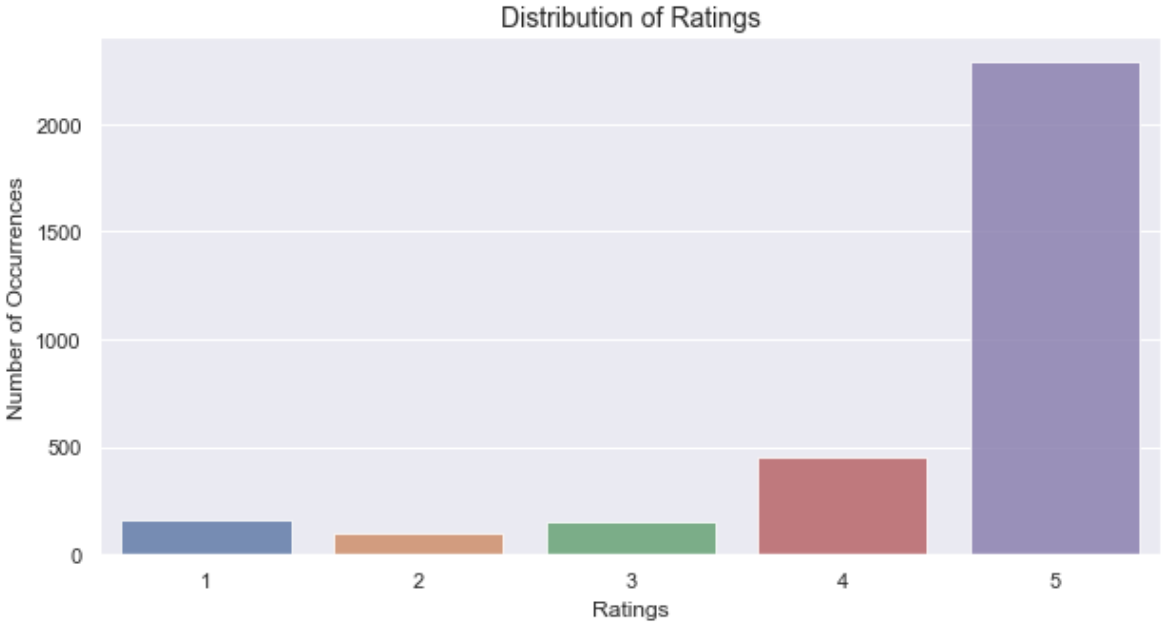
Fig 1.0



Exploring Ratings:

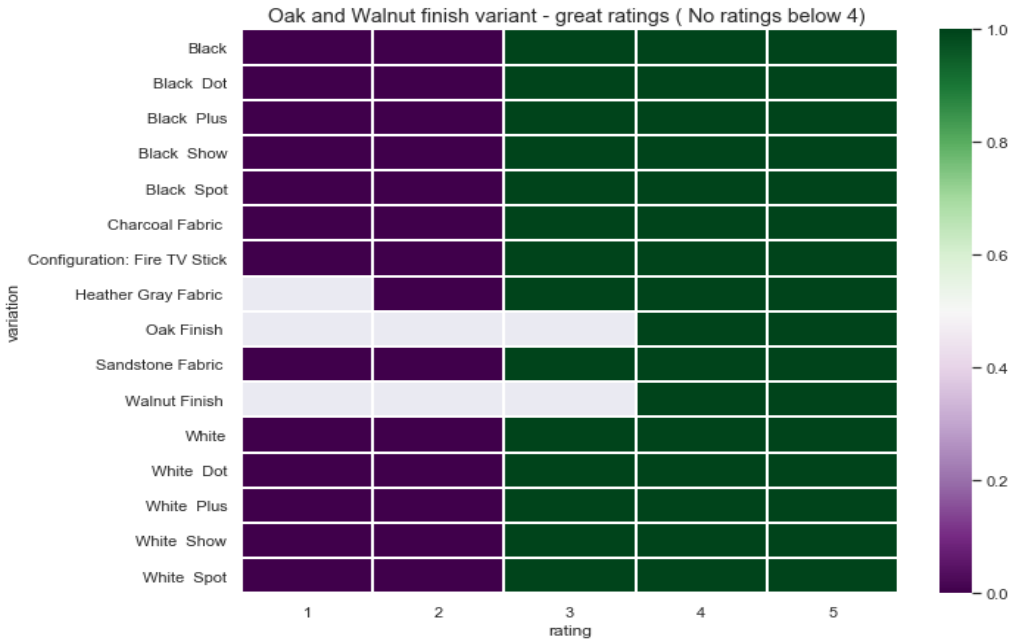
The counts of all 5 types of ratings show that rating 5 is predominant followed by rating 4. Ratings 1 and 2 are comparatively very low. This shows the popularity and high customer satisfaction level for Amazon Alexa product. (fig. 2.0)

Fig. 2.0



A pivot table-based exploration of products variation and ratings (fig. 3.0) show that Heather Gray Fabric never received any rating 1. Oak finish never received rating 1, 2 or 3. Same goes for walnut finish variant.

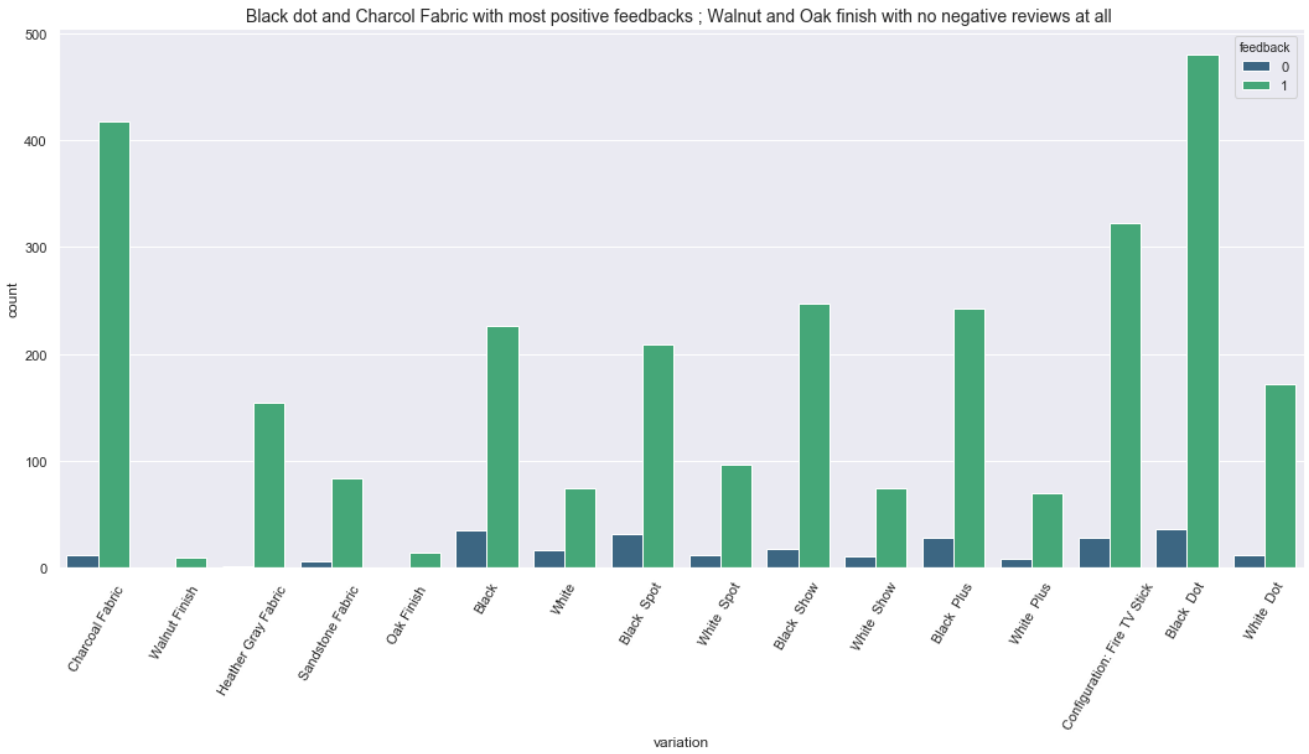
Fig. 3.0



Exploring Variations of Product with respect to Feedbacks:

The variation of feedback i.e. positive and negative was explored with respect to product variation (fig. 4.0). The bar plot shows the ratio of positive and negative reviews for all given variations.

Fig. 4.0



The bar plot shows that Black dot and Charcoal Fabric are having far more positive feedback than negative ones. Walnut and Oak finish have no negative reviews at all.

A violin plot (fig. 5.0) on rating and feedback to show the count distribution of ratings with respect to feedback.

Quite expected that feedback 0 or negative feedbacks are centered around rating 1 and 2 whereas positive feedback is centered around rating 3, 4 and 5.
The distribution plot (fig. 5.1) of ratings with respect to feedback confirms the same:

It is clear that the product receives mostly rating 5 followed by rating 3. In negative reviews, rating 1 is dominant followed by rating 2.

Fig. 5.0

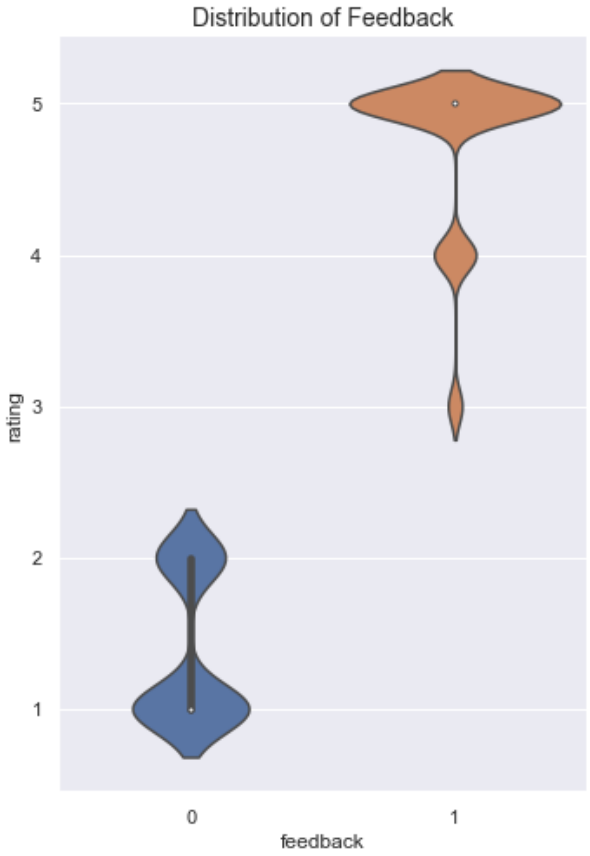


Fig 5.1

Exploring Reviews:
Reviews are written in plain English. These need to be cleared first.
Reviews are cleaned in following ways:

- Short form like won't, can't, 's, 'nt, 'll, 'm etc. are corrected first.
- Carriage returns, new lines, quotes are removed to make the text as a single string.
- Only alphabets are extracted from the text.
- Word are made lower case
- Stop words are removed.

Review Length: Number of Words
One of the important features that can be extracted easily from the raw text data is the number of words in each review. Serious reviews or useful reviews are expected to be having more words. It is found that on average, number of words are more in negative reviews than the positive reviews (fig. 6.0).

Average Word Length
We also extracted another feature which calculates the average word length of each review. This can also potentially help us in improving our model.
Here, we simply take the sum of the length of all the words and divide it by the total length of the review. This can be considered as a normalized length of review.

It is found that on average, number of average word length is almost equal in negative reviews and positive reviews (fig.7.0).

Fig. 6.0

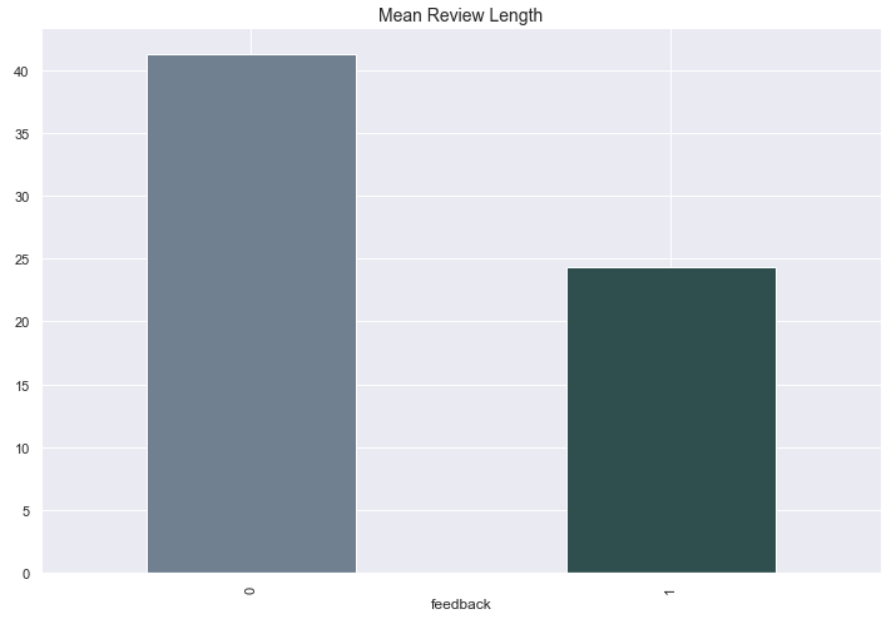
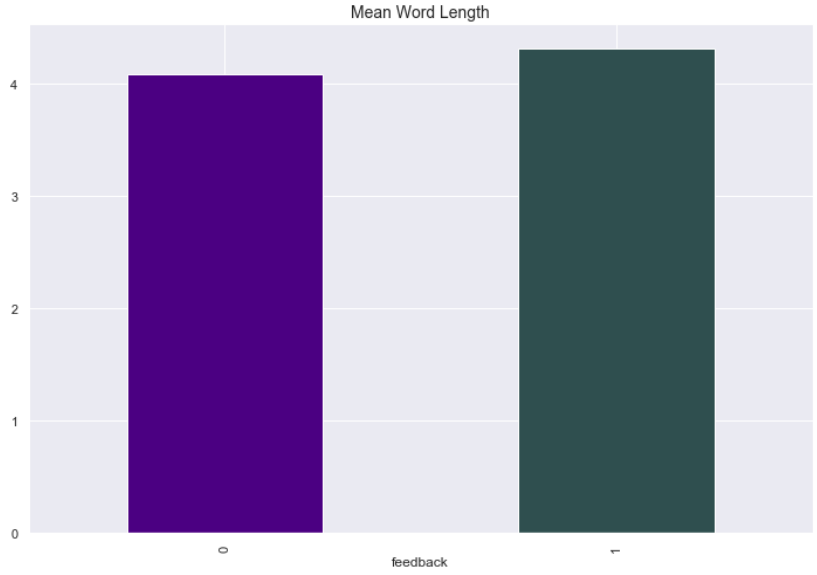


Fig. 7.0



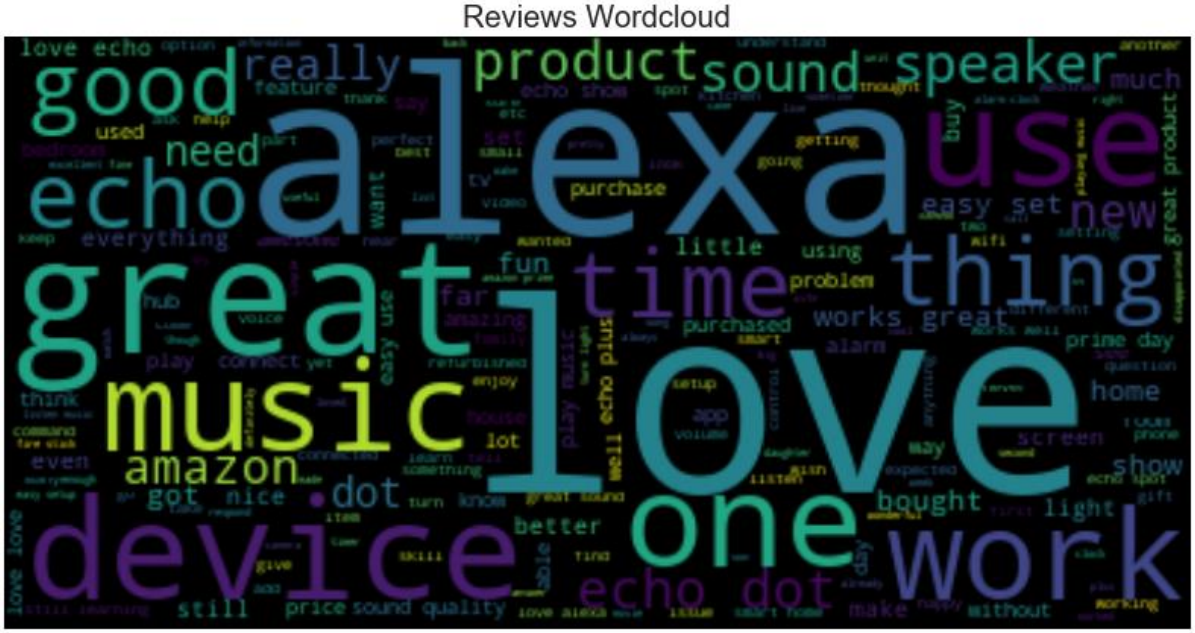
Statistics on length of reviews

The mean for the length of review is found out to be 25.75.
The standard deviation for the length of reviews is 35.10.
The maximum for the length of reviews is 526.

We checked the histogram-based distribution of the length of reviews. Most of the reviews are having review length within 100. Rarely there are reviews having length more than 200.

Which are the most frequent words in the reviews?
We plotted the wordcloud of the reviews and found out the most frequent words (fig. 8.0).

Fig. 8.0



Exploring Variations of Product:
We also checked how different variations of Alexa were sold.
It is evident that Black dot is the top selling product variation followed by Charcoal Fabric. Oak finish and Walnut finish are two least selling variations (fig. 9.0).

We made boxplots (fig. 10.0) of the ratings with respect to different variations.
This is quite biased as most of the reviews are rating 5. Those rating 1 and 2 are considered outliers in each product variation.

Fig. 9.0

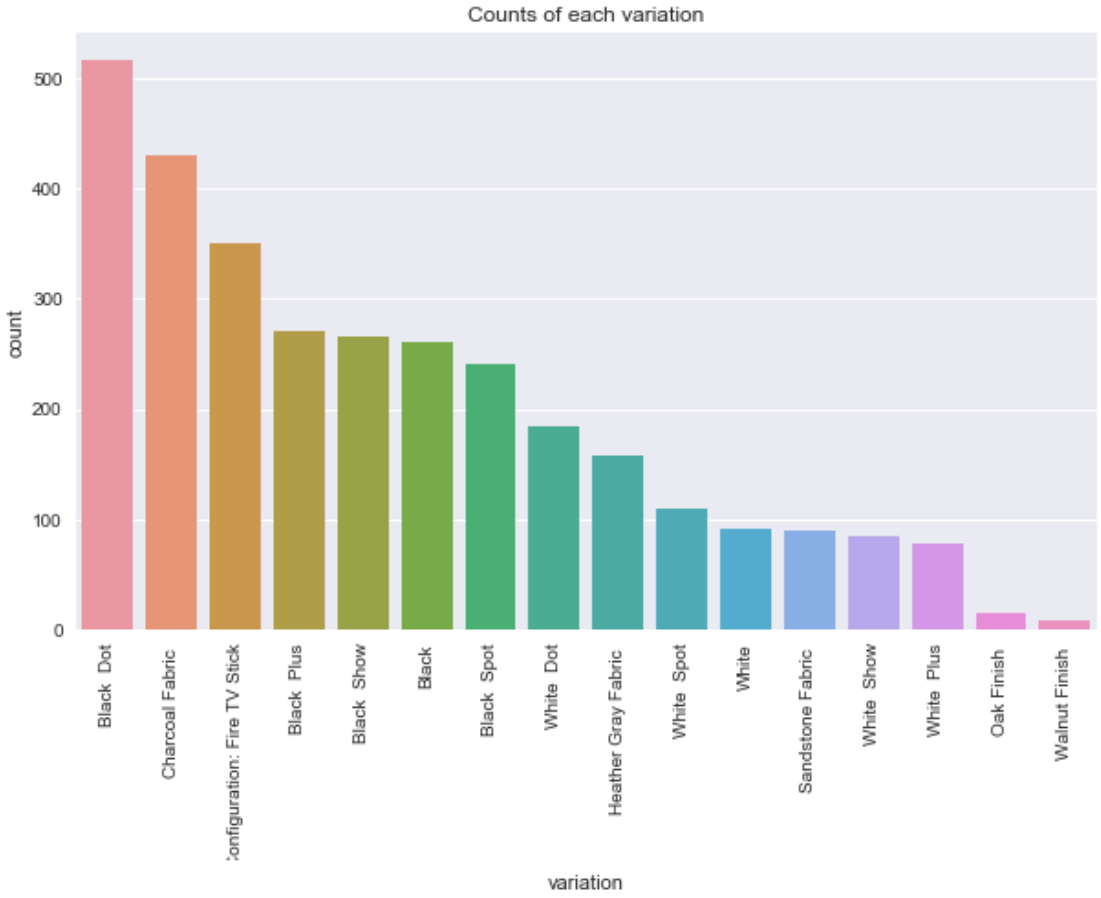
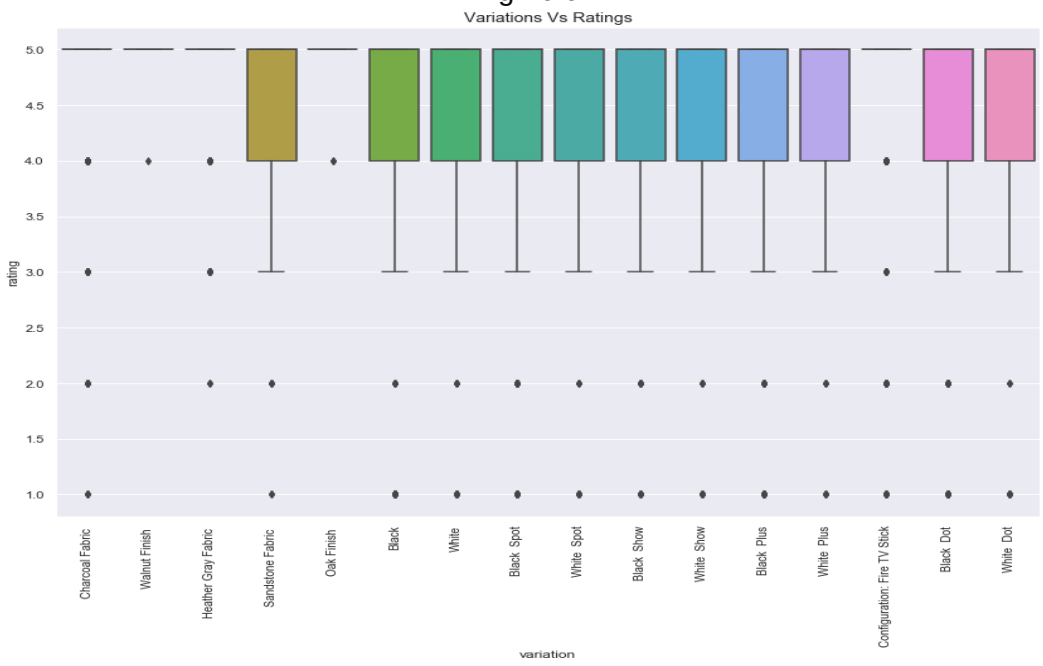
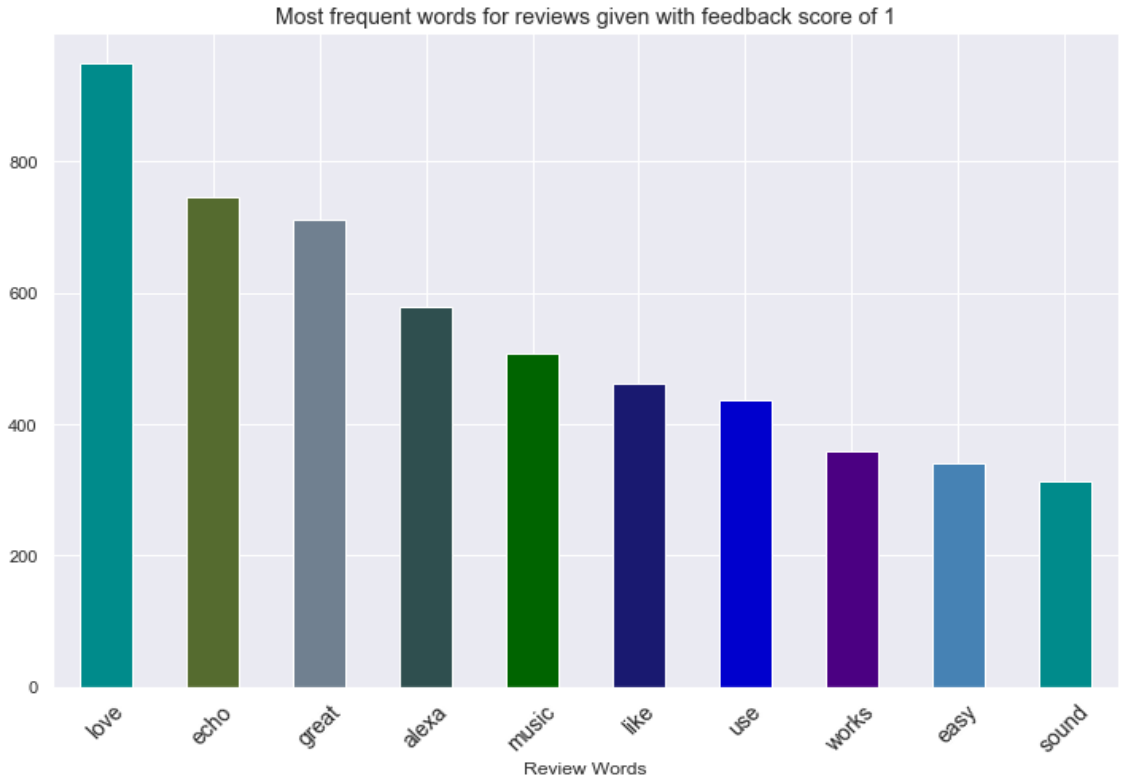


Fig 10.0



We checked the most frequently used words in positive reviews (fig. 11.0):

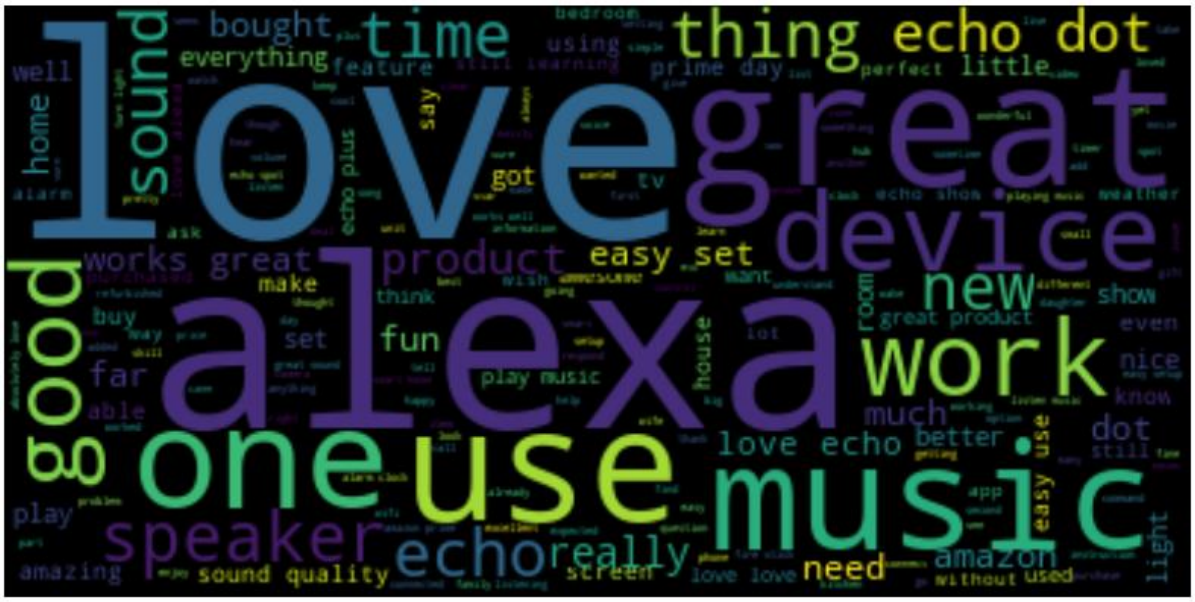
Fig. 11.0



This was also analyzed in wordcloud (fig. 12.0). This is crucial to find out the linguistic nature of the reviews given by users for different types of feedback.

Fig. 12.0

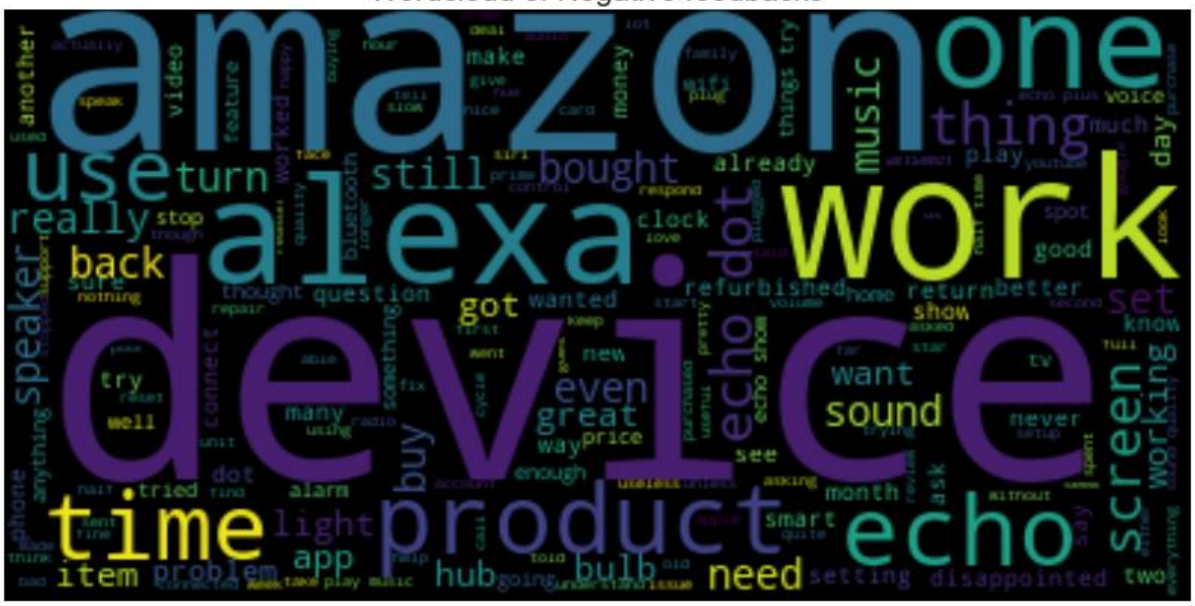
Wordcloud of Positive feedbacks



Similarly, most frequent words for negative reviews are also found out (fig. 13.0).

Fig. 13.0

Wordcloud of Negative feedbacks



Machine Learning Prediction:

The dataset column fields are:

- *Rating*: Ratings given by Alexa users on a scale of 1 to 5.
- *Date*: Date on which rating and review was posted.
- *Variation*: Product variation.
- *Verified Reviews*: The review text.
- *Feedback*: Positive or negative indicator for review. 1 means positive, 0 means negative.

Here date variable does not contain much information about the target variable since the reviews do not show any specific trend with respect to date. Hence, we need to drop the date variable from the list of predictor variables.

As we checked the value counts of the variations, the black dot variation appears at top with presence of 516 number of times followed by Charcoal Fabric variation having 430 appearances.

Variation is a categorical variable with 16 possible types. We need to encode the categorical variable. But before that, we need to clean the text of the values of this categorical variable.

Preprocessing:

We cleaned the shortened form of the words, treated carriage return, new lines, quotations etc. Finally, we kept only Alphabet values and nothing else. Hence all number and punctuations were removed. We made the text in small letters and removed stop words.

Review is text data and needs to be cleaned first. We preprocessed the review text in the same way as mentioned above. We kept preprocessed versions of variation and review data and dropped the actual columns. Feedback column was marked as target variable.

We split the train and test data in 75%-25% manner.

Text Analysis is a major application field for machine learning algorithms. However, the raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

In order to address this, we need to extract numerical features from text content, by:

- tokenizing strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.

- counting the occurrences of tokens in each document.

- normalizing and weighting with diminishing importance tokens that occur in the majority of samples / documents.

In this scheme, each individual token occurrence frequency (normalized or not) is treated as a feature.

The vector of all the token frequencies for a given document is considered a multivariate sample.

A corpus of documents can thus be represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.

Vectorization is the general process of turning a collection of text documents into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or “Bag of n-grams” representation. Documents are described by word occurrences while completely ignoring the relative position information of the words in the document.

As most documents will typically use a very small subset of the words used in the corpus, the resulting matrix will have many feature values that are zeros (typically more than 99% of them).

In order to be able to store such a matrix in memory but also to speed up algebraic operations matrix / vector, implementations will typically use a sparse representation.

Encoding preprocessed variation:

As variation is categorical variable, we encoded it. The preprocessed version of variation was encoded using count vectorizer. Count vectorizer converts a collection of text documents to a matrix of token counts. It produces a sparse representation of the counts using scipy’s sparse csr_matrix.

Since we did not provide an a-priori dictionary and did not use analyzer for feature selection, the number of features is equal to the vocabulary size found by analyzing the preprocessed variation data.

We put the option binary as True that sets all non-zero counts to 1. This is useful for discrete probabilistic models that model binary events rather than integer counts.

Using training data count vectorizer is used to learn the vocabulary dictionary and to return term-document matrix. Test data was used only to transform documents to document-term matrix by extracting token counts out of raw text documents using the vocabulary fitted or learnt.

Training date has 2362 rows and test data has 788 rows. After encoding variation data, number of columns in vectorized matrix is 18.

Encoding preprocessed reviews:

Review is text data and need to be encoded too before it is used in any Machine Learning model. We used Term Frequency Inverse Document Frequency based vectorizer to encode the text data of reviews.

TF IDF Vectorizer converts a collection of raw documents to a matrix of TF-IDF features. TF–IDF means term-frequency times inverse document-frequency.

We used TF IDF vectorizer to encode review text data with n-gram range 1 to 2, minimum df = 5, L2 normalization. We extracted unigram and bigram tokens from the text. When building the vocabulary ignore terms that have a document frequency strictly lower than the given Minimum df threshold.

Processing numerical values:

We have numerical values such as rating, review length and average word length. We used min max scaling to scale the numerical values. Min max scaling transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range of zero and one.

The relationship between rating and feedback is quite straightforward. If rating is 1 or 2, feedback is negative. If rating is more than 2, feedback is positive. If we keep the rating variable as predictor, model will learn to classify the feedback based on rating and it's very easy. We are interested to know if the model can classify feedback based on text data. Hence, we are not keeping rating as predictor to make the problem more interesting.

We merged the encoded review and variation data, scaled numerical values together to get the final set of training and test data.

ML Models:

Random Forest classifier:

We first used Random Forest classifier. The it is an averaging algorithm based on randomized decision trees. It is perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due

to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

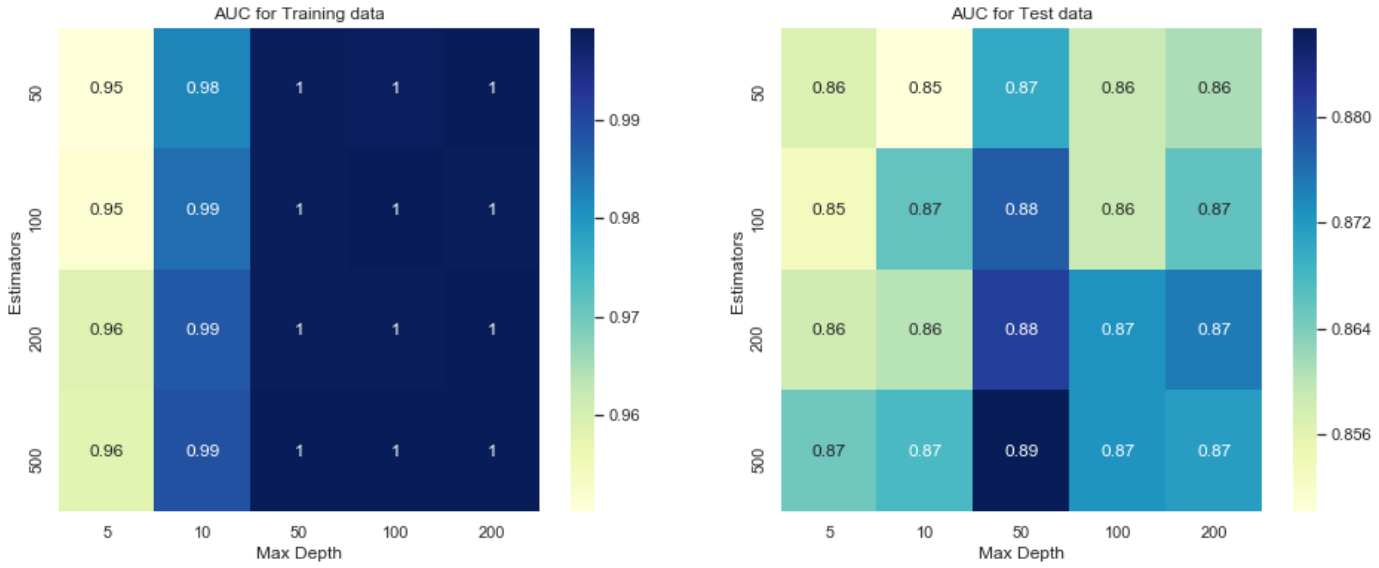
We started with hyper parameter tuning of Random forest. Two crucial parameters – number of estimators and max depth were tuned. Number of estimators is nothing but the number of trees in the forest. Max depth is the maximum depth of the tree. We also made class weight as balanced which is weights associated with classes in the form {class_label: weight}. The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data.

We used cross validated grid search for hyper parameter tuning. Grid Search is an exhaustive search over specified parameter values for an estimator. We searched based on number of estimators as 50, 100, 200, 500 and max depth as 5, 10, 50, 100, 200. 3-fold cross validation was used in grid search. ROC AUC scoring was used to evaluate the predictions on validation sets.

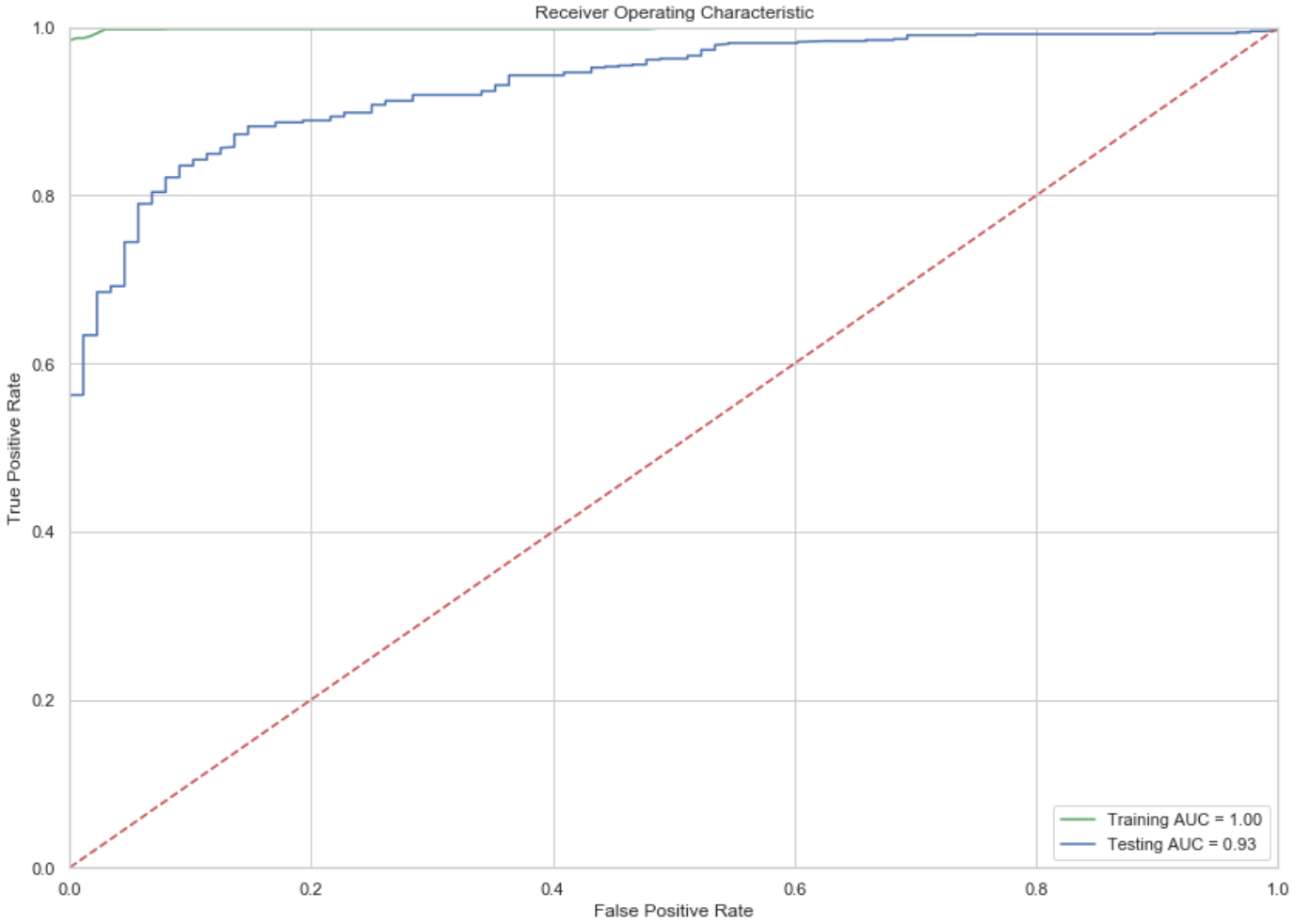
The distribution of ROC AUC values based on hyper parameters values are shown below:

Training Data Set:

Train/Test Data Set:

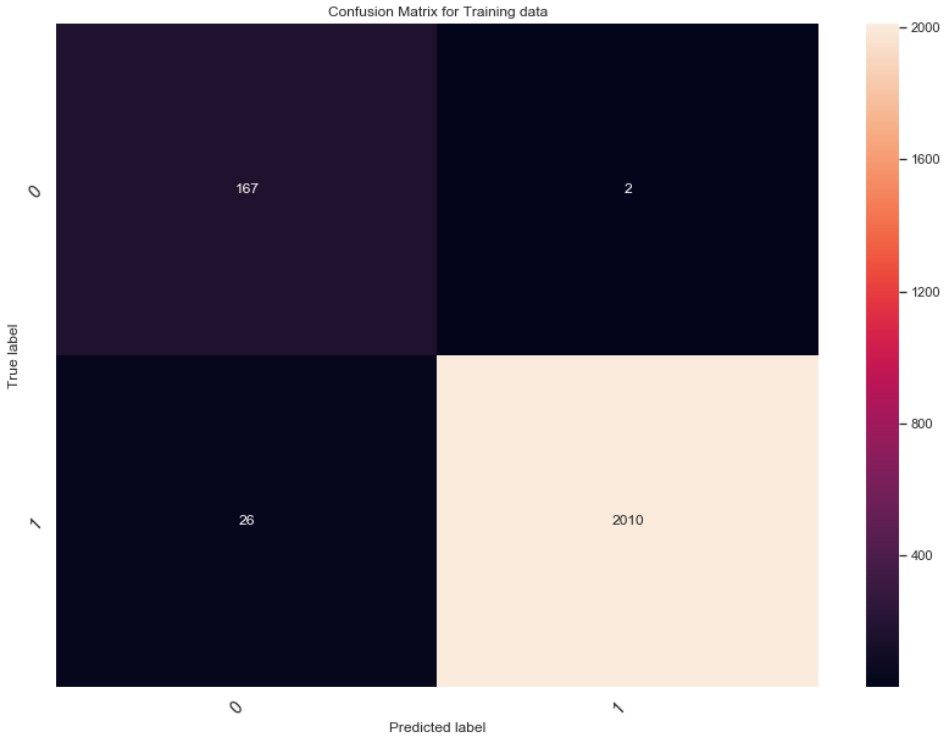


Once the best values of the hyper parameters are extracted, we made a final version of the random forest classifier with the best values. The ROC curve is shown below:

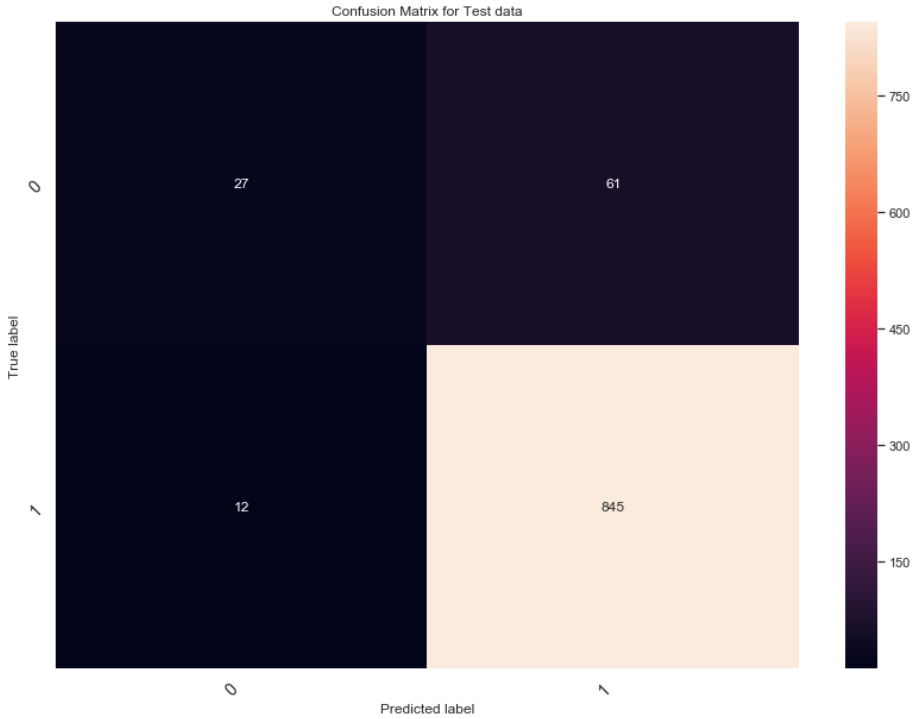


We plotted the confusion matrix:

Training data:



Test Data:



In test set, 93 % accuracy was calculated.

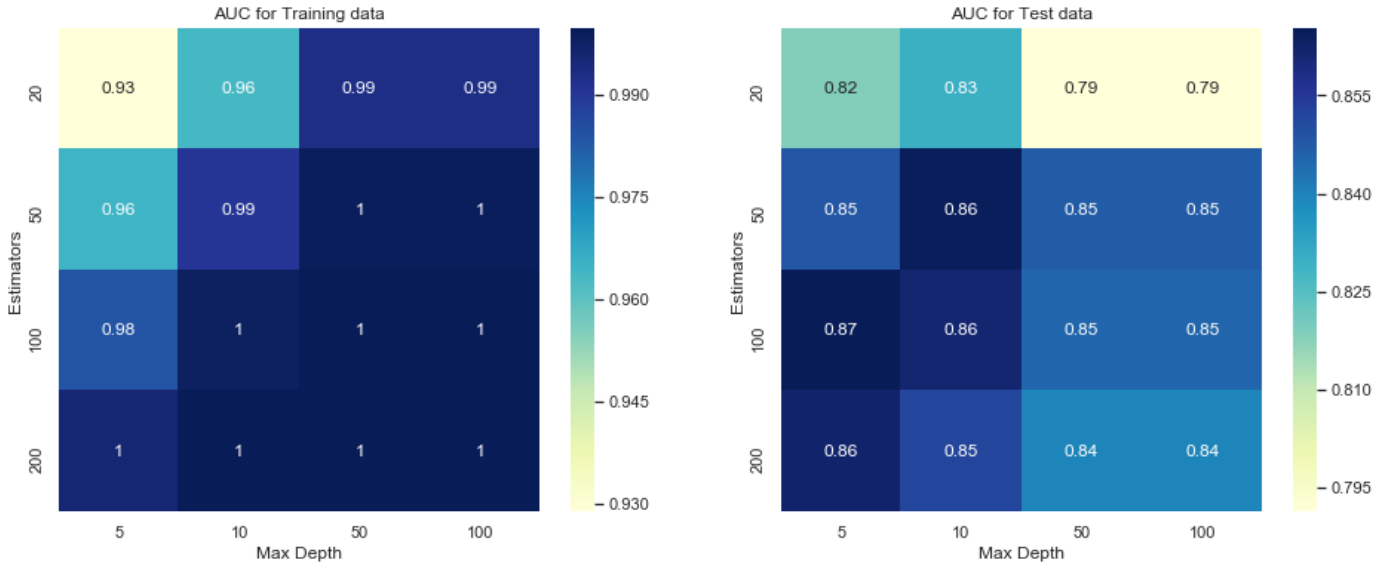
Extreme Gradient Boosting Classifier:

We also tried with Extreme Gradient Boosting (XGB) classifier. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.

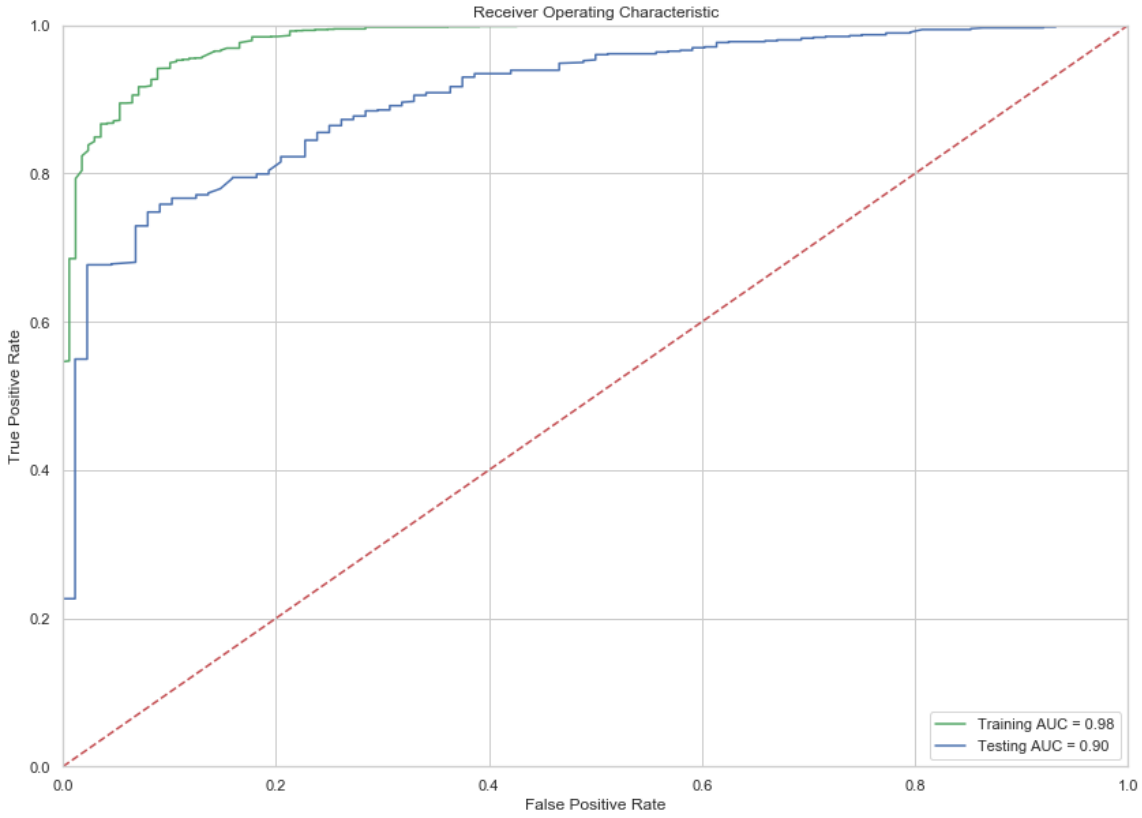
We started with hyper parameter tuning of XGB. Two crucial parameters – number of estimators and max depth were tuned.

We used cross validated grid search for hyper parameter tuning. We searched based on number of estimators as 50, 100, 200, 500 and max depth as 5, 10, 50, 100, 200. 3-fold cross validation was used in grid search. ROC AUC scoring was used to evaluate the predictions on validation sets.

The distribution of ROC AUC values based on hyper parameters values are shown below:

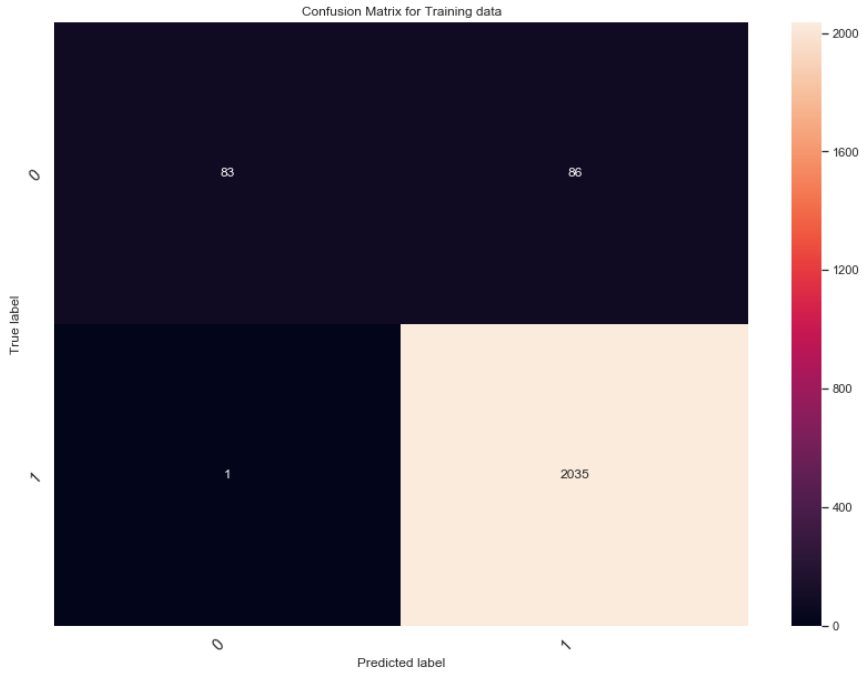


ROC curve of the final XGB model with best hyperparameter values is shown below:

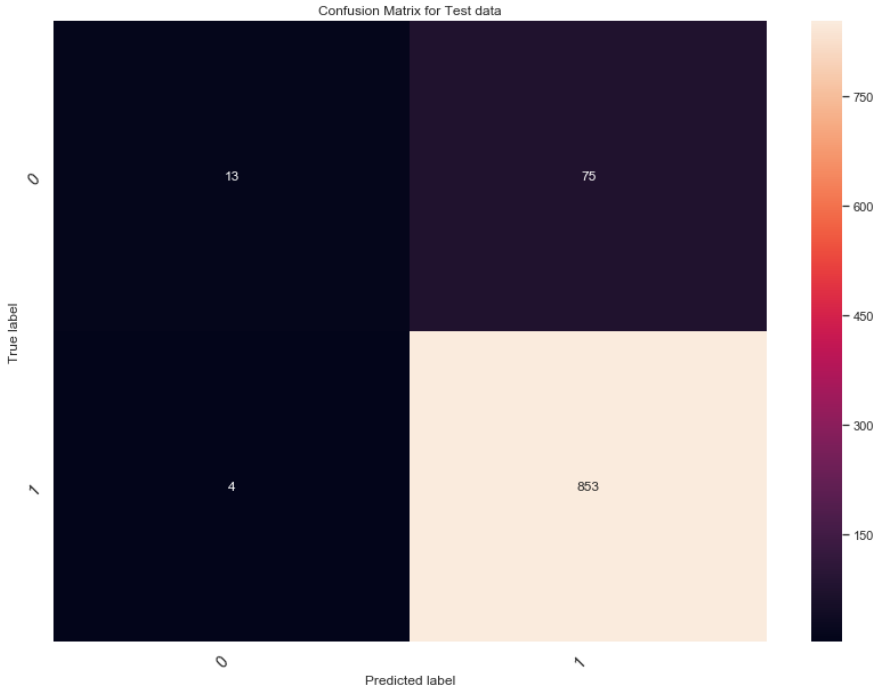


We plotted the confusion matrix.

Training data:



Test Data:



Test accuracy was calculated as 91.6%.

ML models' performance in summarized form:

Model	Max Depth	n_estimators	AUC
Random Forest	50	500	92.27%
XGB Classifier	5	100	91.64%