MACHINE LEARNING ENGINEER
NANODEGREE

2020

STARBUCKS CAPSTONE CHALLENGE

PROJECT REPORT

KAUSTUBH KATKAR

Contents

1. PROJECT DEFINITION:

Domain:

Starbucks Corporation operates as a roaster, marketer, and retailer of specialty coffee worldwide. The company operates in three segments: Americas; International; and Channel Development [1]. The brand of Starbucks has grown to such a scale that it attracts customers globally in large numbers making it imperative to develop customer retention strategies along with customer acquisition.

The cost of these operations is usually high for a franchise of this scale which is why discovering efficient means of retaining customers and identifying their interests accurately will have a significant impact on profitability [2].

In this project, Starbucks Corporation has provided a simulated version of the data their Starbucks application collects to develop insights from. Exploring the data and identifying trends will identify the customer base, the unsuccessful strategies as well as reinforce good strategies. The task is to be able to clean the data, explore the data and build a model that will successfully predict whether or not a customer will respond to an offer by completing it.

Problem Statement:

To explore, analyse and identify the best offer that will ensure prolonged customer engagement with the Starbucks brand. These tasks will be performed on simulated datasets that mimics the data of the actual Starbucks rewards application. The final model will be able to determine whether an offer will generate a response or not from the customer.

Datasets:

The datasets are provided by Starbucks as a Capstone Challenge for Udacity's Machine Learning Engineer NanoDegree. The entire data is distributed over 3 datasets:

1. *portfolio.json* : This dataset contains information specific to each offer type. It has 10 records with 6 attributes and each record identifies a particular offer.

field	datatype	description
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	eg. mobile, email, etc.

2. *profile.json* :
   This dataset consists the profiles of all the members of the app. This dataset contains 17000 records. Further exploration reveals that certain attributes are missing which will require cleaning and wrangling to make the dataset consistent. Also, attributes such as *became_member_on* will require feature engineering to be relevant and useful when training the model.

field	datatype	description
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

3. *transcript.json* :
   All the transactional activity within the application is maintained in this dataset. The samples here are records of viewing, receiving and completing an offer along with general purchase/transaction. Feature engineering is again an important phase to extract useful information from this dataset. It contains 306534 records.

field	datatype	description
event	str	record description
person	str	customer id
time	int	time in hours since start of test. The data begins at time t=0
value	dict of strings {str}	either an offer id or transaction amount depending on the record

Metrics:

I will evaluate the performance of our model based on the accuracy of its predictions. To determine accuracy of the model I first have to define the target variable that I need to consider to establish if a prediction is accurate. As the data does not contain the information of the amount spent during an offer or which specific offers were viewed and completed, there is an emphasis on the importance of feature engineering phase for this case. Through feature engineering I derived attributes that can train the model effectively as well as the target attribute which is the label of an offer being *successful*.

Apart from accuracy, the recall and precision of our models are also considered when selecting the final model. As recall is the ration of true positives to all actual positives { $\frac{True\ Positives}{True\ Positives + False\ Negatives}$ } and precision is true positives to all predicted positives { $\frac{True\ Positives}{True\ positives+False\ Positives}$ } [3] , these metrics provide us with knowledge of where the models might require tuning to improve on their performance.

Through *classification_report* from *sklearn.metrics* library I can describe these metrics on each model. Then, using the f-1 score and accuracy as reference final model for prediction is selected. F-1 score is another means of determining the model's accuracy by taking the harmonic mean of precision and recall [4].

2. DATA ANALYSIS:

Data Exploration:

Exploring portfolio:

On viewing the structure of portfolio dataset, the following observations were made:

- The channels for every offer type are grouped in lists.
- offer_type attribute has 3 unique labels.
- difficulty attribute has 5 unique labels but are integers that could be formatted
- offer duration attribute is in days.
- offer id is a string of characters and integers.

Cleaning portfolio:

Firstly, one-hot encode for every channel in the channel list. As this is a dataset with only 10 records, channels can be determined manually. For lager dataset of this nature I would require to check unique values for channel lists of each record. Again, conduct one-hot encoding over offer_type as well. Then normalize the difficulty attribute by dividing it will the max value of this attribute. This ensures that difficulty is now between 0 and 1. Convert the offer duration for days to hours (Multiply by 24). Create a new attribute offer_label and label it from 1 to 10 i.e length of dataset. Once features are normalized, redundant or unwanted columns like offer_type and channels can be dropped.

Exploring profile:

Notable observations about the profile dataset:

- There are several '*None*' entries in the cells for gender attribute.
- Similarly, there are several '*NaN*' entries in the cells of income attribute.
- The records with '*None*' gender and "*NaN*' income coincide and the age attribute for each of these records is 118.
- became_member_on attributes contain 8 digits long integers which are a combination of year, month and day in YYYYMMDD format.
- Customer's id is a string of characters and integers.

Cleaning profile:

Identify and drop all the records with 'NaN' income and 'None' gender. If a single feature value is missing replace it with the mean of that feature. In our case all the missing values are coincidental and hence it stands to reason that these are outliers and can be dropped. There are 2175 such records and after dropping these values I still retain 87.3% of our dataset. Now distribute the became_member_on attribute to member_year, member_month and member_day attributes. Drop the became_member_on attribute. There is no need to create labels for customer id as once after all the customer's demographic information will be merged in a combined set the customer id will not provide any useful information. In that even customer id will be dropped from the combined dataset.

Exploring transcript:

Notable observations about the transcript dataset:

- Contains events within the app and the type of event is specified in the event attribute.
- time feature contains time of the event's occurrence in hours.
- The feature 'value' holds dictionaries of each sample and there are a total of 4 keys across all dictionaries. These keys are *offer id, reward, amount* and *offer_id*. The offer id key is used in the dataset for 'offer received', and 'offer viewed' events whereas the 'offer_id' key is used for 'offer completed' events.
- The customer id is contained in the 'person' attribute

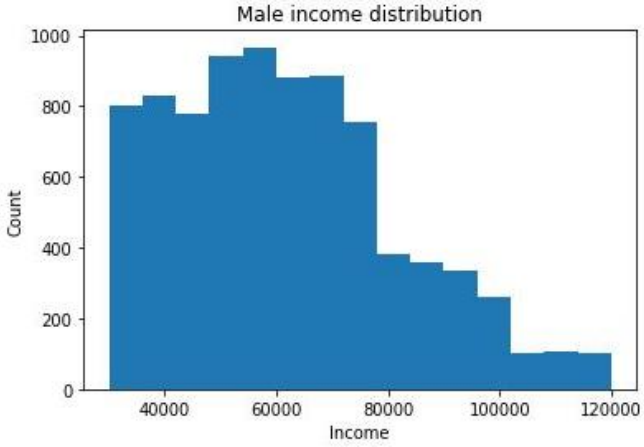
Cleaning transcript:

Either create event_code for each event (integers 1 to 4) or one-hot code event attribute. Transform the dictionary keys from the value attribute to columns and dictionary values to cell entries. At this stage offer id and offer_id columns are redundant as their event already provides sufficient information, hence, merge their values and drop 'offer_id' column. At this stage there are many missing values within the dataFrame for reward, amount and offer id columns (when event = "transaction"). As missing values in rewards and amount indicates that the transaction did not involve expenditure, I can replace these values with integer 0.

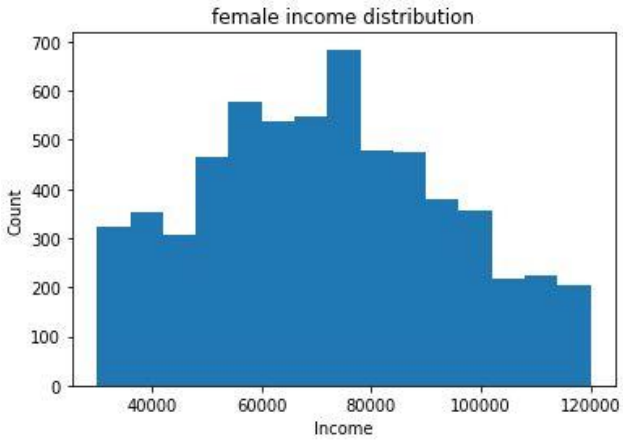
Data Visualization:

profile:

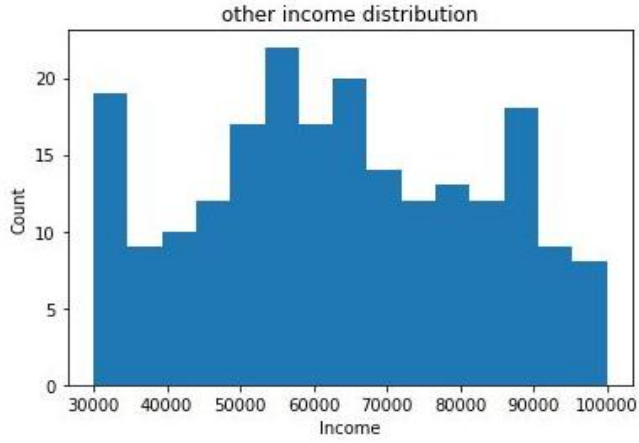
Distribution of income for males:



Distribution of income for females:



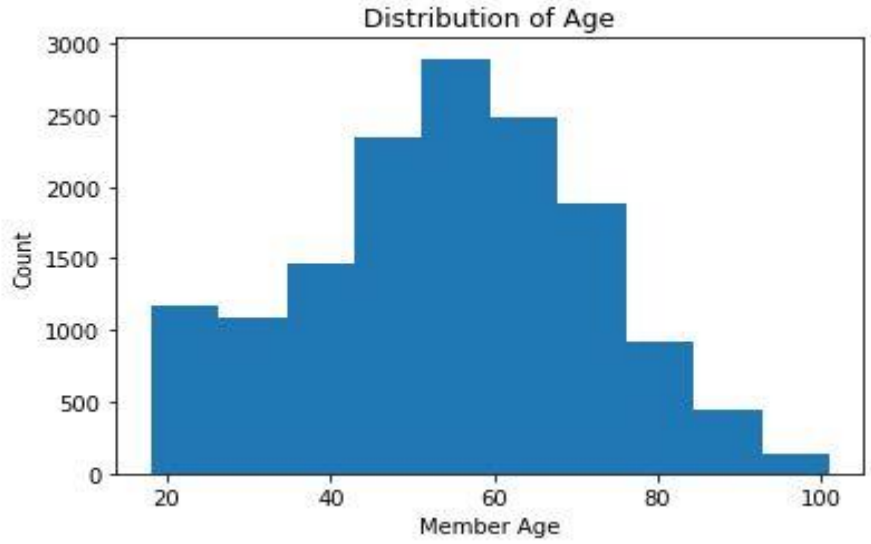
Distribution of income for others:



Findings:

- Most of the male customers have income between 57000 and 75000.
- The highest count observed for females is between 66000 and 75000
- More female customers earn over 102,000 as compared to male and other customers.
- The income distribution of the other gender is over a considerably small set of samples and hence observations made cannot be considered decisive.

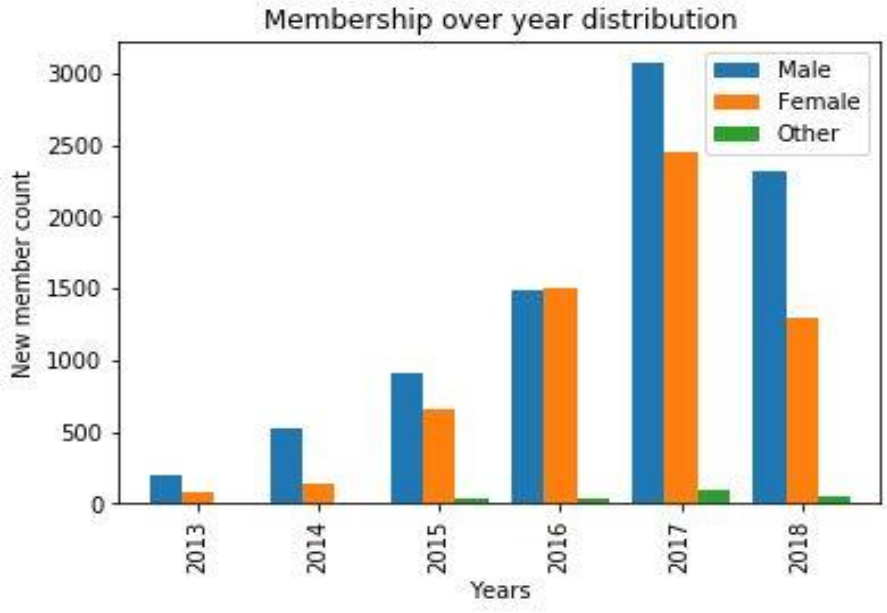
Distribution by age:



Findings:

- Majority of the app users lie in the age range of 45 and 70

Distribution of membership over the years

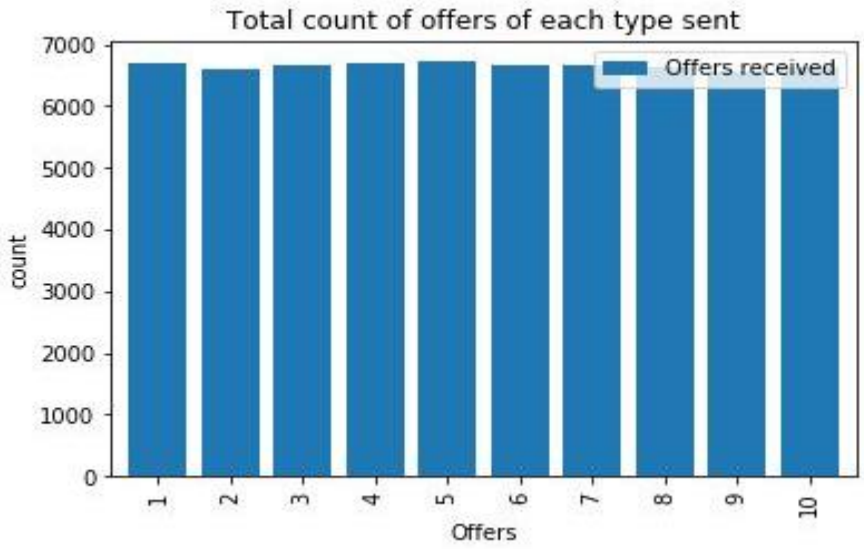


Findings:

- The year 2017 had the most members joining the app in all gender categories.
- 2016 had more females register to the app.
- The trend of membership is almost similar among all genders.

transcript:

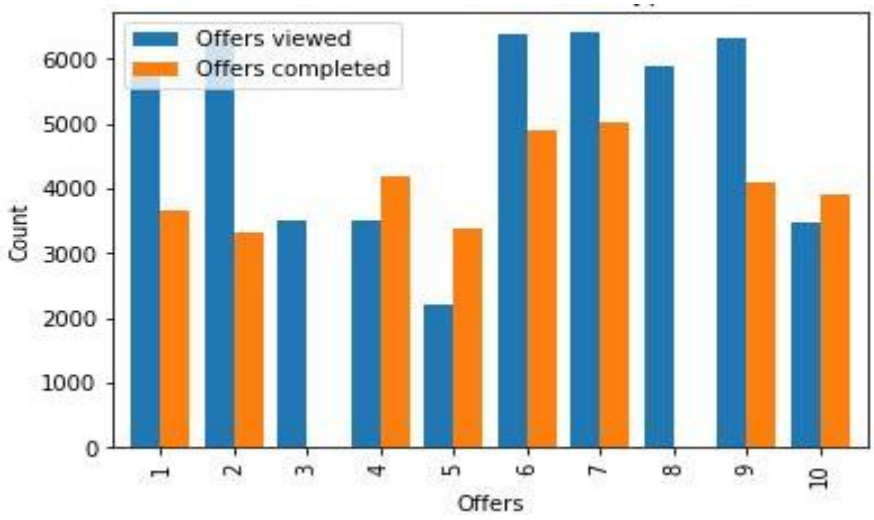
Offers distributed:



Findings:

- Each type of offer was distributed to almost equal number of customers.

Offers viewed vs completed:



Findings:

- Offers 3 and 8 are not influencing any customers to make a purchase. These are informationals.
- Offers 4, 5 and 10 have more completions than views for the offers. This implies that the usual customer spending has led to completion in these cases.
- The remaining offers have a natural trend where the completed offers are lesser in comparison to the views

3. METHODOLOGY

Feature Engineering:

Now I will create a combined dataset to be used for training the machine learning models.

Firstly, drop all the transcript records that belong to customers which are not present in the profile dataset. These should be the records belonging to the 2175 outliers that were dropped from the profile dataset. This makes the transcript dataset consistent with customer profile. Append the offer_labels from portfolio to transcript dataset. I can now drop the id attribute in portfolio to maintain a clean portfolio data frame.

Next, I will create a data frame containing two new attributes; namely 'successful' and 'total_amount'. The 'successful' attribute will determine whether an offer was received, viewed and completed and it will be set to 1 on completion. It will be 0 otherwise. This is the target attribute in our case that I will make predictions about. The 'total_amount' attribute will consist of the amount spent by the customer after receiving the offer and before the end of its duration.

To achieve this, I will divide transcript events into two data frames: an offer data frame and a transaction data frame. The offer data frame will contain all the offer received, offer viewed and offer completed events. The transaction data frame will contain all the transaction events. Then, for each unique customer fetch the customer offers from the offer set and customer transaction from the transaction set. If an offer is viewed and completed then create a 'successful' label with value 1. Otherwise create 'successful' label with value 0. In the same iteration get and sum the amounts for each transaction within the offer duration and add it in the 'total_amount' attribute. Repeat this for all unique customers.

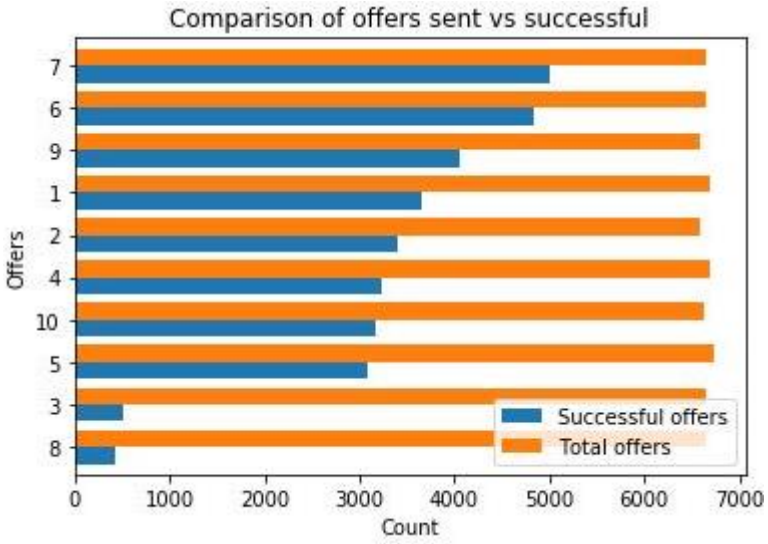
Note: The attribute successful determines if an offer received a response from the customer. It only labels 1 for offers that were viewed and completed. If an offer was just received and completed but not viewed, it will label this situation as 0. This is because it implies that the offer was completed by general spending habits and the offer did not influence the customer's decision to make purchases.

Once a data frame with 'successful' and 'total_amount' is created, merge the relevant offer attributes to it from the portfolio dataset. This merge is conducted on the 'offer_label' feature. After this, merge the relevant demographic features from the profile dataset to the new data frame on the customer 'id' feature. Out of the membership date attributes, I have only chosen the membership year because annual trends are usually more influential for the customer base in general context.

The combined data frame created still has the customer 'id' attribute which will not be useful for the learning models. As the relevant demographic features are already appended, drop the customer ids. The combined data frame now contains all the important attributes for our learning models and has a total of 66,501 records with 18 features. Save this data set so that it could be used in the '*Modeling.ipynb'* Jupyter notebook for training and prediction.

Now I can extract trends in offer success, amount spent and ages responding to offers from the combined data frame.

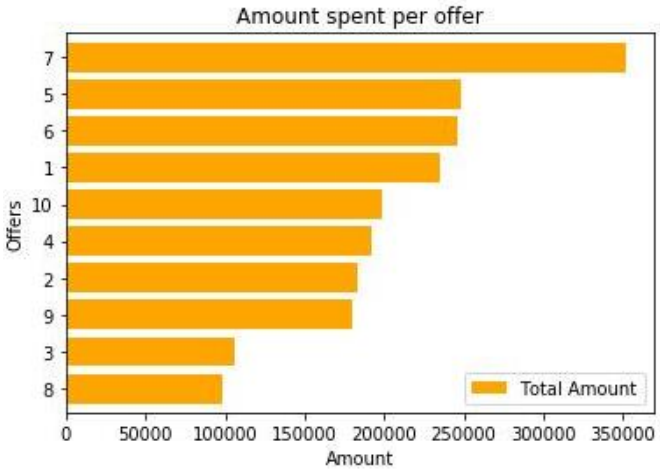
Offer Success:



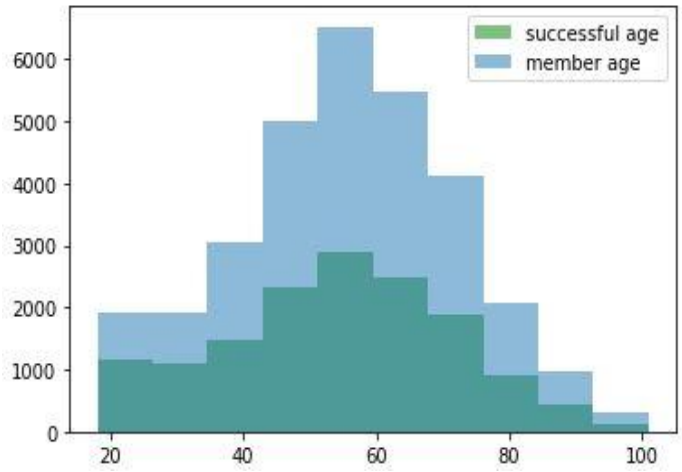
Findings:

- Offers 6 and 7 have the highest success rate. These are discount offers.
- The informationals i.e. offers 3 and 8 have the least conversion.
- Offer 9 is a *BOGO* offer and has the least difficulty amongst offers that are not informationals. It is the 3rd most popular offer.

Total Amount Spent:



Age Trend:



Findings

- The Offer 7 is most impactful as it has the highest success rate and most amount spent.
- Although Offer 6 has been successful more number of times in comparison to Offer 5, Offer 5 has more amount spent on it. This is interesting because in terms of conversion rate, Offer 5 has performance in the bottom 3 but total business attracted by it is still the 2nd best.
- There are no interesting insights with the success rate over ae distribution. The success rate is evenly distributed with the ages of the customer base.
-

Implementation:

Split the combined dataset into train, test and validation sets. This can be done using train_test_split function from the *sklearn.model_selection [10]* library.

Divide 67% (2/3) of the dataset for training and 33% (1/3) of the dataset for test in the first split. Then split 10% of the test set and assign it to validation. Start with training the benchmark model to establish threshold for model performance. In our case I are considering the LogisticRegression model; *from sklearn.linear_model*; as benchmark.

Import accuracy_score, classification_report and f1_score from sklearn.metrics to evaluate the model. The classification_reports for each model can be found in the '*Modeling.ipynb'* notebook. As the model selection is determined using F1 score and accuracy metrics [9], I will report these metrics.

Models trained:

Models	**Library**
LogisticRegression	*sklearn.linear_model [5]*
GradientBoostingClassifier	*sklearn.ensemble [6]*
DecisionTreeClassifier	*sklearn.tree [7]*
RandomForestClassifier	*sklearn.ensemble [8]*

Evaluation:

After training and validating all the models I get the following performance metrics:

	accuracy	f1_score
Logistic Regression	0.874715	0.856247
GradientBoosting	0.912984	0.906050
Decision Tree	0.882005	0.868193
Random Forest	0.912984	0.904548

Insights:

All the learning models perform well on the validation set. The Gradient Boosting Classifer, Decision Tree Classifier and Random Forest Classifier all perform better than the Logistic Regression classifier which is the benchmark for our models.

The performance of both Random Forest Classifier and GradientBoosting Classifier is very identical. Although the F1 score of the GradientBoosting model is more but the difference is not substantial. Hence, either model can be selected for predicting the test set labels.

Model Selection:

I determined Random Forest Classifier to be the best model for the following reasons:

- It has good performance on the validation set.
- The decision trees within the Random forest can be exported (using ghraphviz library) and is useful for understanding the rationale leading to a good performance. This will help optimize business decisions for Starbucks.
- GradientBoosting Classifier is harder to hypertune to optimize performance.

Refining/Hyperparameter tuning:

To hypertune the RandomForest Classifier use RandomizedSerchCV from *sklearn.model_selection* library [11]. Tune the model parameters' dictionary with n_estimators, max_features and min_sample_split parameters with distributions to be tested for hypertuning. As I do not know what combination will work best, I have used distributions over each parameter instead of selecting specific integers. Each configuration of the model will be trained and tested and best parameters will be retained as the final configuration. The total number of models trained will be number of iterations times the number of cross validations. In our training I have input 100 iterations and 5 cross validations so I will be training 500 models in this process.

Final Best configuration found by random search:

```
Best set of hyperparameters:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 0.34082380566905934,
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 0.012776363022143807,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 98,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 1,
 'verbose': 0,
 'warm_start': False}
```

Although I only tuned 3 parameters, any of the above parameters can be tuned and tested to check for performance improvement.

4. RESULTS:

Model Evaluation:

After training the model with optimized parameter, predict the target for the test set. I observed the following performance:

```
RandomForest Classifier accuracy:  0.9117006733836261
RandomForest Classifier classification report :
              precision    recall  f1-score   support

           0       0.94      0.89      0.91     10433
           1       0.88      0.94      0.91      9318

    accuracy                           0.91     19751
   macro avg       0.91      0.91      0.91     19751
weighted avg       0.91      0.91      0.91     19751
```

	accuracy	f1_score
Random Forest	0.911701	0.909223

There is a fractional increase in the F1 score after hypertuning but it is not substantial. The evaluation infers that this model is able to maintain its accuracy over new unseen data. The performance of the model is appropriately high. Hence, this model will perform well for unseen samples in the application.

5. CONCLUSION:

Reflection:

In this project I successful created a Machine Learning model to tackle the challenge of identifying the conversion of an offer successfully. I successful wrangled and explored the data to understand the distributions and trends of the customer base. I was able to observe the patterns of offer distribution by Starbucks.

I was able to evaluate our model against the benchmark model and justify the selection of the best model. Then the models hyperparametes was tuned and performance was evaluated. The model performed well over unseen data with an accuracy of 91% and F1 score of 91%.

Future Scope:

This application can be further developed on in the following directions:

- A model can be generated to identify the days and months when the distributed offers are most successful. This will require relevant data of events over the years within the region which can include festivals, sporting events, college and work schedules.
- With some feature engineering I could also be able to develop a model for recommending offers to a customer. For this I will have to establish a threshold value and test the customer interest of every offer and recommend the highest one above the threshold.

References:

[1] https://finance.yahoo.com/quote/SBUX/profile/

[2] https://www.dcrstrategies.com/customer-incentives/5-reasons-customer-retention-business/

[3] https://en.wikipedia.org/wiki/Precision_and_recall

[4] https://en.wikipedia.org/wiki/F1_score

[5] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[6] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[7] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[9] https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

[10] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[11] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html