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 classification problem and would be solved by using supervised machine learning algorithms.

Metrics

Since, our combined dataset created from the Starbucks datasets (offer data, customer demographic data and transaction data) is balanced in terms of distribution of target class (customers whom respond to an offer represented by class 1 and whom do not respond to an offer represented by class 0), performance metrics like accuracy, precision, recall, and f1_score are good measure for evaluating a model.

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.*

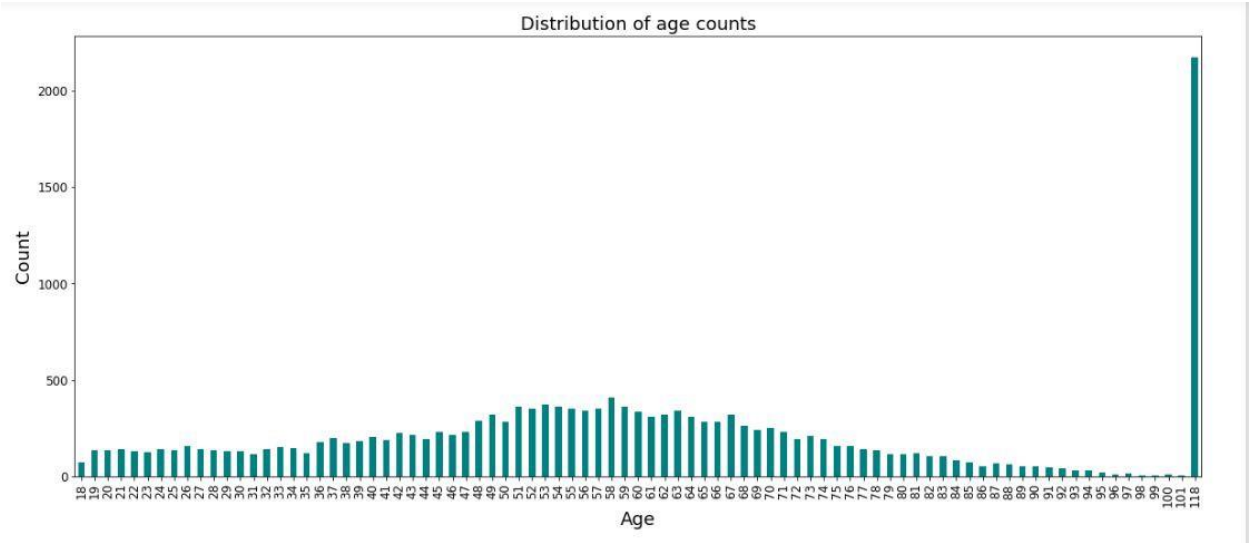
```
# Print portfolio dataframe
portfolio
```

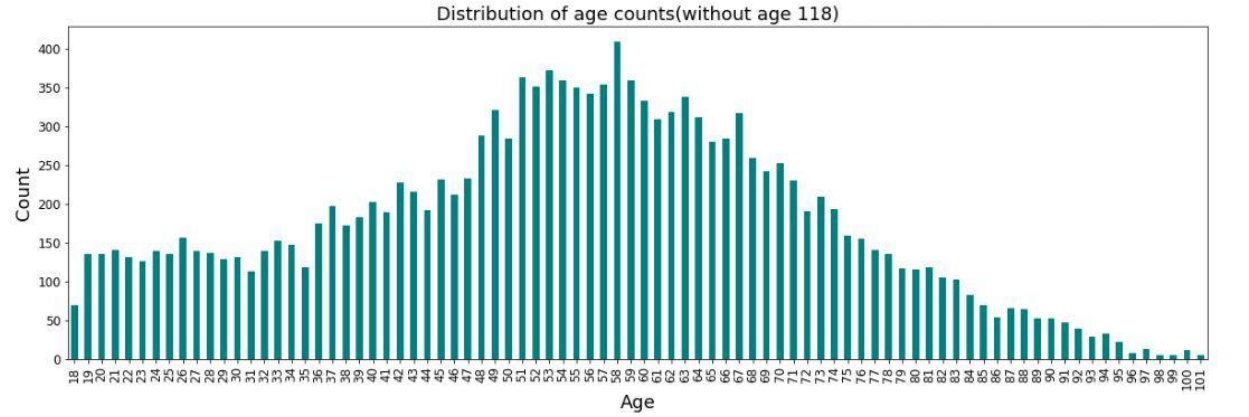
	offer_id	difficulty	duration	reward	bogo	discount	informational	web	email	mobile	social
0	ae264e3637204a6fb9bb56bc8210ddfd	10	168	10	1	0	0	0	1	1	1
1	4d5c57ea9a6940dd891ad53e9dbe8da0	10	120	10	1	0	0	1	1	1	1
2	3f207df678b143eea3cee63160fa8bed	0	96	0	0	0	1	1	1	1	0
3	9b98b8c7a33c4b65b9aebfe6a799e6d9	5	168	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

profile data summary

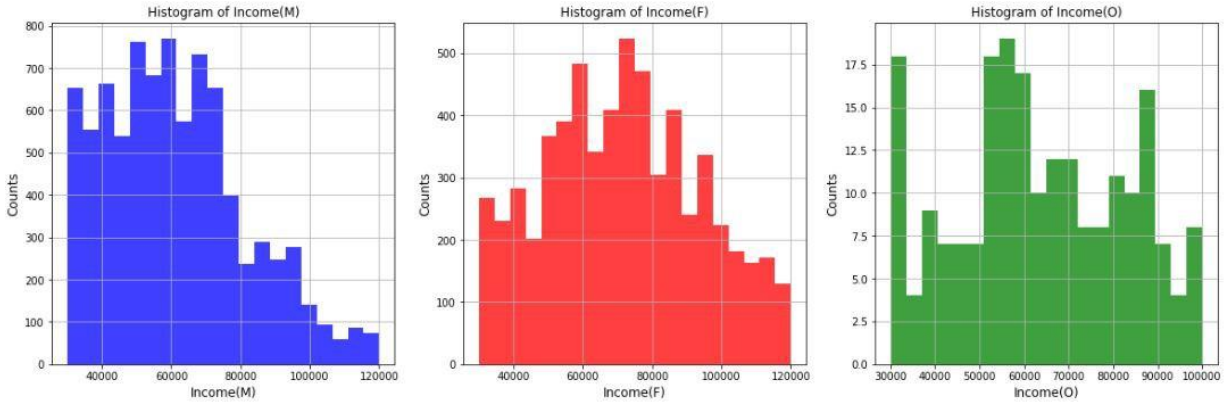
5. *There are 2175 observations with age 118 having gender as None and income as nan . Value 118 in age feature represents null value and therefore the observations of customers with age 118 were removed.*

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'.*

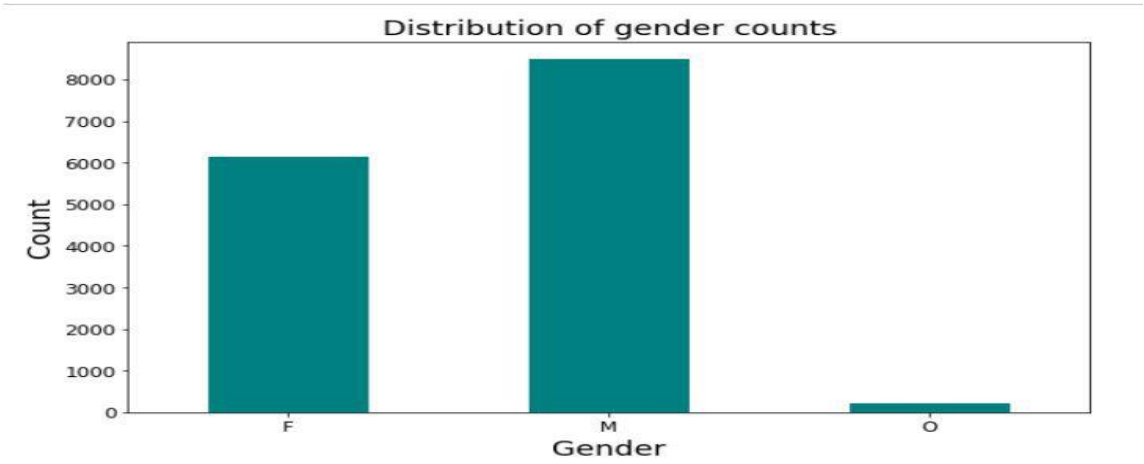




7. 'Income' feature was binned into features 'income_30ths', 'income_40ths', 'income_50ths', 'income_60ths', 'income_70ths', 'income_80ths', 'income_90ths', 'income_100ths', 'income_110ths' and 'income_120ths'.



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



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.*

```
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

```
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

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

3. *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.*

4. *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.*

```
# 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
```

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

3. *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.*

```
# 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
```

4. *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

```
# 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
```

1. *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.

2. *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.*

3. *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. *Before training Classifier's like Logistic Regression, Random Forest, Adaboost & GradientBoosting, combined data was split into training and test data.*

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

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

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

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

2. *From above, we can observe that our training & test set is also balanced in terms of distribution of target class.*
3. *After combined data was split into train and test data, features like 'difficulty', 'duration', 'reward', 'membership_tenure', 'total_amount' were scaled using MinMaxScaler.*
4. *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 produced better f1_score than baseline model's f1_score.*

```
# 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...

5. *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.*

```
# 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
**************************************
```

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.*

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

```
(16626, 52)
```

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

```
(16626, 1)
```

```
# 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,)
```

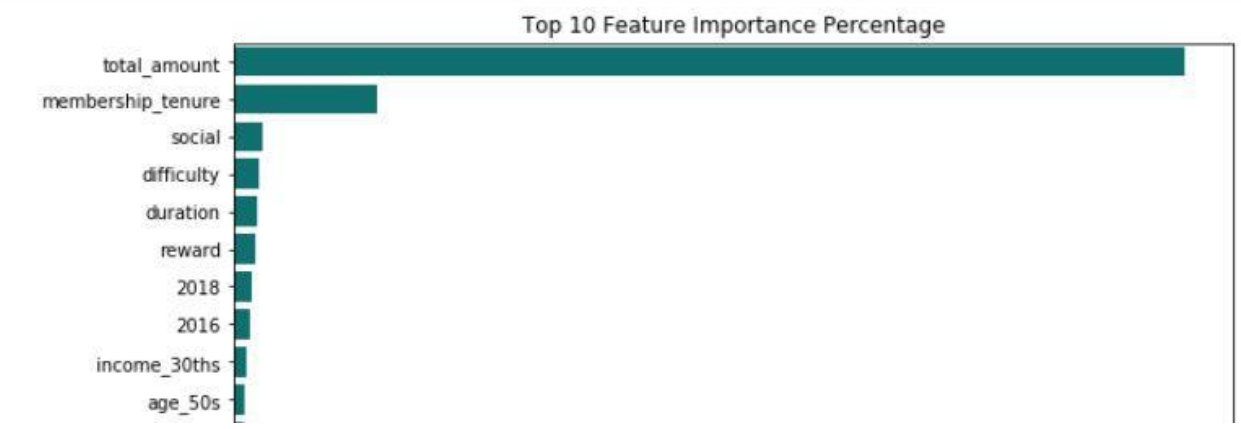
2. *Our best estimator produced f1_score of 0.9336 on test data, which is quite good.*

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.*

```
# 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



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*

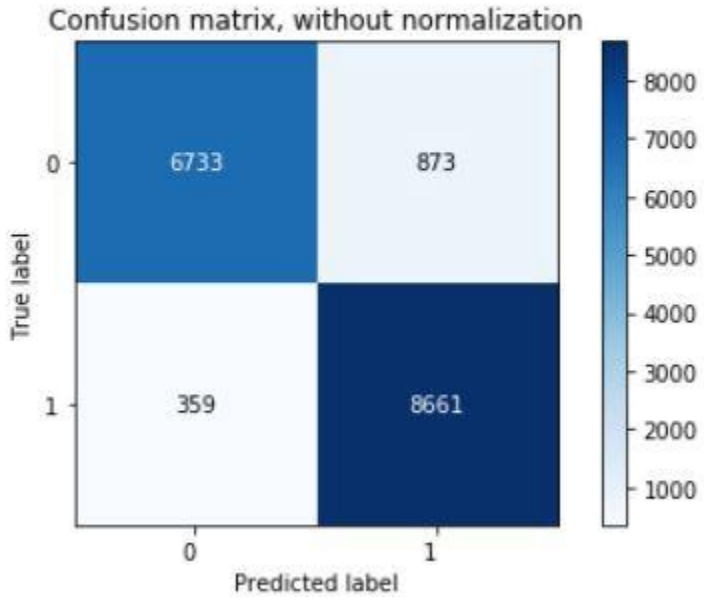
than the benchmark model or baseline model or naïve model reported earlier having f1_score of 0.6996.

V. Conclusion

Free-Form Visualization

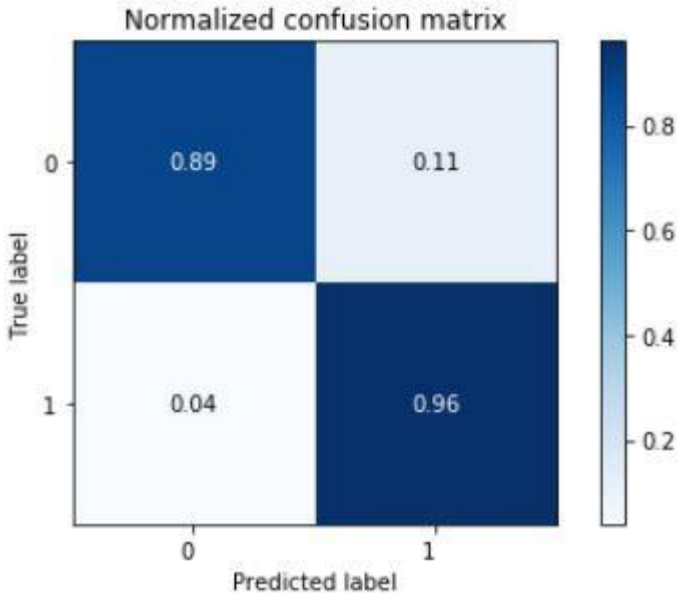
1. *Plot confusion matrix without normalization for predictions on the test set.*

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



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

```
Normalized confusion matrix
[[ 0.89  0.11]
 [ 0.04  0.96]]
```



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 transaction 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 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. *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.