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 modelling for customer behaviour, 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 behaviour 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 modelling technique for customer behaviour through Starbucks dataset example. Starbucks provided simulated data that mimics customer behaviour 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 behaviour 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 behaviour 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 behaviour: 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. 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

II. Analysis

Data Exploration and Exploratory Visualization

The data set contains simulated data that mimics customer behaviour 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

- 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

profile.json

- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- 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

- 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.

```
In [4]: portfolio = clean_portfolio()
        print(portfolio)
```

Out[4]:

	offerid	difficulty	durationdays	reward	bogo	discount	informational	email	mobile	social	web
0	ae264e3637204a6fb9bb56bc8210ddfd	10	7	10	1	0	0	1	1	1	0
1	4d5c57ea9a6940dd891ad53e9dbe8da0	10	5	10	1	0	0	1	1	1	1
2	3f207df678b143eea3cee63160fa8bed	0	4	0	0	0	1	1	1	0	1
3	9b98b8c7a33c4b65b9aebfe6a799e6d9	5	7	5	1	0	0	1	1	0	1
4	0b1e1539f2cc45b7b9fa7c272da2e1d7	20	10	5	0	1	0	1	0	0	1
5	2298d6c36e964ae4a3e7e9706d1fb8c2	7	7	3	0	1	0	1	1	1	1
6	fafdcd668e3743c1bb461111dcafc2a4	10	10	2	0	1	0	1	1	1	1
7	5a8bc65990b245e5a138643cd4eb9837	0	3	0	0	0	1	1	1	1	0
8	f19421c1d4aa40978ebb69ca19b0e20d	5	5	5	1	0	0	1	1	1	1
9	2906b810c7d4411798c6938adc9daaa5	10	7	2	0	1	0	1	1	0	1

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.

Print few rows of the customer profile data

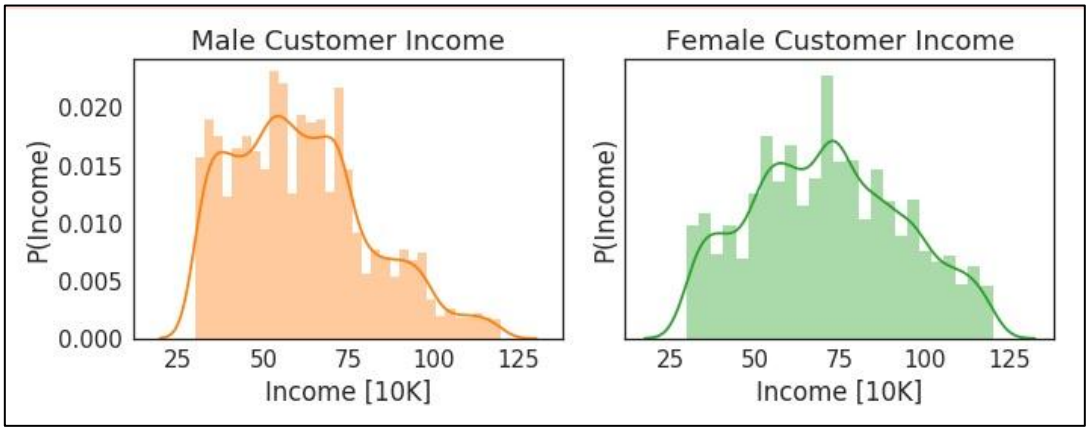
```
In [5]: profile.head()
```

Out[5]:

	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0
2	118	20180712	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0
4	118	20170804	None	a03223e636434f42ac4c3df47e8bac43	NaN

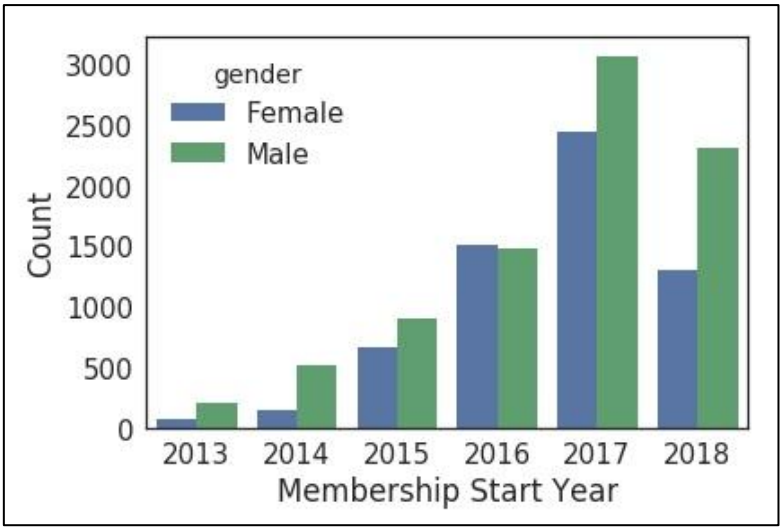
Plot Income Distribution as a Function of Gender

Results suggest that the minimum and maximum income for both male and female customers is approximately the same. However, male customer income is slightly biased towards lower values compared to female customer income.



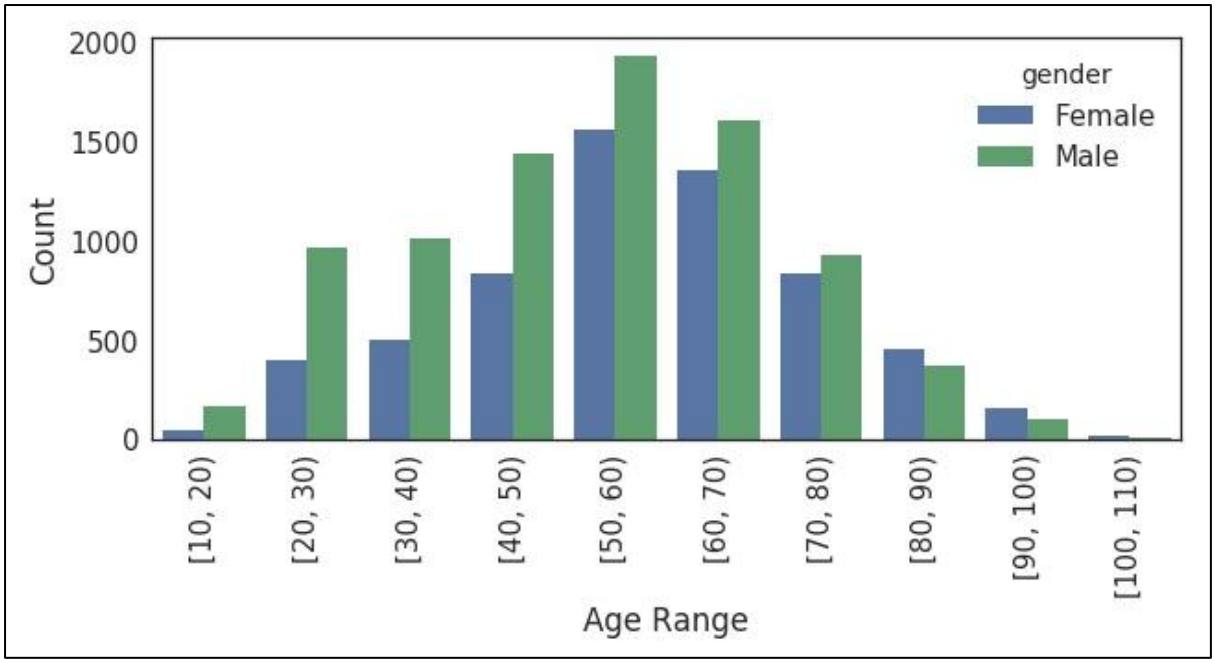
Evaluate Membership Start Year Statistics

The results suggest that most customers recently joined the Starbucks rewards program. These results also suggest that there are more male customers than female customers.



Evaluate Age Range Statistics

These results suggest that the average customer age is between 50 and 60 years old.



Transaction data EDA

- Customer transaction record attributes
  - event (str) - Record description (i.e. transaction, offer received, offer viewed, etc.)

- person (str) - Customer id
- time (int) - Time in hours. The data begins at time t=0
- value - (dict of strings) - Either an offer id or transaction amount depending on the record

- Customer transaction data EDA conclusions
  - Need to separate offer and customer purchase data
  - Results suggest ~ 45 % of the events are customers purchases and ~ 55% of the events describe customer offers

```
In [19]: transcript.head()
```

Out[19]:

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2	offer received	e2127556f4f64592b11af22de27a7932	0	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}

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

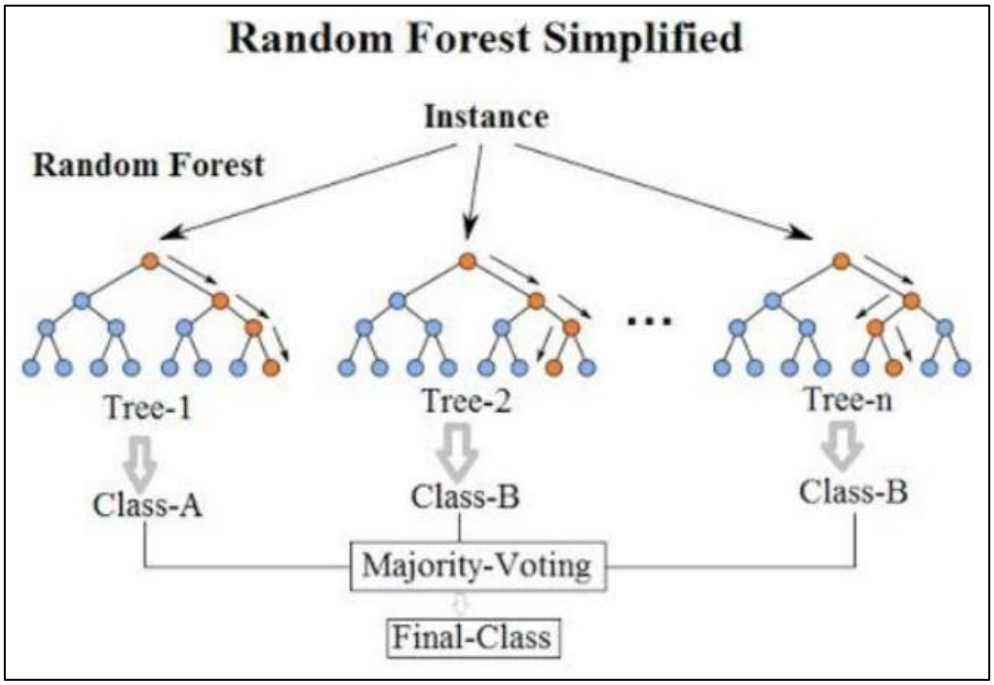
important in the classification. Since, there are numerical, binary and categorical variables in our dataset which have been OneHotEncoded and also, we would like to find out the important features which influences a customer to respond to an offer or not, Random Forest algorithm is a suitable choice in this case.

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 = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

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

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on, or by taking the majority vote in the case of classification trees.



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.

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, √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.**

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 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 algorithms 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 analysing them theoretically, is difficult.
- If the data consists of categorical variables with different number of levels, then the algorithm gets biased in favour 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 behaviour 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.

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. These results suggest that distribution of offers in the simulated Starbucks mobile application data is approximately uniform. They also imply that the percentage of an offer's success ranges from ~ 6% to 75%, with the two least successful offers being informational.

```
In [64]: percent_success = initialize_percent_success(portfolio,
                                                     training_data)

         percent_success #printing success
```

Out[64]:

	offerid	count	percentsuccess	difficulty	durationdays	reward	bogo	discount	informational	email	mobile	social	web
0	fafdcd668e3743c1bb461111dcafc2a4	5223	75.090944	10	10	2	0	1	0	1	1	1	1
1	2298d6c36e964ae4a3e7e9706d1fb8c2	5259	72.485263	7	7	3	0	1	0	1	1	1	1
2	f19421c1d4aa40978ebb69ca19b0e20d	5203	61.925812	5	5	5	1	0	0	1	1	1	1
3	ae264e3637204a6fb9bb56bc8210ddfd	5312	54.725151	10	7	10	1	0	0	1	1	1	0
4	4d5c57ea9a6940dd891ad53e9dbe8da0	5172	51.798144	10	5	10	1	0	0	1	1	1	1
5	9b98b8c7a33c4b65b9aebfe6a799e6d9	5247	48.275205	5	7	5	1	0	0	1	1	0	1
6	2906b810c7d4411798c6938adc9daaa5	5238	47.098129	10	7	2	0	1	0	1	1	0	1
7	0b1e1539f2cc45b7b9fa7c272da2e1d7	5330	45.816135	20	10	5	0	1	0	1	0	0	1
8	3f207df678b143eea3cee63160fa8bed	5256	7.800609	0	4	0	0	0	1	1	1	0	1
9	5a8bc65990b245e5a138643cd4eb9837	5228	6.006121	0	3	0	0	0	1	1	1	1	0

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.

2. Before training Classifier's like Logistic Regression, Random Forest, naive predictor & GradientBoosting, combined data was split into training and test data.

3. Estimated Feature Importance

- "*Feature importance*" refers to a numerical value that describes a feature's contribution to building a model that maximizes its evaluation metric
- These results suggest that the top five features for this problem are:

1. Offer difficulty (how much money a customer must spend to complete an offer)
2. Offer duration
3. Offer reward
4. Customer income
5. Whether a customer created an account on the Starbucks rewards mobile application in 2018

IV. Results

Model Evaluation, Validation and Justification

Evaluate naive predictor performance

- A naive predictor assumes that all customer offers were successful

```
In [77]: naive_predictor_accuracy = accuracy_score(y_train,np.ones(len(y_train)))
         naive_predictor_f1score = f1_score(y_train, np.ones(len(y_train)))

         print("Naive predictor accuracy: %.3f" % (naive_predictor_accuracy))
         print("Naive predictor f1-score: %.3f" % (naive_predictor_f1score))

         Naive predictor accuracy: 0.471
         Naive predictor f1-score: 0.640
```

Construct Logistic Regression model

- Perform random search of model hyperparameter space
  - Hyperparameter Tuning the Random Forest in Python
- Results suggest that a logistic regression model's accuracy and f1-score is better than the naive predictor
  - **Accuracy**
    - Naive predictor: 0.471
    - Logistic regression: 0.722
  - **F1-score**
    - Naive predictor: 0.640
    - Logistic regression: 0.716

Evaluate Logistic Regression Model Performance

```
In [80]: evaluate_model_performance(lr_random.best_estimator_,
                                    X_train,
                                    y_train)

         LogisticRegression model accuracy: 0.723
         LogisticRegression model f1-score: 0.718

Out[80]: (0.7232408325074331, 0.7175507187177841)
```

Construct Random Forest Model

- Perform random search of model hyperparameter space
- Results suggest that a random forest model's accuracy and f1-score is better than the naive predictor
    - **Accuracy**
        - Naive predictor: 0.471
        - Random forest: 0.742
    - **F1-score**
        - Naive predictor: 0.640
        - Random forest: 0.735

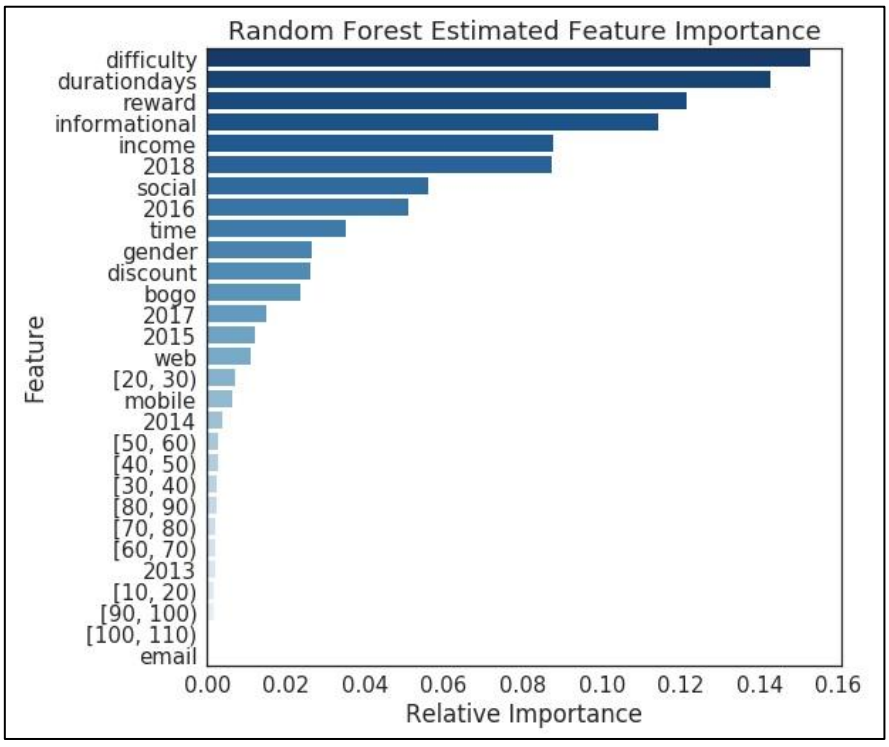
Evalaute Random Forest Model Performance

```
In [82]: evaluate_model_performance(rf_random.best_estimator_,
                                   X_train,
                                   y_train)

         RandomForestClassifier model accuracy: 0.748
         RandomForestClassifier model f1-score: 0.742

Out[82]: (0.7477128916673019, 0.7415304707789037)
```

Plot Estimated Feature Importance



Construct a Gradient Boosting classifier

- Perform random search of model hyperparameter space
- Results suggest that a gradient boosting model's accuracy and f1-score is better than the naive predictor
    - **Accuracy**
        - Naive predictor: 0.471
        - Gradient boosting: 0.736
    - **F1-score**
        - Naive predictor: 0.640
        - Gradient boosting: 0.725

Evaluate Gradient Boosting Model Performance

```
In [86]: evaluate_model_performance(gb_random.best_estimator_,
                                   X_train,
                                   y_train)

         GradientBoostingClassifier model accuracy: 0.743
         GradientBoostingClassifier model f1-score: 0.734

Out[86]: (0.743443622779599, 0.7341247111339352)
```

Tune the best model

- Model ranking based on training data accuracy
    1. ***RandomForestClassifier model accuracy: 0.742***
    2. ***GradientBoostingClassifier model accuracy: 0.736***
    3. ***LogisticRegression model accuracy: 0.722***
    4. ***Naive predictor accuracy: 0.471***
- Model ranking based on training data F1-score
    1. ***RandomForestClassifier model f1-score: 0.735***
    2. ***GradientBoostingClassifier model f1-score: 0.725***
    3. ***LogisticRegression model f1-score: 0.716***
    4. ***Naive predictor f1-score: 0.640***
- Results suggest that the random forest model has the best training data accuracy and F1-score

Bias and variance are two characteristics of a machine learning model. Bias refers to inherent model assumptions regarding the decision boundary between different classes. On the other hand, variance refers a model's sensitivity to

changes in its inputs. A logistic regression model constructs a linear decision boundary to separate successful and unsuccessful offers. However, my exploratory analysis of customer demographics for each offer suggests that this decision boundary will be non-linear. Therefore, an ensemble method like random forest or gradient boosting should perform better.

Both random forest and gradient boosting models are a combination of multiple decision trees. A random forest classifier randomly samples the training data with replacement to construct a set of decision trees that are combined using majority voting. In contrast, gradient boosting iteratively constructs a set of decision trees with the goal of reducing the number of misclassified training data samples from the previous iteration. A consequence of these model construction strategies is that the depth of decision trees generated during random forest model training are typically greater than gradient boosting weak learner depth to minimize model variance. Typically, gradient boosting performs better than a random forest classifier. However, gradient boosting may overfit the training data and requires additional effort to tune. A random forest classifier is less prone to overfitting because it constructs decision trees from random training data samples. Also, a random forest classifier's hyperparameters are easier to optimize

Out[87]:

	classifiertype	accuracy	f1score
0	randomforest	0.747713	0.741530
1	gradientboosting	0.743444	0.734125
2	logisticregression	0.723241	0.717551
3	naivepredictor	0.470916	0.640303

Refine Best Model

- Refine model hyperparameter space

Evaluate Test Data Performance

```
In [91]: evaluate_model_performance(best_clf,
                                   X_test,
                                   y_test)

         RandomForestClassifier model accuracy: 0.743
         RandomForestClassifier model f1-score: 0.733

Out[91]: (0.7433102081268583, 0.733222407099279)
```

V. Conclusion

- Chose to solve was to build a model that predicts whether a customer will respond to an offer.
- Strategy for solving this problem has four steps. First,
- Combined offer portfolio, customer profile, and transaction data.
- Assessed the accuracy and f1-score of a naive model that assumes all offers were successful.
- Compared the performance of logistic regression, random forest, and gradient boosting models.
- Analysis suggests that a random forest model has the best training data accuracy and f1-score.
- Refined random forest model hyperparameters using a grid search.
- Analysis suggests that the resulting random forest model has an training data accuracy of 0.753 and an f1-score of 0.746.

The test data set accuracy of 0.736 and F1-score of 0.727 suggests that the random forest model I constructed did not overfit the training data.

"Feature importance" refers to a numerical value that describes a feature's contribution to building a model that maximizes its evaluation metric.

A random forest classifier is an example of a model that estimates feature importance during training. My analysis of the Starbucks Capstone Challenge customer offer effectiveness training data suggests that the top five features based on their importance are:

1. Offer difficulty (how much money a customer must spend to complete an offer)
2. Offer duration
3. Offer reward
4. Customer income
5. Whether a customer created an account on the Starbucks rewards mobile application in 2018

Since the top three features are associated with a customer offer,

- It may be possible to improve the performance of a random forest model by creating features that describe an offer's success rate as a function of offer difficulty, duration, and reward.
- These additional features should provide a random forest classifier the opportunity to construct a better decision boundary that separates successful and unsuccessful customer offers.