Starbucks Capstone Challenge Project Report

I. Definition

Project Overview

The Starbucks project is coming out from customer marketing domain. Traditional marketing analytics or scoreboards are essential for evaluating the success or failure of organization's past marketing activities. But today's marketers or organizations can leverage advanced marketing techniques like predictive modeling for customer behavior, predictive lead scoring, and all sorts of strategies based on predictive analytics insights.

Various Cases for Predictive Marketing Analytics are:

- *Detailed Lead Scoring*
- *Lead Segmentation for Campaign Nurturing*
- *Targeted Content Distribution*
- *Lifetime Value Prediction*
- *Churn Rate Prediction*
- *Upselling and Cross-Selling Readiness*
- *Understanding Product Fit*
- *Optimization of Marketing Campaigns*

In this Project, we would deal with the case of 'Optimization of Marketing Campaigns'. Predictive techniques can make an organization marketing investment much more efficient and helps in regularly validating results.

Connecting customer information to the operational data provides valuable insight into customer behavior and the health of your overall business.

Refer below links in order to have better understanding of the impact of predictive analytics for better marketing performance.

1. *Predictive Analytics: What it is and why it matters*
2. *Marketing Analytics for Data-Rich Environments*
3. *How to Use Predictive Analytics for Better Marketing Performance*

The entire code for this project analysis can be found here.

Problem Statement

In this project, we would go through the predictive modeling technique for customer behavior through Starbucks dataset example. Starbucks provided simulated data that mimics customer behavior on the Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks. Not all users receive the same offer, and this data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

In this project, Starbucks wants to connect offer data, customer data and transaction data (operational data) to gain insights about customer behavior and overall effectiveness of offers as a value for business.

With the above key statement in mind, below is the main objective or problem motivation or problem statement:

Build a model that predicts whether a customer would respond to an offer or not.

Above problem is a binary classification problem and could be solved by using supervised machine learning classification algorithms like Logistic Regression, Random Forest, Adaboost, Gradient Boosting. The brief outline of the solution

of this problem is that the Starbucks different datasets (offer, demographic & transcript) would be combined after data pre-processing (cleaning & transformation) and feature engineering to form a combined dataset. The appropriate and correct combined data would be fed into various machine learning classification models which would find the hidden traits of the customer behavior which plays an important role in influencing the customer to either respond to an offer or not. The trained classification models would classify the customers into target classes (1 if a customer would respond or 0 if a customer would not respond). The performance of the models would be compared based on appropriate performance metric (discussed in below section) for this project. The best performing classification model would be selected based on performance metric & training time and would be refined further by fine tuning the hyperparameters of the best estimator of the selected model.

Below are few case studies from customer marketing domain:

1. *How we use trees to predict customer behavior: random forests in action*
2. *Evaluation of Classification and Ensemble Algorithms for Bank Customer Marketing Response Prediction*
3. *"How Social Media Insights Turned Around Lexus' Holiday Campaigns" by Infegy*
4. *Machine-Learning Techniques for Customer Retention: A Comparative Study*

Metrics

Since, our combined dataset created from the Starbucks datasets (offer data, customer demographic data and transaction data) is nearly balanced (slightly imbalanced) in terms of distribution of target class (customers whom respond to an offer (54%) represented by class 1 and whom do not respond to an offer (46%) represented by class 0), performance metrics like accuracy would not be the right measure for performance of the model in this case and should be

avoided. Instead precision, recall and f1_score are good measures for evaluating a model in this case.

For this case, evaluating a model with precision and recall would provide better insight to its performance rather than accuracy. Because, Starbucks would like to send offers to those customers whom have more chances of redeeming the offers rather than to send offers to all customers which would allow Starbucks to have efficient marketing and would be able to achieve more value for business from the offers. F1-score metric is "the harmonic mean of the precision and recall metrics" and is better way of providing greater predictive power on the problem and how good the predictive model is making predictions. Refer Classification Accuracy is Not Enough: More Performance Measures You Can Use for more information.

II. Analysis

Data Exploration and Exploratory Visualization

The data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app and can be found here. The data is contained in three files:

- *portfolio.json — containing offer ids and meta data about each offer (duration, type, etc.)*
- *profile.json — demographic data for each customer*
- *transcript.json — records for transactions, offers received, offers viewed, and offers completed*

Here is the schema and explanation of each variable in the files:

portfolio.json schema

- *id (string) — offer id*
- *offer_type (string) — type of offer i.e. BOGO, discount, informational*
- *difficulty (int) — minimum required spend to complete an offer*
- *reward (int) — reward given for completing an offer*
- *duration (int) — time for offer to be open, in days*
- *channels (list of strings)*

profile.json schema

- *age (int) — age of the customer (missing value encoded as 118)*
- *became_member_on (int) — date when customer created an app account (format YYYYMMDD)*
- *gender (str) — gender of the customer (note some entries contain 'O' for other rather than M or F)*
- *id (str) — customer id*
- *income (float) — customer's income*

transcript.json schema

- *event (str) — record description (i.e. transaction, offer received, offer viewed, etc.)*
- *person (str) — customer id*
- *time (int) — time in hours since start of test. The data begins at time t=0*
- *value — (dict of strings) — either an offer id or transaction amount depending on the record*

Each of the portfolio, profile and transaction dataset was cleaned and transformed and below is the summary of each of the datasets.

portfolio data summary

1. *There are no null values in portfolio dataframe.*
2. *There are 3 offer types i.e. 4 bogo offers, 4 discount offers and 2 informational offers.*
3. *Categorical variables like offer_type & channels are OneHotEncoded.*
4. *Portfolio dataset features after cleaning & transforming are 'offer_id', 'difficulty', 'duration', 'reward', 'bogo', 'discount', 'informational', 'web', 'email', 'mobile' & 'social'. Below is the final portfolio dataset. See Figure 1.*

```
# Print portfolio dataframe
portfolio
```

	offer_id	difficulty	duration	reward	bogo	discount	informational	web	email	mobile	social
0	ae264e3637204a6fb9bb56bc8210ddfd	10	168	10	1	0	0	0	1	1	1
1	4d5c57ea9a6940dd891ad53e9dbe8da0	10	120	10	1	0	0	1	1	1	1
2	3f207df678b143eea3cee63160fa8bed	0	96	0	0	0	1	1	1	1	0
3	9b98b8c7a33c4b65b9aebfe6a799e6d9	5	168	5	1	0	0	1	1	1	0
4	0b1e1539f2cc45b7b9fa7c272da2e1d7	20	240	5	0	1	0	1	1	0	0
5	2298d6c36e964ae4a3e7e9706d1fb8c2	7	168	3	0	1	0	1	1	1	1
6	fafdcd668e3743c1bb461111dcafc2a4	10	240	2	0	1	0	1	1	1	1
7	5a8bc65990b245e5a138643cd4eb9837	0	72	0	0	0	1	0	1	1	1
8	f19421c1d4aa40978ebb69ca19b0e20d	5	120	5	1	0	0	1	1	1	1
9	2906b810c7d4411798c6938adc9daaa5	10	168	2	0	1	0	1	1	1	0

Figure 1 - final portfolio dataset

profile data summary

5. *There are 2175 observations with age 118 having gender as None and income as nan . Value 118 in age feature represents null value and therefore the observations of customers with age 118 were removed. See Figure 2 and 3.*

6. *Converted 'age' and 'income' feature into bins. 'Age' feature was binned into features 'age_10s', 'age_20s', 'age_30s', 'age_40s', 'age_50s', 'age_60s', 'age_70s', 'age_80s', 'age_90s' and 'age_100s'.*

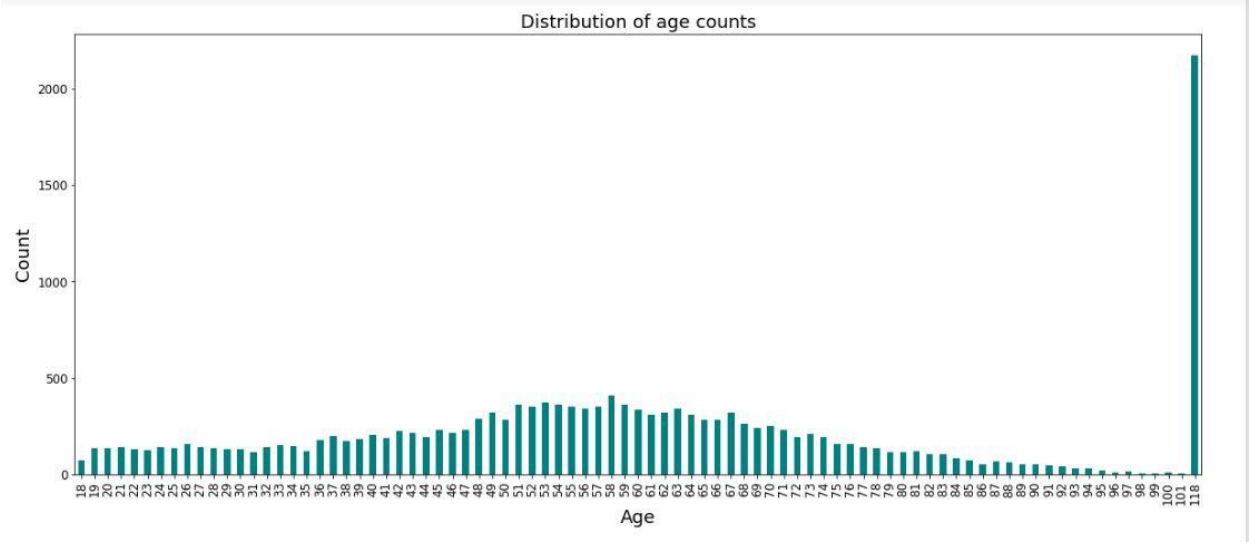

Figure 2 – Distribution of age counts

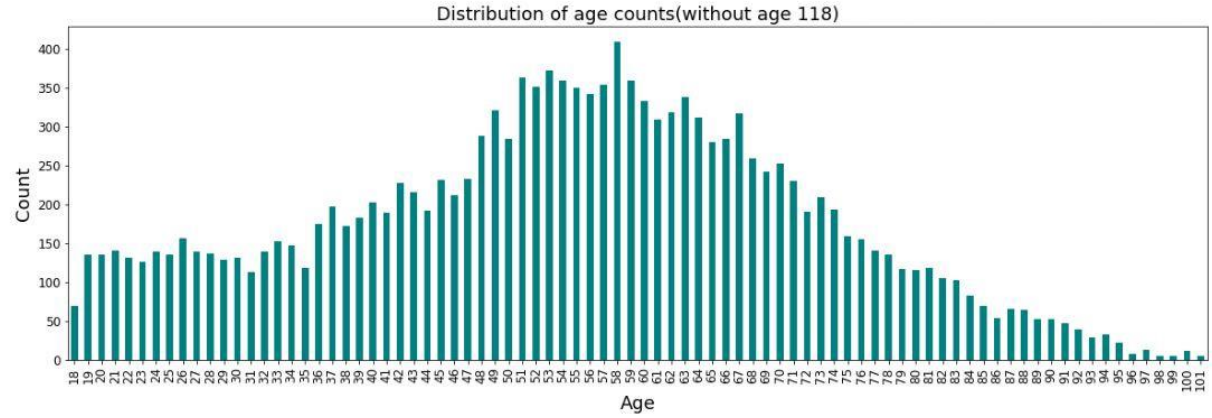

Figure 3 - Distribution of age counts(without age 118)

7. *'Income' feature was binned into features 'income_30ths', 'income_40ths', 'income_50ths', 'income_60ths', 'income_70ths',*

'income_80ths', 'income_90ths', 'income_100ths', 'income_110ths' and 'income_120ths'. See Figure 4.

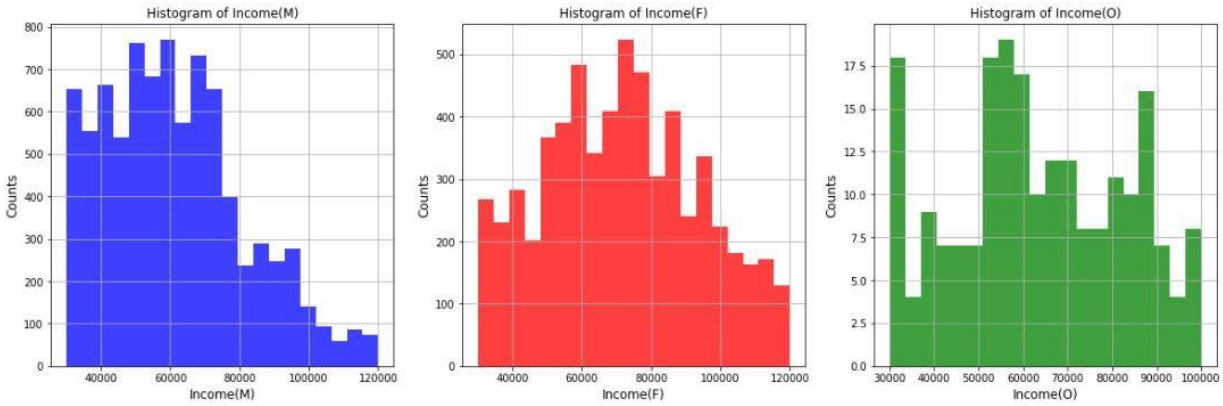

Figure 4 - Histogram of Income per Gender

8.'gender' feature was OneHotEncoded into 'F', 'M' and 'O' depicting females, males and others. See Figure 5.

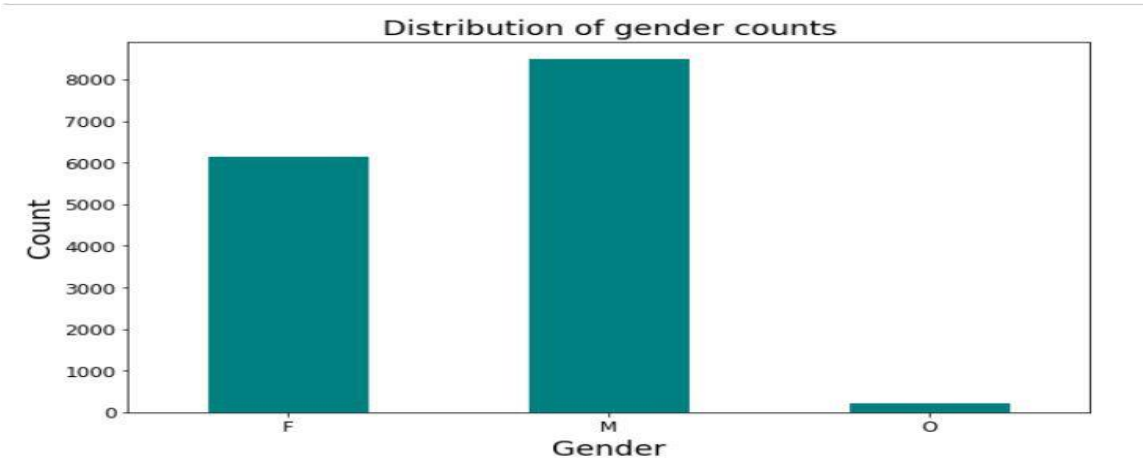

Figure 5 - Distribution of gender counts

9. Feature engineering was performed on feature ' became_member_on' and few others features were generated namely —

i) *'membership_tenure' — denoting length of membership of a customer*
ii) *'2013', '2014', '2015', '2016', '2017' and '2018' — year in which the customer became Starbucks member.*

iii) *'month_1', 'month_2', 'month_3', 'month_4', 'month_5', 'month_6', 'month_7', 'month_8', 'month_9', 'month_10', 'month_11' and 'month_12' — month in which customer became Starbucks member.*

10. *Profile dataset features are : 'customer_id', 'age_10s', 'age_20s', 'age_30s', 'age_40s', 'age_50s', 'age_60s', 'age_70s', 'age_80s', 'age_90s', 'age_100s', 'membership_tenure', '2013', '2014', '2015', '2016', '2017', '2018', 'month_1', 'month_2', 'month_3', 'month_4', 'month_5', 'month_6','month_7', 'month_8', 'month_9', 'month_10', 'month_11', 'month_12', 'F', 'M', 'O', 'income_30ths', 'income_40ths', 'income_50ths', 'income_60ths', 'income_70ths', 'income_80ths', 'income_90ths', 'income_100ths', 'income_110ths', 'income_120ths'.*

transcript data summary

11. *There are no null values in transcript dataframe.*

12. *There are 11.02 % i.e. 33772 observations in transcript dataframe which belong to customers with age 118. Therefore, we removed observations with customers having age 118 as we still have around 89% of data for further analysis. There are 45.45% of events as transactions, 24.38% of events as offers received, 18.28% of events as offers viewed and 11.89 % of events as offers completed.*

13. *We would separate events with type 'transactions' and 'offers' to be used later on for analysis. Below are the separate datasets for offers and transactions. See Figure 6 and 7.*

```
offers_df.head()
```

	customer_id	offer_id	time	offer_recd	offer_view	offer_comp
0	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0	1	0	0
2	e2127556f4f64592b11af22de27a7932	2906b810c7d4411798c6938adc9daaa5	0	1	0	0
5	389bc3fa690240e798340f5a15918d5c	f19421c1d4aa40978ebb69ca19b0e20d	0	1	0	0
7	2eeac8d8feae4a8cad5a6af0499a211d	3f207df678b143eea3cee63160fa8bed	0	1	0	0
8	aa4862eba776480b8bb9c68455b8c2e1	0b1e1539f2cc45b7b9fa7c272da2e1d7	0	1	0	0

Figure 6 - Offers dataset

```
transaction_df.head()
```

	customer_id	time	amount
12654	02c083884c7d45b39cc68e1314fec56c	0	0.83
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0	34.56
12659	54890f68699049c2a04d415abc25e717	0	13.23
12670	b2f1cd155b864803ad8334cdf13c4bd2	0	19.51
12671	fe97aa22dd3e48c8b143116a8403dd52	0	18.97

Figure 7 - transaction dataset

Algorithms and Techniques

1. *For predicting the customers whether they would respond to an offer or would not respond to an offer, we would use common Supervised Machine Learning classification algorithms like Logistic Regression, Random Forest Classifier, Adaboost Classifier and Gradient Boosting Classifier. Refer [Top 10 Machine Learning Algorithms] & [Boosting Algorithms: AdaBoost, Gradient Boosting and XGBoost] to know more about the algorithms.*

2. *Below is the overview of classification algorithms used:*

Logistic Regression

The name of this algorithm could be a little confusing in the sense that Logistic Regression machine learning algorithm is for classification tasks and not regression problems. The name 'Regression' here implies that a linear model is fit into the feature space. This algorithm applies a logistic function to a linear combination of features to predict the outcome of a categorical dependent variable based on predictor variables. Logistic Regression majorly makes predictions to handle problems which require a probability estimate as output, in the form of 0/1.

Logistic regression algorithm is suited when the need is to classify elements into two categories based on the explanatory variable which is called Binary Logistic Regression. For example-classify females into 'young' or 'old' group based on their age. For our case, we need to classify the customers into two categories based on (offer, demographic & transaction data), therefore, we would use Logistic regression.

Advantages of using Logistic Regression

- *Easier to inspect and less complex.*
- *Robust algorithm as the independent variables need not have equal variance or normal distribution.*
- *These algorithms do not assume a linear relationship between the dependent and independent variables and hence can also handle non-linear effects.*

Drawbacks of using Logistic Regression

- *When the training data is sparse and high dimensional, in such situations a logistic model may over fit the training data.*
- *Logistic regression algorithms require more data to achieve stability and meaningful results. These algorithms require minimum of 50 data points per predictor to achieve stable outcomes.*
- *It is not robust to outliers and missing values.*

Random Forest

Random Forest is the machine learning algorithm that uses a bagging approach to create a bunch of decision trees with random subset of the data. A model is trained several times on random sample of the dataset to achieve good prediction performance from the random forest algorithm. In this ensemble learning method, the output of all the decision trees in the random forest, is combined to make the final prediction. The final prediction of the random forest algorithm is derived by polling the results of each decision tree or just by going with a prediction that appears the most times in the decision trees. It can handle a large amount of training data efficiently and are inherently suited for multi-class problems.

Random Forest maintains accuracy when there is a missing data, have resistance to outliers, avoid the overfitting problem, works well with numerical, binary and categorical values, without scaling, transformation or modification. It has Implicit feature selection as it gives estimates on what variables are important in the classification. Since, there are numerical, binary and categorical variables in our dataset which have been OneHotEncoded and also, we would like to find out the important features which influences a customer to respond to an offer or not, Random Forest algorithm is a suitable choice in this case.

How Random Forest Algorithm works

Bagging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners (Decision trees). Given a training set $X = x_1, \ldots, x_n$ *with responses* $Y = y_1, \ldots, y_n$, *bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:*

For $b = 1, \ldots, B$:

1. *Sample, with replacement,* n *training examples from* X, Y; *call these Xb, Yb.*
2. *Train a classification or regression tree* f_b on X_b, Y_b.

After training, predictions for unseen samples x' *can be made by averaging the predictions from all the individual regression trees on* x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

or by taking the majority vote in the case of classification trees. See Figure 8.

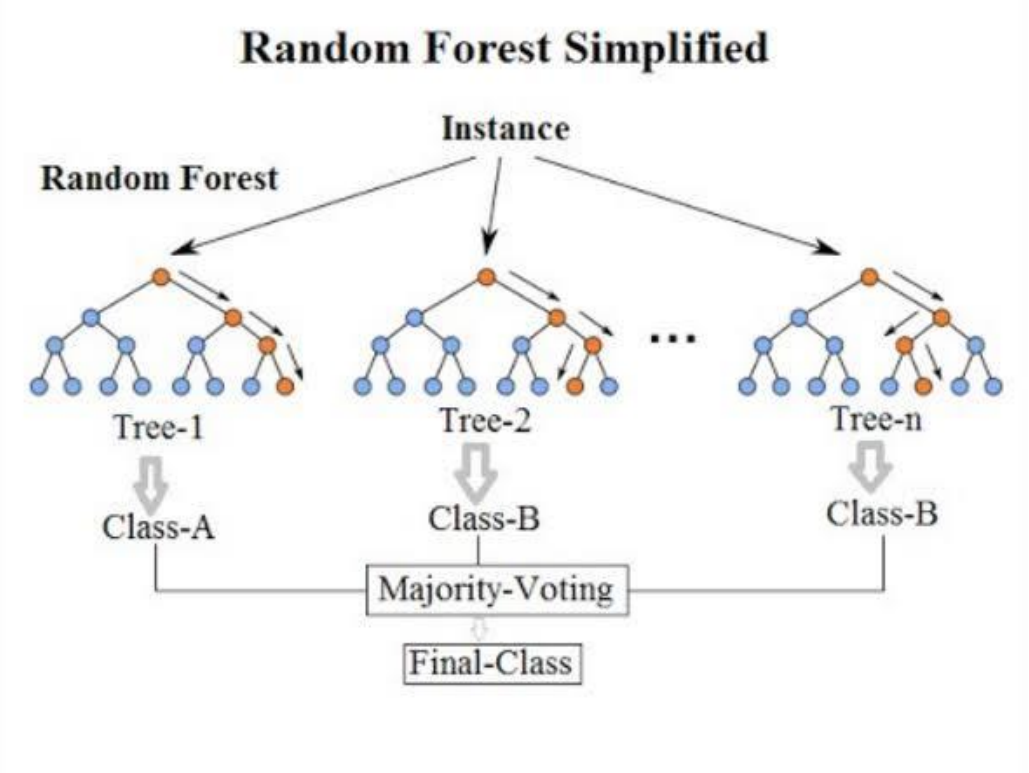

Figure 8 - Random Forest Algorithm

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

The number of samples/trees, B, *is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and*

nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

Bagging to random forests

The above procedure describes the original bagging algorithm for trees. Random forests differ in only one way from this general scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions can be found here.

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend $p/3$ (rounded down) with a minimum node size of 5 as the default. In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters. Refer here for information on this.

Random Forest algorithm pseudocode

There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage.

Creation pseudocode

1. *Randomly select "**K**" features from total "**m**" features where **k << m.***

2. *Among the "**K**" features, calculate the node "**d**" using the best split point.*
3. *Split the node into **daughter nodes** using the **best split.***
4. *Repeat the **a to c** steps until "l" number of nodes has been reached.*
5. *Build forest by repeating steps **a to d** for "n" number times to create **"n" number of trees.***

Figure 9 shows the process of randomly selecting features

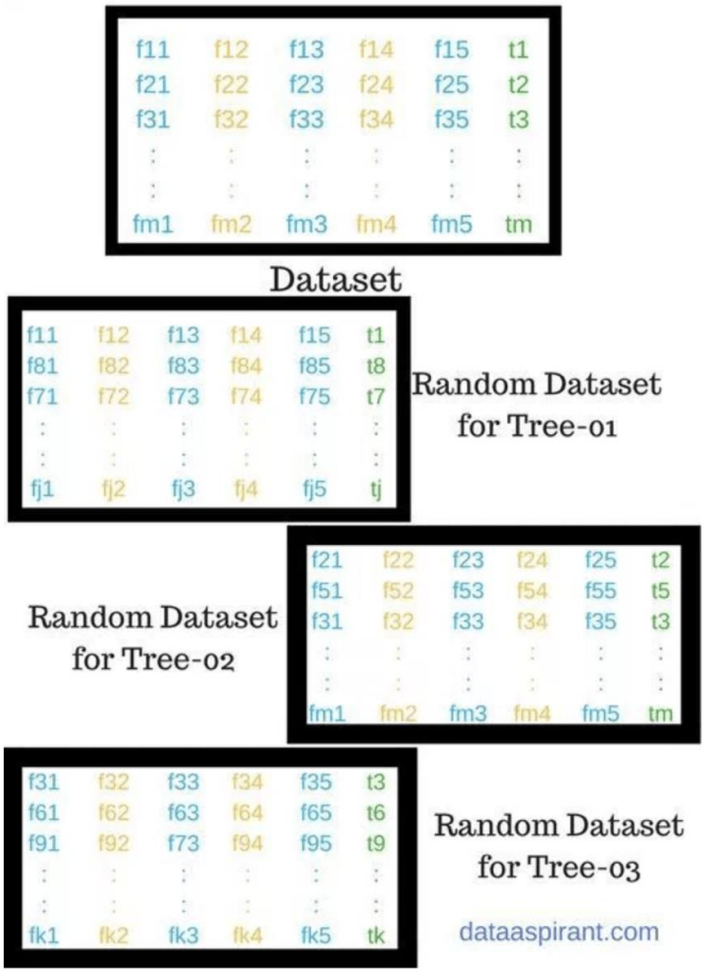

Figure 9 - process of randomly selecting features

In the next stage, with the random forest classifier created, prediction is made. The random forest prediction pseudocode is shown below:

Prediction pseudocode

1. *Take the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target).*
2. *Calculate the votes for each predicted target.*
3. *Consider the high voted predicted target as the final prediction from the random forest algorithm.*

Variable importance

Random forests can be used to rank the importance of variables in a regression or classification problem in a natural way. The following technique was described in Breiman's original paper.

The first step in measuring the variable importance in a data set $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$ *is to fit a random forest to the data. During the fitting process the out-of-bag error for each data point is recorded and averaged over the forest (errors on an independent test set can be substituted if bagging is not used during training).*

To measure the importance of the j*-th feature after training, the values of the* j*-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the* j*-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.*

Features which produce large values for this score are ranked as more important than features which produce small values. The statistical definition of the variable importance measure can be found here.

Advantages of using Random Forest

- *Overfitting is less of an issue with Random Forests, unlike decision tree machine learning algorithms. There is no need of pruning the random forest.*
- *Random Forest is one of the most effective and versatile machine learning algorithm for wide variety of classification and regression tasks, as they are more robust to noise.*
- *It is difficult to build a bad random forest. In the implementation of Random Forest Machine Learning algorithms, it is easy to determine which parameters to use because they are not sensitive to the parameters that are used to run the algorithm. One can easily build a decent model without much tuning.*
- *Has higher classification accuracy.*
- *It can handle thousands of input variables without variable deletion.*
- *It generates an internal unbiased estimate of the generalization error as the forest building progresses.*
- *It has methods for balancing error in class population unbalanced data sets.*

Drawbacks of using Random Forest

- *They might be easy to use but analyzing them theoretically, is difficult.*
- *If the data consists of categorical variables with different number of levels, then the algorithm gets biased in favor of those attributes that have more levels. In such situations, variable importance scores do not seem to be reliable.*
- *When using Random Forest algorithm for regression tasks, it does not predict beyond the range of the response values in the training data.*

***Adaboost* (Adaptive Boosting)**

The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. AdaBoost works on improving the areas where the base learner fails. The base learner is a machine learning algorithm which is a weak learner and upon which the boosting method is applied to turn it into a strong learner. Any machine learning algorithm that accept weights on training data can be used as a base learner. Mostly, Decision stumps (which is just one node of a Decision Tree) are used as the base learner or as a weak classifier. AdaBoost therefore acts as a meta algorithm, which can be used as a wrapper for other classifiers

It fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.

Since, Boosting grants power to machine learning models to improve their predictions for binary classification problems. we would use Adaboost algorithm to build strong model to predict the customer response in our case and find out the best features which influence the customer response.

Advantages of using Adaboost

- *It is fast, simple and easy to program.*
- *It has the flexibility to be combined with any machine learning algorithm.*
- *It can achieve similar classification results with much less tweaking of parameters or settings.*

Drawbacks of using Adaboost

- *It can be sensitive to noisy data and outliers.*
- *Weak classifiers being too weak can lead to low margins and overfitting.*
- *Time and computation expensive.*

Gradient Boosting

Gradient Boosting is also a boosting algorithm and it also tries to create a strong learner from an ensemble of weak learners. This algorithm is similar to Adaptive Boosting(AdaBoost) but differs from it on certain aspects. Gradient Boosting trains many models in a gradual, additive and sequential manner.

The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners. While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function. The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimize.

We would use Gradient Boosting Algorithm in our case as it also builds a strong model using weak learners just like Adaboost and would see which boosting (Adaptive or Gradient) builds a better performing model for binary classification.

Advantages of using Gradient Boosting

- *Often provides predictive accuracy that cannot be beat.*
- *Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible.*
- *Often works great with categorical and numerical values as is.*

Drawbacks of using Gradient Boosting

- *It will continue to improve to minimize all errors. This can overemphasize outliers and cause overfitting. Must use cross-validation to neutralize.*
- *Time and computation expensive.*
- *The high flexibility results in many parameters that interact and influence heavily the behavior of the approach and can be harder to tune*

3. *We would combine the portfolio, profile and transcript data.*

4. *We would drop the features like 'customer_id', 'offer_id', 'time', 'email' which would not play any role in training the model.*
    - *i) Drop 'customer_id' & 'offer_id' as these are unique identifier.*
    - *ii) Drop 'time' as it is not required. It was used to check if a customer responded to an offer or not and to calculate total amount and then feed the appropriate data in the correct form as input to the machine learning algorithms.*
    - *iii) Drop 'email' as it has only 1 value in all the observations and the value is 1.*

5. *Combined data after removing unnecessary features would be split into training and testing set. Training set would be fed into Machine Learning Classifiers as input and our model would be trained to find the hidden patterns in the training set and would classify the customers into target classes (1 if a customer would respond or 0 if a customer would not respond) which would be the output.*

Benchmark

1. *Below is the distribution of the target class in the combined data. See Figure 10 and 11.*

```
# Separate features and labels of combined_data_df
X = combined_data_df.drop(columns=['cust_response'])
y = combined_data_df.loc[:,['cust_response']]
```

```
# Distribution of target class
y.squeeze().value_counts()
```

```
1    35854
0    30647
Name: cust_response, dtype: int64
```

Figure 10 - Distribution of target class in combined data

```
# percenatge of distribution of target class
round((y.squeeze().value_counts()/y.squeeze().count())*100,2)
```

```
1    53.91
0    46.09
Name: cust_response, dtype: float64
```

Figure 11 - Percentage of distribution of target class in combined data

From above, we can observe that our dataset is nearly balanced in terms of distribution of target class. Our target class has nearly equal number of customers whom responded to an offer (54%) and whom did not responded to an offer (47%). Since, our dataset is nearly balanced, we do not have to deal with techniques to combat class imbalance in this case. To know more about class imbalance and how to deal with it, refer [8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset].

2. *We would create a baseline model (a naïve model) against which we would compare our model to determine if our model is performing better than baseline model or not. Our baseline model would predict that all users would respond to the offer. So, we would calculate f1_score of the baseline model. See Figure 12.*

```
# Create a baseline model against which we would compare our model to determine
# if our model is performing better than baseline model or not
# Our baseline model would predict that all users would respond to the offer
# So, we will calculate f1_score of the baseline model

baseline_model_f1_score = f1_score(y_train.squeeze().values, np.ones(y_train.shape[0]))
print('baseline model have f1_score: {}'.format(round(baseline_model_f1_score,4)))
```

```
baseline model have f1_score: 0.6996
```

Figure 12 - baseline model f1_score

3. *Our base model produced f1_score of 0.6996. Therefore, our other trained classifier should produce better f1_score than the threshold f1_score of 0.6996.*

III. Methodology

Data Preprocessing

1. *The difficulty which was encountered during the coding process or data pre-processing process was implementing the logic and strategy while preparing the combined data through function create_combined_data() and it consumed a lot of time. This function is the foundation for getting the right data and for implementing the forthcoming machine learning techniques which solely depends upon the correct data in the right format. But this difficulty could be justified by :*

   "Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning' is basically feature engineering." - Prof. Andrew Ng.

2. *See Figure 13 for count of customers per offer type and event type.*

```
# Calculate the count of customers by grouping offer type and event
offers_df.groupby(['offer_type', 'event'])['customer_id'].count()
```

```
offer_type     event
bogo           offer completed    15258
               offer received     26537
               offer viewed       22039
discount       offer completed    17186
               offer received     26664
               offer viewed       18461
informational  offer received     13300
               offer viewed        9360
Name: customer_id, dtype: int64
```

Figure 13 - grouping customers by offer type and event type

From above, we can observe that offer type 'bogo' and 'discount' has event 'offer received', 'offer viewed' and 'offer completed' whereas offer type 'informational' has only event 'offer received' and 'offer viewed'. Therefore, for offer type 'bogo' and 'discount', we will consider the offer as completed when there would be an event 'offer viewed' followed by 'offer completed' within the offer period whereas for offer type 'informational', we will consider it as completed when a customer makes a purchase transaction after viewing the informational offer within the offer period.

3. *If a customer spends at least minimum amount in purchases during the validity period, the customer completes the offer. If a customer views and completes the offer within the offer duration, the target variable or label 'cust_response' would be assigned a value of 1 else 0.*

4. *transaction, demographic and offer data were processed through function create_combined_data() to get the combined data which have the following features: 'customer_id', 'offer_id', 'time', 'difficulty', 'duration', 'reward', 'bogo', 'discount', 'informational', 'web', 'email', 'mobile', 'social', 'membership_tenure', 'F', 'M', 'O', 'age_10s', 'age_20s', 'age_30s', 'age_40s', 'age_50s', 'age_60s', 'age_70s', 'age_80s',*

'age_90s', 'age_100s', '2013', '2014', '2015', '2016', '2017', '2018', 'month_1', 'month_2', 'month_3', 'month_4', 'month_5', 'month_6', 'month_7','month_8', 'month_9', 'month_10', 'month_11', 'month_12', 'income_30ths', 'income_40ths', 'income_50ths', 'income_60ths', 'income_70ths', 'income_80ths', 'income_90ths', 'income_100ths', 'income_110ths', 'income_120ths', 'total_amount' and 'cust_response'.

Implementation

1. *For all the 4 classification models, default parameter values were used initially and only the random_state variable was set in order to reproduce the result. We would refine the parameters of the best model with the help of GridsearchCV later on. See figure 14.*

```
# Initialize classification algorithms
lor = LogisticRegression(random_state=42) # LogisticRegression
rfc = RandomForestClassifier(random_state=42) # RandomForestClassifier
abc = AdaBoostClassifier(random_state=42) # AdaBoostClassifier
gbc = GradientBoostingClassifier(random_state=42) # GradientBoostingClassifier
```

Figure 14 - Default parameters for all 4 classifiers

2. *Before training Classifier's like Logistic Regression, Random Forest, Adaboost & GradientBoosting, combined data was split into training and test data. See Figure 15, 16, 17 and 18.*

```
# Distribution of the target class in training set
y_train.squeeze().value_counts()
```

```
1    26834
0    23041
Name: cust_response, dtype: int64
```

Figure 15 - Distribution of target class in training set

```
# Percenatge of distribution of the target class in training set
round((y_train.squeeze().value_counts()/y_train.squeeze().count())*100,2)
```

```
1    53.8
0    46.2
Name: cust_response, dtype: float64
```

Figure 16- Percentage of distribution of target class in training set

```
# Distribution of the target class in test set
y_test.squeeze().value_counts()
```

```
1    9020
0    7606
Name: cust_response, dtype: int64
```

Figure 17 - Distribution of target class in test set

```
# Distribution of the target class in test set
round((y_test.squeeze().value_counts()/y_test.squeeze().count())*100,2)
```

```
1    54.25
0    45.75
Name: cust_response, dtype: float64
```

Figure 18 - Percentage of distribution of target class in test set

3. *From above, we can observe that our training & test set is also nearly balanced in terms of distribution of target class.*

4. *After combined data was split into train and test data, features like 'difficulty', 'duration', 'reward', 'membership_tenure', 'total_amount' were scaled using MinMaxScaler. It transforms features by scaling each feature to a given range, default is between 0 and 1.*

5. *After scaling, various classifier's were trained on train data i.e. training features were trained against the training label i.e. 'cust_response' feature and all classifier's produced better f1_score than baseline model's f1_score. See Figure 19.*

```
# Create clf_df dataframe from clf_dict
clf_dict['best_f1_score'] = clf_scores
clf_dict['time_taken(s)'] = clf_time_taken
clf_dict['best_est'] = clf_best_ests
clf_df = pd.DataFrame(clf_dict, index=clf_names)
clf_df
```

	best_f1_score	time_taken(s)	best_est
LogisticRegression	0.842906	5.38	LogisticRegression(C=1.0, class_weight=None, d...
RandomForestClassifier	0.921618	4.29	(DecisionTreeClassifier(class_weight=None, cri...
AdaBoostClassifier	0.896027	18.22	(DecisionTreeClassifier(class_weight=None, cri...
GradientBoostingClassifier	0.920881	60.06	([DecisionTreeRegressor(criterion='friedman_ms...

Figure 19 - f1_score of all 4 classifier's along with training time

6. *From above, all our trained classifier produced better f1_score than the threshold f1_score of 0.6996. RandomForestClassifier and GradientBoostingClassifier got nearly equal f1_score(approx. 0.92) but RandomForestClassifier took very less time to train than the GradientBoostingClassifier. Therefore, best performing classifier algorithm among the above 4 classifiers was RandomForestClassifier.*

Refinement

1. *The hyperparameters of the trained RandomForestClassifier was further fine tuned using GridSearchCV and our model got improved and produced a better f1_score of 0.9319 with cross-validation. See Figure 20.*

```
# Tune the best classifier(RandomForestClassifier) with the help of param grid in GridSearchCV
# The fine tuned model will be used with the test set
param_grid = {#'n_estimators': [10, 50, 80, 100],
              'n_estimators': [50],
              #'max_depth': [None, 2, 3, 4],
              'max_depth': [None],
              #'min_samples_split': [2,3,4],
              'min_samples_split': [3],
              #'min_samples_leaf': [1,2,3],
              'min_samples_leaf': [1]
             }

rfc = RandomForestClassifier(random_state=42)
rfc_best_score, rfc_best_est, _ = fit_classifier(rfc, param_grid)
rfc_best_est
```

```
Training RandomForestClassifier :
RandomForestClassifier
Time taken : 19.24 secs
Best f1_score : 0.9319
****************************************
```

Figure 20 - Fine tuning of hyperparameters of trained Random Forest Classifier

IV. Results

Model Evaluation, Validation and Justification

1. *Test data was prepared in exactly the same way as trained data and classes for test data was predicted using the fine tuned RandomForestClassifier. See Figure 21 and 22.*

```
# Print shape of dataframe X_test_scaled
X_test_scaled.shape
```

```
(16626, 52)
```

```
# Print shape of y_test
y_test.shape
```

```
(16626, 1)
```

Figure 21 - Shape of test features and labels

```
# Classification of test data using best model trained on train data
y_pred = rfc_best_est.predict(X_test_scaled)
y_pred
```

```
array([0, 1, 1, ..., 0, 0, 1])
```

```
# Print shape of y_pred
y_pred.shape
```

```
(16626,)
```

Figure 22 - Test set predictions and shape of test set predictions

2. *Our best estimator produced f1_score of 0.9336 on test data, which is quite good.*

3. *true negatives, false positives, false negatives and true positives from confusion matrix were calculated which are below:*
   true negatives: 6733
   false postives: 873
   false negatives: 359
   true postives: 8661

4. *Feature importances given by best estimator of the trained model were calculated. Below are the top 10 features along with their importances. See Figure 23 and 24.*

```
# Print top 10 features
feat_imp_df[:10]
```

	feature	feat_imp	feat_imp_perc
0	total_amount	0.618193	61.82
1	membership_tenure	0.093657	9.37
2	social	0.018677	1.87
3	difficulty	0.016082	1.61
4	duration	0.015524	1.55
5	reward	0.014503	1.45
6	2018	0.012365	1.24
7	2016	0.010421	1.04
8	income_30ths	0.007922	0.79
9	age_50s	0.007517	0.75

Figure 23 - Top 10 features with their importances

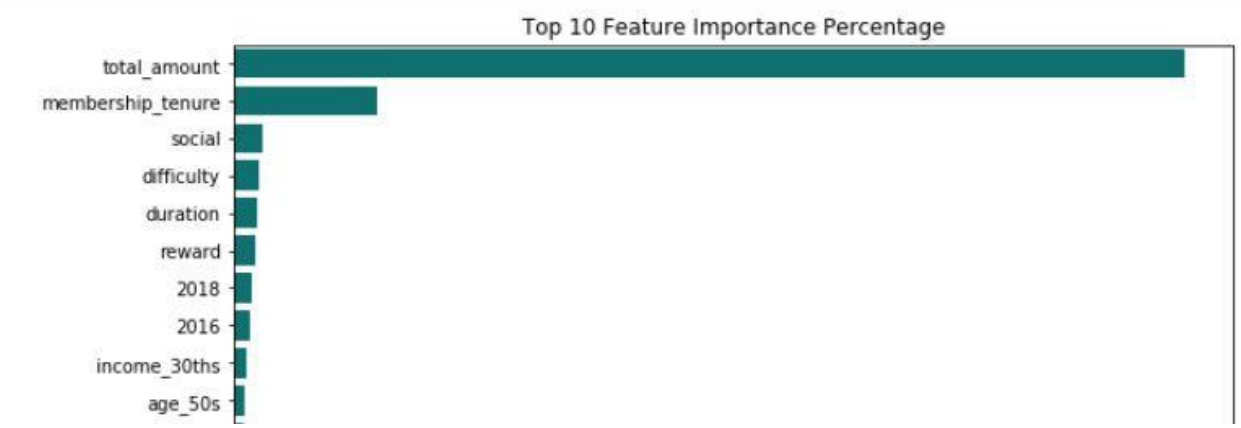

Figure 24 - Horizontal bar plot of top 10 features importances

5. *Below is the observation about top 10 features –*

- *Top 10 features which influence whether the customer will respond to an offer or not after viewing the offer are: 'total_amount', 'membership_tenure' , 'social', 'difficulty', 'duration', 'reward', '2018', '2016', 'income_30ths' and 'age_50s'.*

- *'total_amount' spent by the customer is the biggest and largest feature which could one sided influence if the customer will complete an offer after viewing the offer i.e. how much a customer spend is likely to decide if a customer will complete an offer. Therefore, knowing how much a customer can spend beforehand can definitely help in determining if a customer will respond to an offer and complete it. For this, another supervised learning algorithms i.e. regression models would be required in order to predict the total amount an individual could spend based on offer data and customer demographic data.*
- *After 'total_amount', 'membership_tenure' is the second largest feature - which represents, how long the customer has been the member of Starbucks reward program heavily influence if the customer will complete an offer after responding to it.*
- *After 'membership_tenure', 'social feature' - which represents, if Starbucks sent the offer to customer via social media is likely to be responded more than the other mode of communication.*
- *After 'social' feature, the 'difficulty' feature - which represents, minimum amount to spent in order to complete the offer influence if the customer would response and complete the offer.*
- *After 'difficulty' feature, 'duration' feature - which represents, how long the offer is valid influence the customer response and completion of the offer.*
- *After 'duration' feature, 'reward' feature - which represents, how much amount as a reward, a customer is getting back influence if the customer would response and complete the offer.*
- *After 'reward' feature, '2018' and '2016' feature, which represents - if a customer became member of Starbucks reward program in years 2016 & 2018, had more chance of responding to an offer and completing it.*
- *After '2018' and '2016' feature, 'income_30ths' feature, which represents - if customer's income is in 30000's which is the starting income group, then customer is likely to respond more to an offer and complete it.*

- *After 'income_30ths' feature, 'age_50s' feature which represents - if the customer age is in 50's, then customer is likely to respond more to an offer and complete it.*

6. *Since, our final fine tuned RandomForestClassifier best estimator produced f1_score of 0.9336 on test data, our final model is stronger than the benchmark model or baseline model or naïve model reported earlier having f1_score of 0.6996.*

V. Conclusion

Free-Form Visualization

1. *Plot confusion matrix without normalization for predictions on the test set. See Figure 25.*

```
Confusion matrix, without normalization
[[6733  873]
 [ 359 8661]]
```

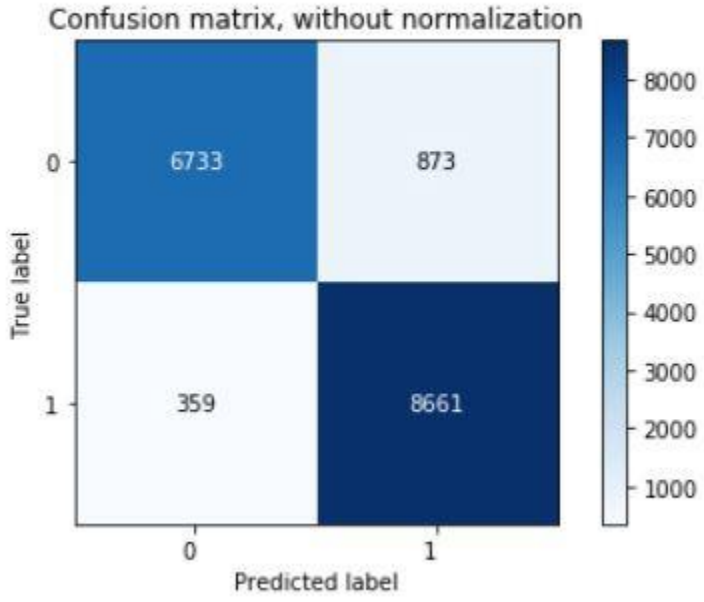

Figure 25 - Confusion matrix, without normalization

2. *Plot normalized confusion matrix for predictions on the test set. See Figure 26.*

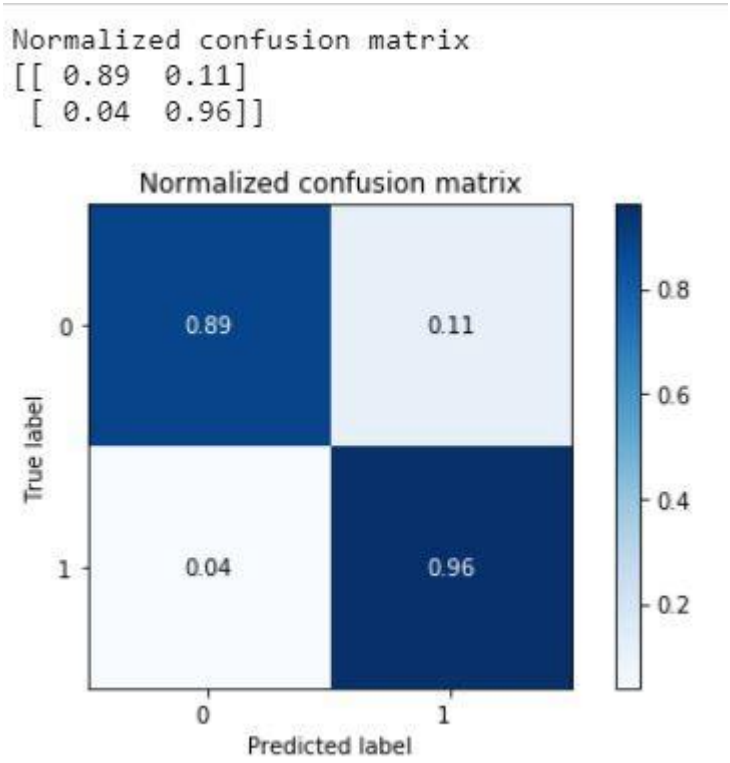

Figure 26 - Confusion matrix with normalization

3. *From above normalized confusion matrix, we observed that there are 4% of chances of misclassifying an individual whom would normally respond to an individual that would not respond to offer. Similarly, there are 11% of chances of misclassifying an individual whom would not respond to an individual whom would respond to an offer. As, False Negatives is less than False Positives, our predictive model is doing good as it has very low chances of missing an individual whom would respond. As Starbucks would not like to miss to send offers to individuals whom would respond to offers, this model would work fine in this case and by using this model Starbucks would not miss sending offers by great extent to individuals whom would respond to offers and therefore, overall business revenue would not get affected. Also, Starbucks would not mind sending offers to*

few individuals whom would not respond if Starbucks is able to make sure that they have covered up the individuals whom would respond to offers by great extent. Therefore, our predictive model is well suited for this case.

Reflection

The most interesting aspect of this project which I really liked was how different set of data i.e. offer data, customer demographic data and transcript data were combined to gain insights using predictive modeling techniques and analysis to provide better business decisions and value to the business. The toughest part of this entire analysis was finding logic and strategy to make combined dataset based on the duration of the offer when it was active for customers.

Improvement

1. *"Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning' is basically feature engineering." - Prof. Andrew Ng. Therefore, more feature engineering could be performed on offer, customer demographic and transaction data in order to have more better model.*

2. *Also, we can improve this project by taking up another problem statement i.e. determining how much a customer could spend based upon the offer data and demographic data using the supervised machine learning regression algorithms which inturn would help in finding out if the customer would respond or not as 'total amount' which a customer could spend is the top most feature in the best trained classifier model.*

3. *Also, we can improve the project by making our nearly balanced (slightly imbalanced) dataset into a perfectly balanced dataset using [8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset]. This would further improve the performance of classification model.*

4. *We can also perform the clustering modeling (behavioral clustering, product-based clustering, brand-based clustering) for customer segmentation into groups based on several variables at once. With it, we can target specific demographics and personas for different targets.*

The entire code for this analysis could be found here.