# Machine Learning Engineer Nanodegree

## Starbucks Capstone Project Report

Deepa Balakrishnan

January 20th, 2022

## Contents

## I.Definition

### Project Overview

**Starbucks** is one of the most well-known companies in the world: a coffeehouse chain with more than 30 thousand stores all over the world. It strives to give their customers always the best service and the best experience. As a side feature, Starbucks offers their free app to make orders online, predict the waiting time and receive special offers. This app also offers promotions for bonus points to these users. The promotional offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). This project is focused on tailoring the customer behavior and responses while using the Starbucks mobile app. To avoid churn rate and trigger users to buy Starbucks products, it is important to know which offer should be sent to specific users.
To study about application of machine learning to predict customer churn, I used the reference: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3835039.

### Problem Statement

The problem we are trying to solve here is, given a promotional offer and the users demographics, how likely is it that the user will be prompted by the offer. Hence this is a **classification problem**. We will be predicting the probability with which the customer will take up an offer or not. Hence it will help the Starbucks to minimize churn rate.

## Metrics

The performance of our model is evaluated in two aspects – classification problem metrics and business impact metrics

### Metrics used for evaluating classification problem:

To evaluate the quality of approach and determine the model that produces best results, I will use classification metrics such as accuracy, **F1-score, recall, precision**, **ROC AUC**, and cohen-kappa. During the data exploration stage, it was noticed that the dataset is imbalanced – with majority of customers not responding to offers. Hence it is not useful to use accuracy as a metric since model will simply predict that the customers will not respond to an offer. F1-score metric can be interpreted as the weighted average of precision and recall. The traditional or balanced F-score (F1 score) is the harmonic mean of precision and recall, where an F1 score reaches its best value at 100 and worst at 0. Our goal here is to obtain a **maximum F1-score**

### Metrics used for evaluating business impact:

Conversion rate is defined as the percentage of users who have completed an offer. Our goal is to predict the probability of a user taking up an offer, hence the business shows success when the **conversion rate** increases. The project development is successful when there is a **lift** of conversion rate when using our model. Hence conversion rate and lift are the two metrics evaluated in this aspect.

# II.Analysis

## Data Exploration

There are 3 available data sources as mentioned below:

1. The first one is **portfolio**: it contains list of all available offers to propose to the customer. Each offer can be a *discount*, a *BOGO (Buy One Get One)* or *Informational* (no real offer), and we've got the details about discount, reward, and duration of the offer.
2. The next data source is **profile**, the list of all customers that interacted with the app. For each profile, the dataset contains some personal information like gender, age, and income.
3. Finally, there is the **transcript** dataset: it has the list of all actions on the app relative to special offers, plus all the customer's transactions. For each record, we've got a dictionary of metadata, like offer_id and amount spent.

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

**portfolio.json**: Offers sent during 30-day test period (10 offers x 6 fields)

- reward: (numeric) money awarded for the amount spent
- channels: (list) web, email, mobile, social
- difficulty: (numeric) money required to be spent to receive reward
- duration: (numeric) time for offer to be open, in days
- offer_type: (string) bogo, discount, informational

- id: (string/hash)

**profile.json:** Rewards program users (17000 users x 5 fields)
- gender: (categorical) M, F, O, or null
- age: (numeric) missing value encoded as 118
- id: (string/hash)
- became_member_on: (date) format YYYYMMDD
- income: (numeric)

**transcript.json:** Event log (306648 events x 4 fields)
- person: (string/hash)
- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
    - offer id: (string/hash) not associated with any "transaction"
    - amount: (numeric) money spent in "transaction"
    - reward: (numeric) money gained from "offer completed"

- time: (numeric) hours after start of test

The portfolio.json contains offer_type column, which describes the types of offers that Starbucks is looking to send its customers:
1. BOGO (Buy-One-Get-One): A user needs to spend a certain amount to get a reward equal to that threshold amount.
2. Informational: There is no reward, but neither is there a requisite amount that the user is expected to spend.
3. Discount: A user gains a reward equal to a fraction of the amount spent.

Offers can be delivered via multiple channels.

Our goal here is to predict customers who have higher chances of responding to a given offer. Hence, we create a new base dataset, with target variable which helps us to build a classification model. The target variable definition is that, in each point of time, for each pair of person-offer, we flag it as:

- **1**: user responded to the promotional offer
- **0**: user did not respond to the promotional offer

This will be used as base dataset to which more features will be appended.

There are a few things to watch out for in this data set. Customers do not opt into the offers that they receive; in other words, a user can receive an offer, never actually view the offer, and still complete the offer. This scenario is not considered as a customer who respond to an offer, since the customer was not influenced by the offer because the customer never viewed the offer.

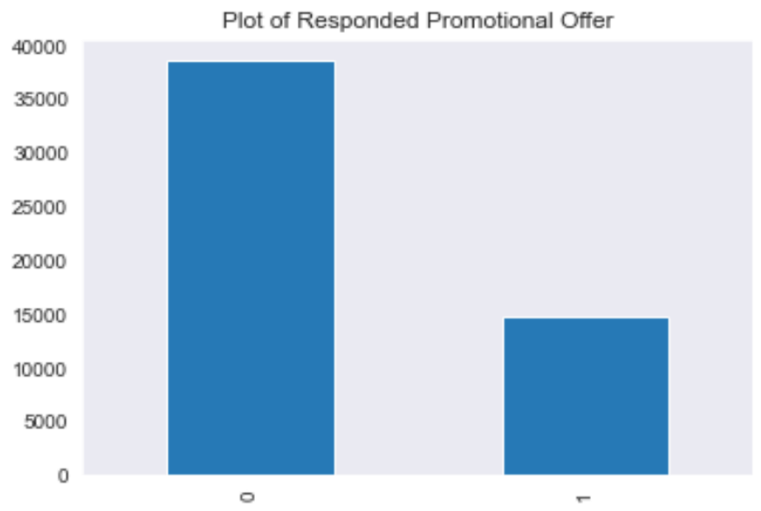
After converting features already available and performing feature engineering, the final dataset contains **76277 entries x 18 variables**. More information about features present in final dataset is as mentioned below:

- **person**: (string/hash) foreign key to profile dataset
- **offer**: (string/hash) foreign key to portfolio dataset
- **time_in_days_received**: (numeric) used to track time of the test period
- **responded_customer**: (numeric)label: 1= responded, 0 = not responded
- **gender**: (category) M, F, O or null
- **age**: (numeric) age or null (already transformed from 118)
- **income**: (numeric) salary
- **period_of_membership**: (numeric) how long customer is a member
- **reward**: (numeric) money awarded for the amount spent
- **difficulty**: (numeric) money required to be spent to get reward
- **duration**: (numeric) duration of the offer
- **offer_type**: (category) bogo, discount, information
- **web**: (numeric) 1 = channel, 0 = not a channel
- **email**: (numeric) 1 = channel, 0 = not a channel
- **mobile**: (numeric) 1 = channel, 0 = not a channel
- **social**: (numeric) 1 = channel, 0 = not a channel
- **average_purchase**: (numeric) in average, how much user has spent so far
- **frequency:** (numeric) how much purchases customer made

To prevent data leakage, we split the dataset into training and testing sets, so that only training data is studied.

## Exploratory Visualization

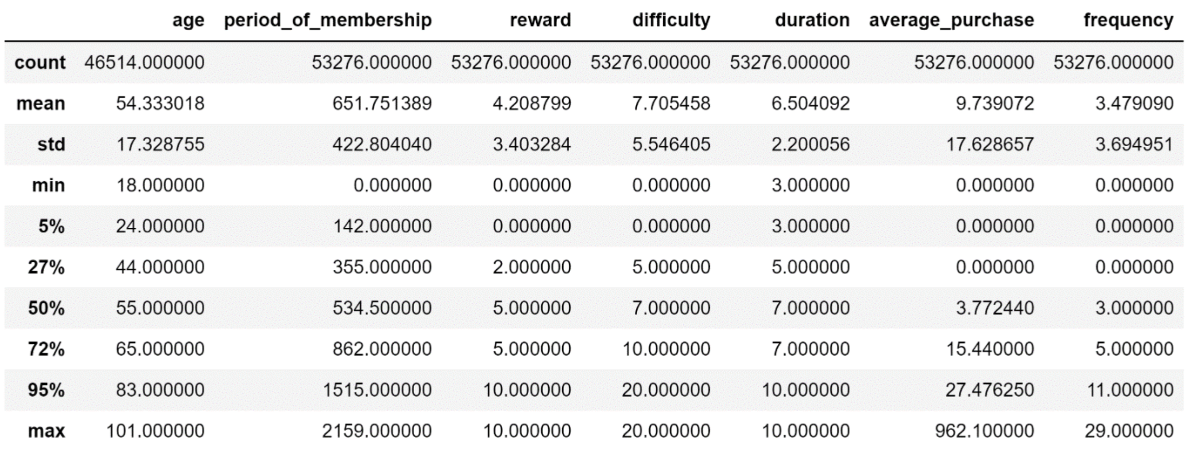
### Visualization for imbalanced dataset:



From the visualization above, it is possible to notice that we are dealing with an **imbalanced dataset**, since approximately 28% of the dataset responded to the promotional offer.

## Statistical Analysis performed on Numerical Features:

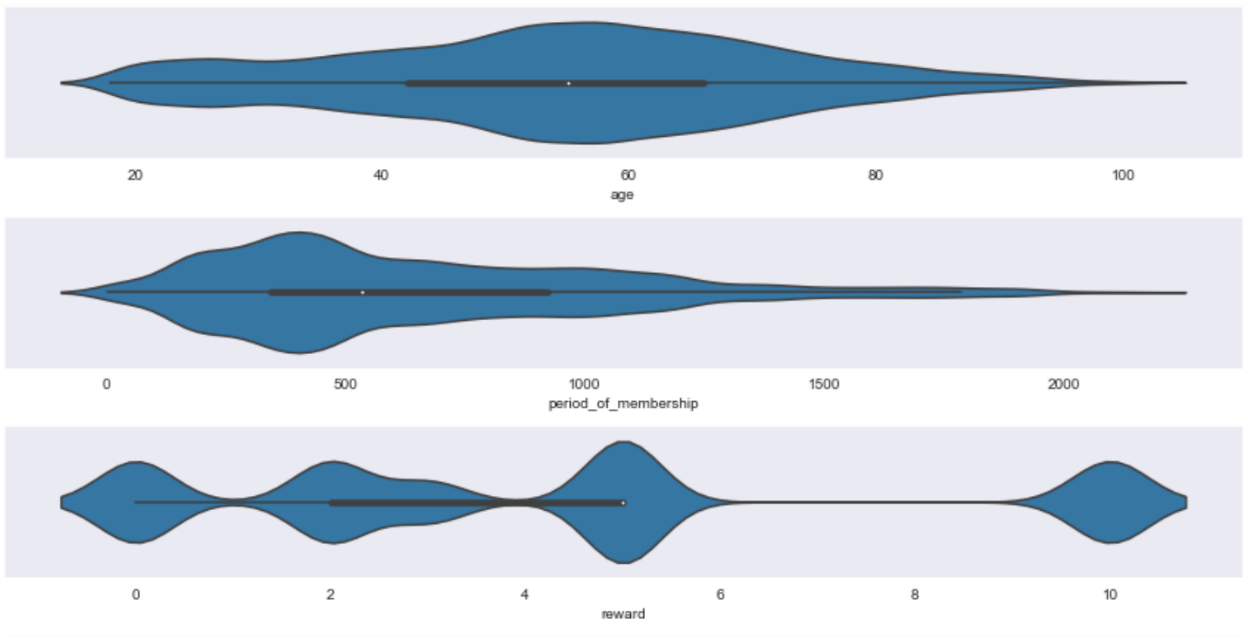
After analyzing the numerical features, we came to know that there are not many outliers except for the average_purchase and frequency fields. As the 95th percentile and maximum values are very different from each other for these two fields.

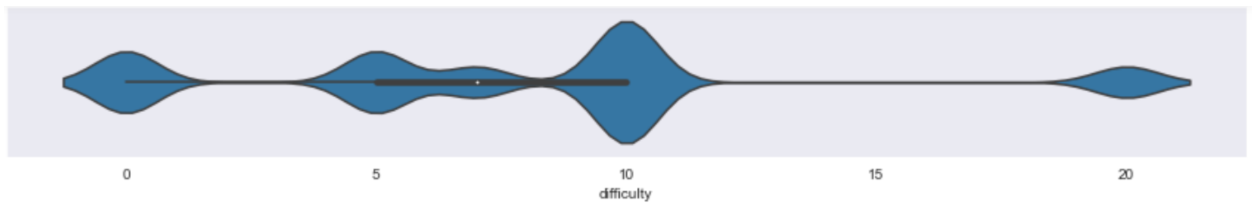


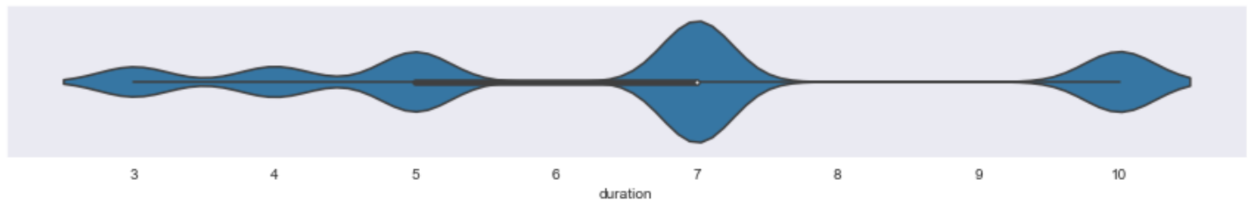
| | age | period_of_membership | reward | difficulty | duration | average_purchase | frequency |
|---|---|---|---|---|---|---|---|
| **count** | 46514.000000 | 53276.000000 | 53276.000000 | 53276.000000 | 53276.000000 | 53276.000000 | 53276.000000 |
| **mean** | 54.333018 | 651.751389 | 4.208799 | 7.705458 | 6.504092 | 9.739072 | 3.479090 |
| **std** | 17.328755 | 422.804040 | 3.403284 | 5.546405 | 2.200056 | 17.628657 | 3.694951 |
| **min** | 18.000000 | 0.000000 | 0.000000 | 0.000000 | 3.000000 | 0.000000 | 0.000000 |
| **5%** | 24.000000 | 142.000000 | 0.000000 | 0.000000 | 3.000000 | 0.000000 | 0.000000 |
| **27%** | 44.000000 | 355.000000 | 2.000000 | 5.000000 | 5.000000 | 0.000000 | 0.000000 |
| **50%** | 55.000000 | 534.500000 | 5.000000 | 7.000000 | 7.000000 | 3.772440 | 3.000000 |
| **72%** | 65.000000 | 862.000000 | 5.000000 | 10.000000 | 7.000000 | 15.440000 | 5.000000 |
| **95%** | 83.000000 | 1515.000000 | 10.000000 | 20.000000 | 10.000000 | 27.476250 | 11.000000 |
| **max** | 101.000000 | 2159.000000 | 10.000000 | 20.000000 | 10.000000 | 962.100000 | 29.000000 |

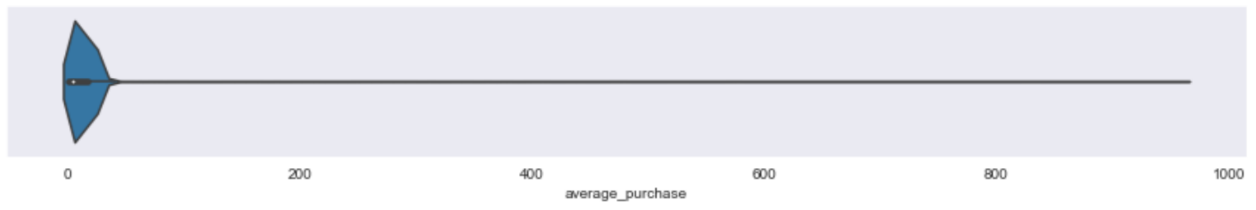
A skewed distribution is noticed for period_of_membership and frequency features, whereas average_purchase concentrates almost all values close to zero - in this case, outliers might be a problem. Perhaps some sort of winsorizing is useful.

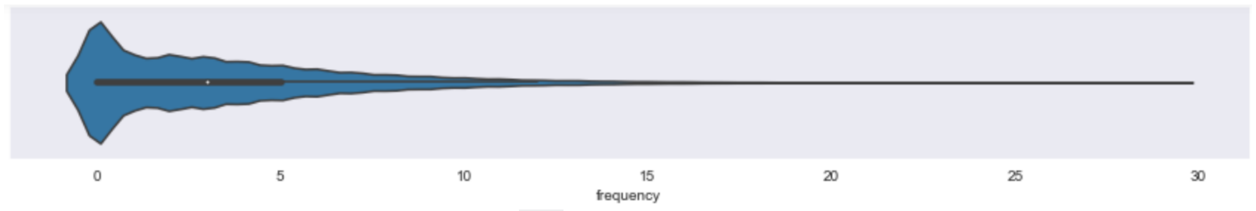
**Violin plot for Numerical Features**





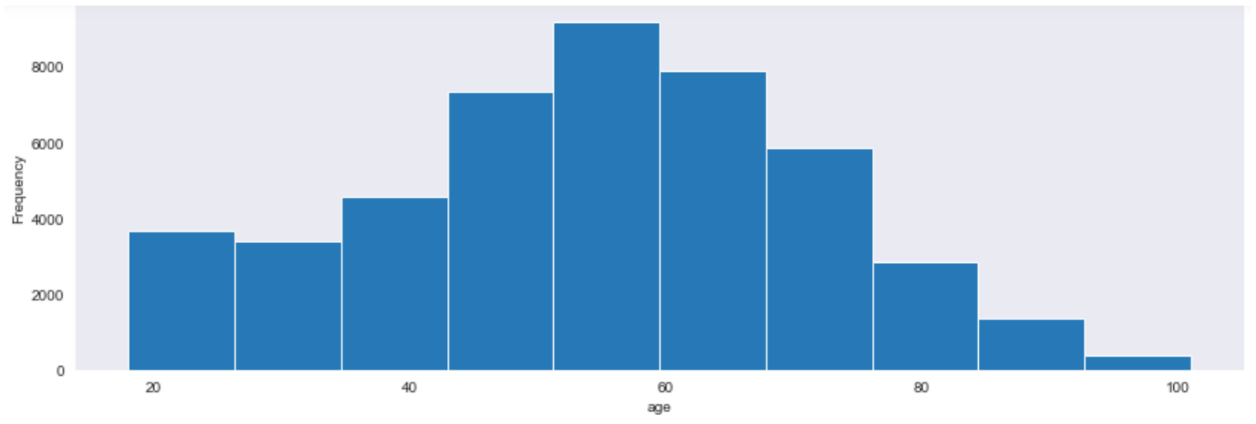


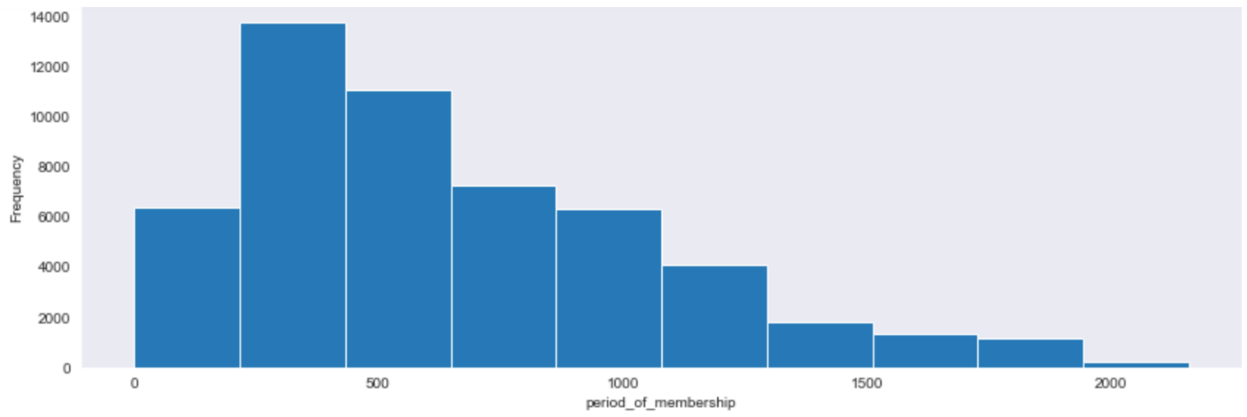


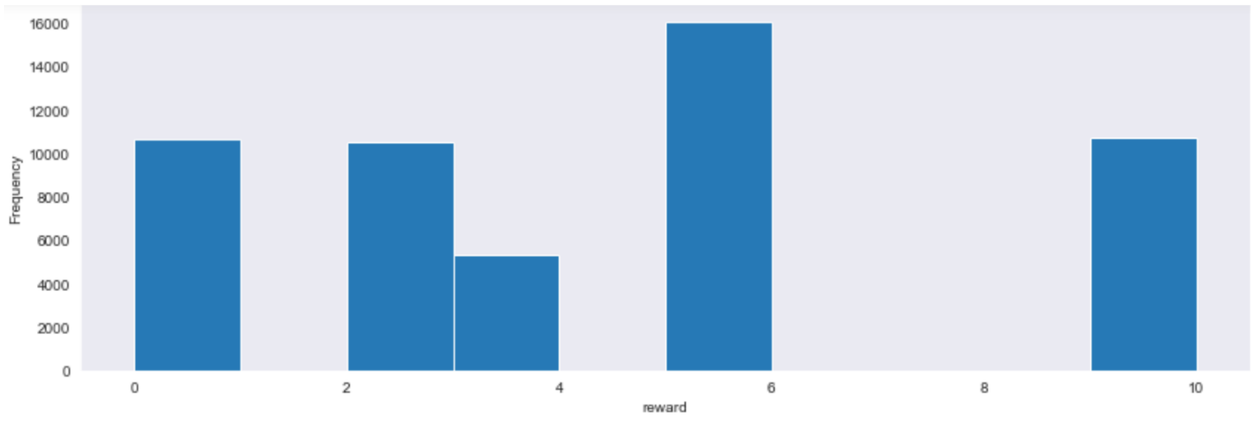


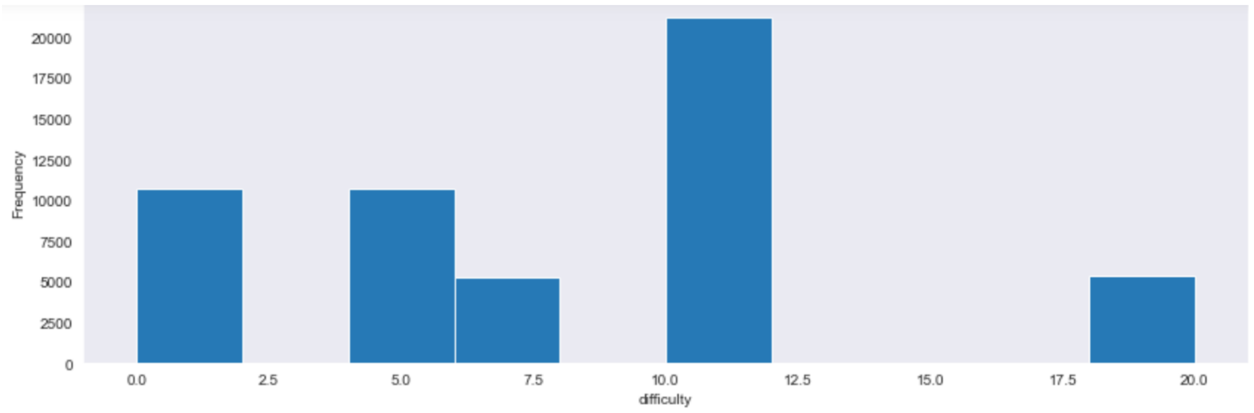
After analysis, noticed that the graph of age is closer to a normal distribution. The features related to the offers available are limited to certain values, which results in very few representative values.

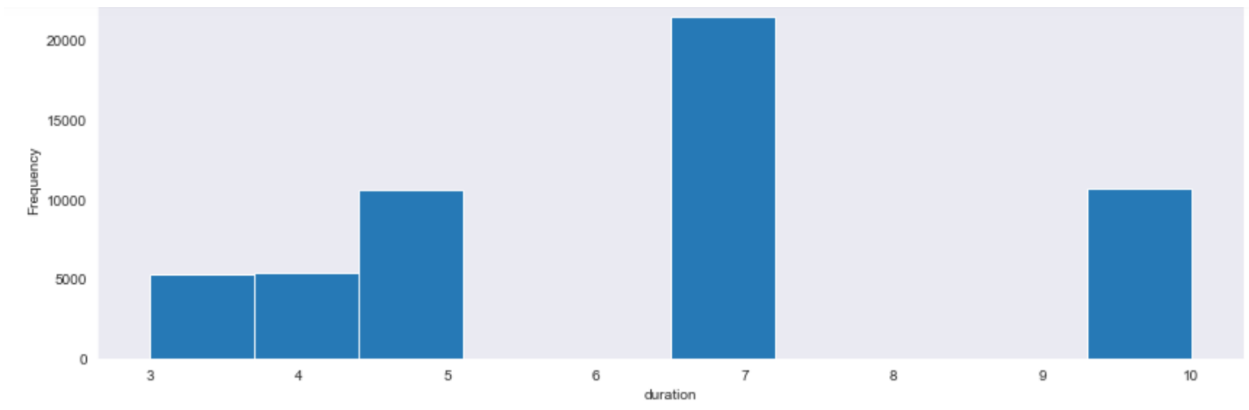
## Histogram plot showing Numerical Features Distribution

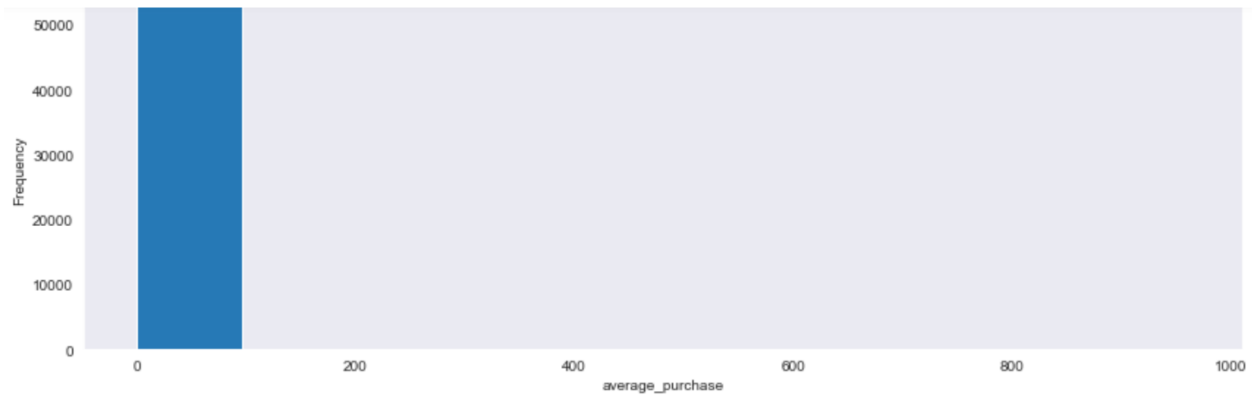


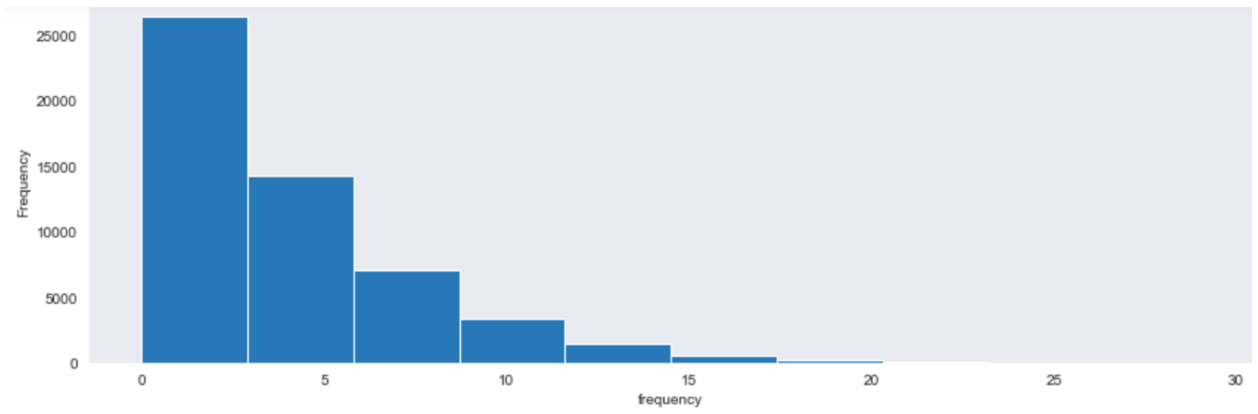

Frequency
period_of_membership


Frequency
reward


Frequency
difficulty

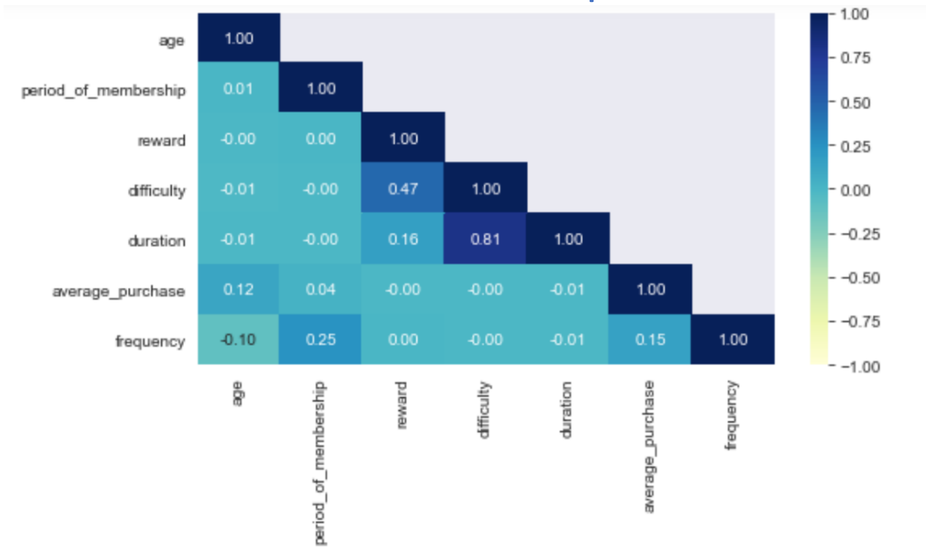




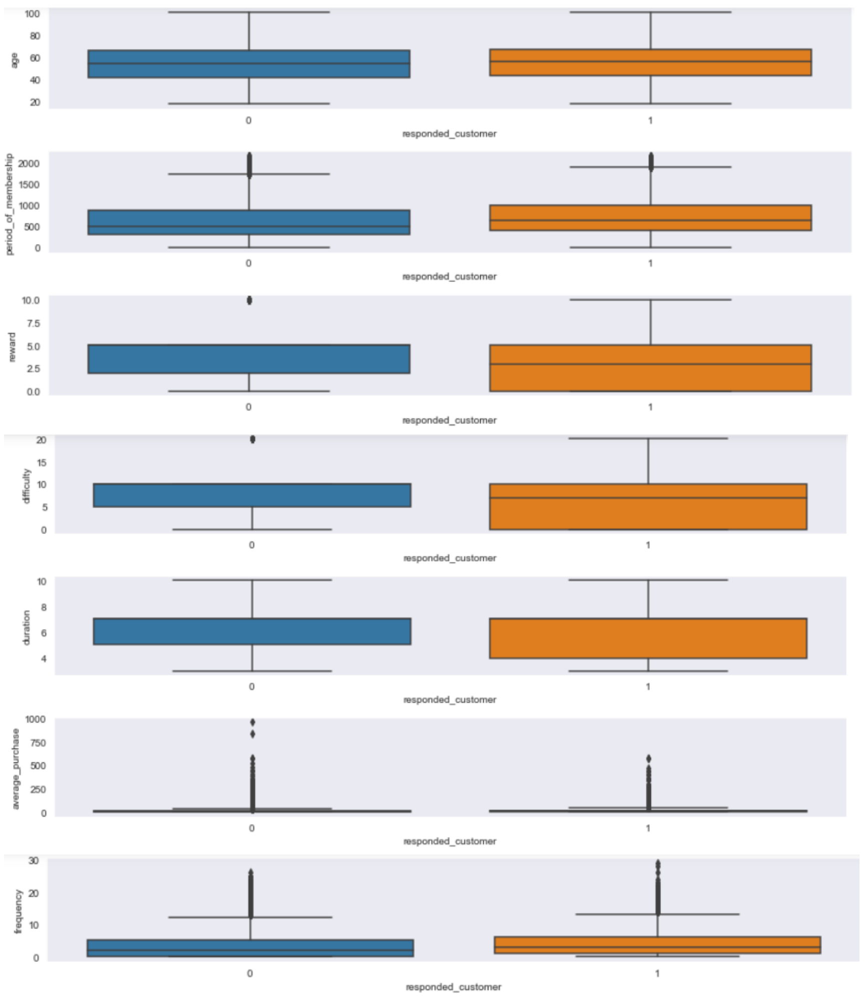


By performing bivariate analysis of numerical features, it is evident from the below correlation heatmap that only difficulty and duration are the only highly correlated pair. We could decide to follow with only one of them after checking its behavior with target feature.

## Correlation Heatmap



From the below visualization it is clear how difficulty discriminates more against the target variable.

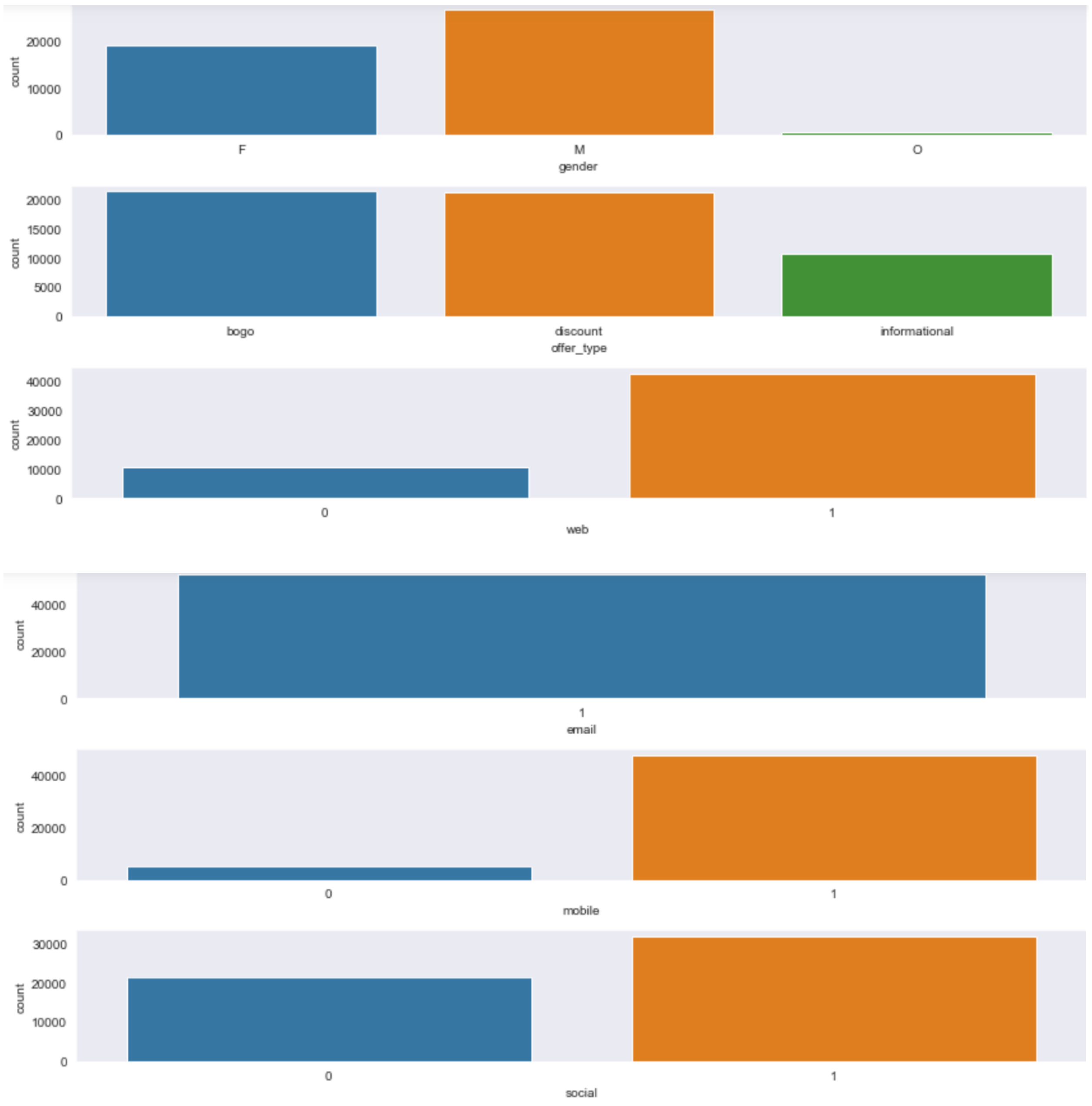


## Statistical Analysis performed on Categorical Features:

Conclusions drawn from the below visualization are as stated below:

- Most customers who took part in the 30-day test period are male
- The most common types of offers are BOGO and discount.
- Informational offers were less often used in this context.

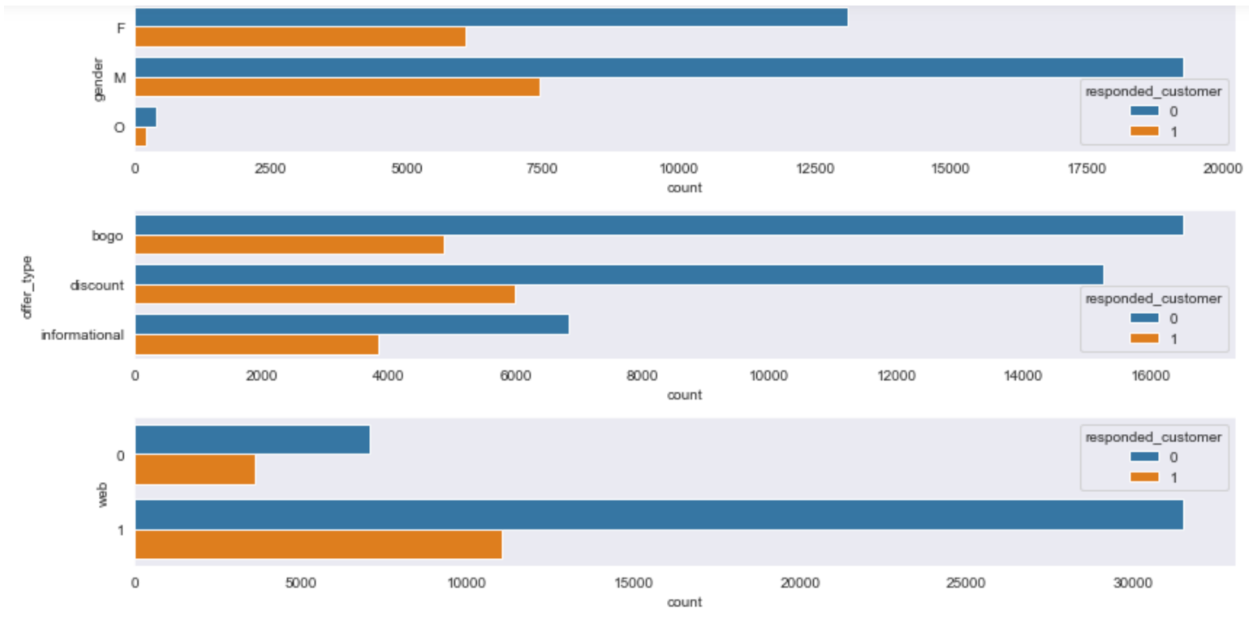
**Categorical Features Count plot**

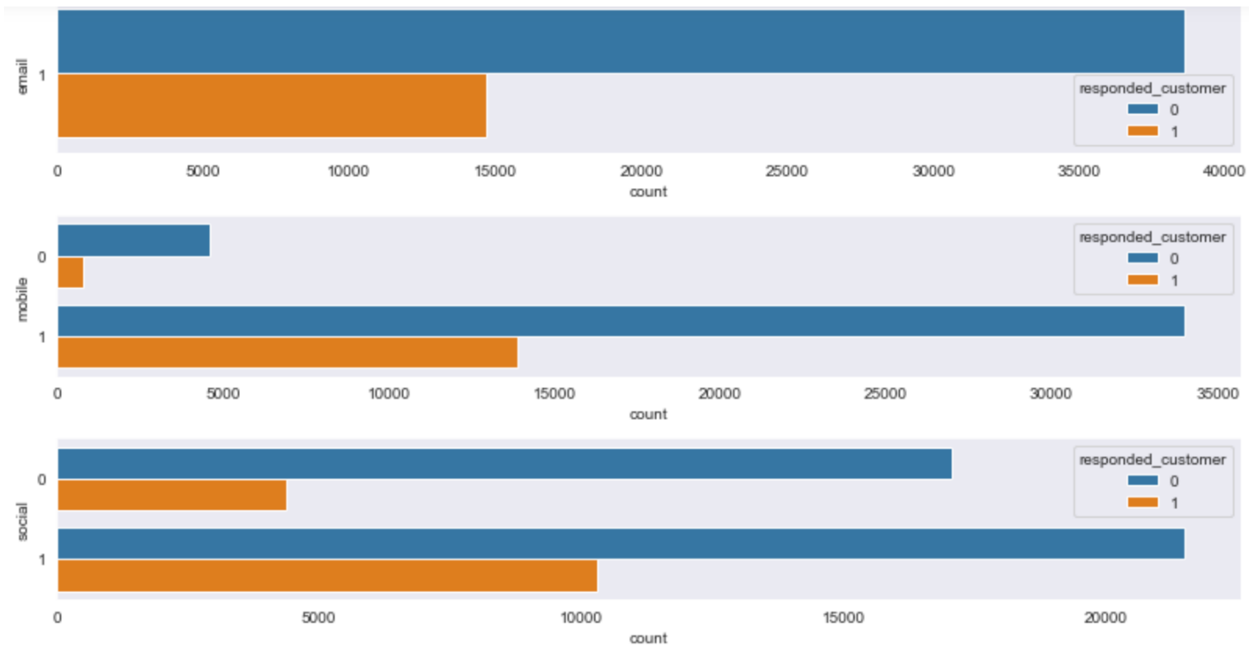


In comparison to other offer types, there is greater share of respondents in **offer_type** - informational.

Regarding **gender**, since there are fewer examples to conclude anything at all with statistical confidence, hence could infer that female customer tend to respond to promotional offers than male customers.

## Count plot for Bivariate Analysis of Categorical Features





## Algorithms and Techniques

As mentioned earlier, since this is a classification problem and a more complex one, we will be using advanced algorithm like **XGBoost.** XGBoost stands for "Extreme Gradient Boosting". XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements Machine Learning algorithms under the Gradient Boosting framework. Boosting is an ensemble learning technique to build a strong classifier from several weak classifiers in series, to avoid bias. This is achieved due to the sequential process of adjusting each single tree's output error in the subsequential tree with a given learning rate. Hence, the capability of decreasing bias. However, the risk that plays a large role in this context is increasing variance. To avoid it, we need to carefully carry out **hyperparameter tuning**.

The challenge here is that such models can adjust very well to the training data, thus leading to overfitting. We should be careful to prevent data redundancy since the same offer could be offered to a customer more than once. So, the same data point could appear in both sets, leading to an overestimate of final performance. Therefore, we will consider the pair offer-person when splitting. This process will help us to avoid overfitting.

After determining the best hyperparameters, final model is retrained using the whole training set, there by including dataset used for validation. Finally, the model performance is evaluated on a test set, which contains previously unseen data.

To study more about the XGBoost algorithm used the below references:

https://www.mygreatlearning.com/blog/xgboost-algorithm/

## Benchmark

Conversion rate is defined as the percentage of users who have completed an offer. We obtained a baseline conversion rate as 27.65%. Here our goal is to build a machine learning model that will predict the probability of a user taking up an offer or not. This will help us to increase the conversion rate.

Additionally, we also obtained the conversion rate for each offer:

| Sl No | Promotional Offer | Benchmark Conversion Rate |
|---|---|---|
| 1 | 0b1e1539f2cc45b7b9fa7c272da2e1d7 | 14.27% |
| 2 | 9b98b8c7a33c4b65b9aebfe6a799e6d9 | 19.45% |
| 3 | ae264e3637204a6fb9bb56bc8210ddfd | 21.94% |
| 4 | 3f207df678b143eea3cee63160fa8bed | 26.23% |
| 5 | 4d5c57ea9a6940dd891ad53e9dbe8da0 | 21.90% |
| 6 | 2906b810c7d4411798c6938adc9daaa5 | 20.61% |
| 7 | f19421c1d4aa40978ebb69ca19b0e20d | 28.94% |
| 8 | 5a8bc65990b245e5a138643cd4eb9837 | 44.36% |
| 9 | fafdcd668e3743c1bb461111dcafc2a4 | 39.93% |
| 10 | 2298d6c36e964ae4a3e7e9706d1fb8c2 | 36.76% |

This benchmark conversion rate is used to evaluate our model performance. The project development is considered successful when there is a **lift** of conversion rate when using our model.

# III.Methodology

## Model Pipeline: Data Preprocessing and Implementation

After the data exploration stage, we obtained a final dataset which is fully feature engineered. This dataset is then split into training, validation, and test set. Then after performing statistical analysis on the train set and learning about the features, we can proceed to training the algorithm.

From the correlation heatmap visualized earlier, we found that **difficulty** has a better relationship to the target variable, hence we removed **duration**.

Next, we selected the features according to variable type- **numerical, categorical, encoded categorical.**

| Feature Type | Feature name | Preprocessing Step |
|---|---|---|
| Numerical | age | Missing values are imputed using mean/median and standard scaling is applied |
| | period_of_membership | |
| | reward | |
| | income | |
| | difficulty | |
| | frequency | |
| | average_purchase | |
| Categorical | offer_type | Missing values are imputed using mode/constant values and One-hot encoding is applied |
| | gender | |
| Encoded Categorical | web | Missing values are imputed using mode or constant values |
| | email | |
| | mobile | |
| | social | |

As discussed in the exploratory visualization section, we identified that the dataset is imbalanced. Addressing this problem during model development by leveraging strategies as mentioned below. Using both simultaneously can lead to better results than relying solely on a single of them. Used https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/ reference to study about these techniques.

- **oversampling**: Synthetic Minority Oversampling Technique (SMOTE) to create synthetic example of positive class.

- **weighting**: Adding different weights according to class in loss function.

Synthetic Minority Oversampling Technique was always applied to the fitting of the algorithm.

Finally, XGBoost classification model was trained on training set, to evaluate the performance of validation set, and hence adjust hyperparameters. Next, with the best obtained hyperparameters, the XGBoost classification is performed on full training and validation sets to evaluate the performance on test set.

## Refinement

Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The key to machine learning algorithms is hyperparameter tuning which will help us to find a best performing model.

Below are the details of hyperparameters set:
**Preprocessing**

- **resampling-ratio**: Defined a continuous value. When performing oversampling with Synthetic Minority Oversampling Technique, this hyperparameter is used to increase training performance.
- **numerical-imputation**: Used to impute missing numerical data. Defined as categorical value - mean, median
- **categorical-imputation**: Used to impute missing categorical data. Defined as categorical value - most_frequent, N/A

**XGBoost**

- **xgboost-estimators**: Defined as integer value, it represents number of boosting rounds
- **xgboost-max-depth**: Defined as integer value, it represents maximum tree depth for base learners
- **xgboost-gamma**: Defined as continuous value, it represents minimum loss reduction required to make a further partition on a leaf node of the tree
- **xgboost-learning-rate**: Defined as continuous value, it represents learning rate used to adjust estimator outputs

F1-score metric can be interpreted as the weighted average of precision and recall. The traditional or balanced F-score (F1 score) is the harmonic mean of precision and recall, where an F1 score reaches its best value at 100 and worst at 0. Our goal here is to obtain a **maximum** F1-score.

The strategy used for hyperparameter tuning is **Random**. The main advantage of using random search is that it allows us to train jobs with a high level of parallelism. Used https://sagemaker-examples.readthedocs.io/en/latest/hyperparameter_tuning/xgboost_random_log/hpo_xgboost_random_log.html to study about Random search and hyperparameter scaling with SageMaker XGBoost.

Below are the best hyperparameter values obtained:

- **Resampling ratio:** 0.9671111857331107
- **Numerical imputation method:** mean
- **Categorical imputation method:** N/A
- **XGBoost number of estimators:** 96
- **XGBoost trees' maximum depth:** 9
- **XGBoost gamma:** 1.630531290749495
- **XGBoost learning rate:** 0.23454914692146572

With these set of hyperparameters, the F1 score obtained was 0.6650.

## IV.Results

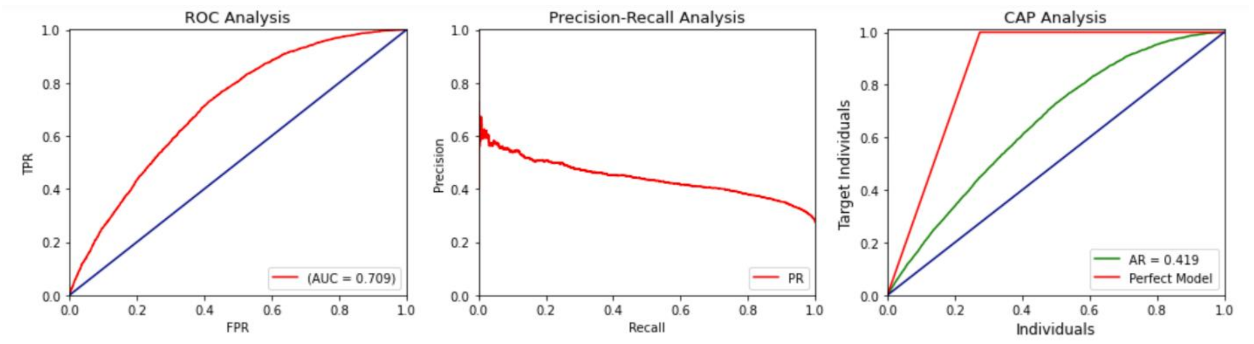
### Model Evaluation and Validation

To estimate the performance of our model in a real-world scenario, we evaluated the model performance on a dataset which the model has not seen before that is the **test** dataset.

By applying bootstrapping, classification metrics are evaluated on the test set as noted below:

| | ACCURACY | F1-SCORE | PRECISION | RECALL | ROC-AUC | COHEN-KAPPA |
|---|---|---|---|---|---|---|
| **0** | 0.6931 ± 0.0019 | 0.4517 ± 0.0035 | 0.4432 ± 0.0038 | 0.4607 ± 0.0042 | 0.7088 ± 0.0022 | 0.2388 ± 0.0043 |

Below is the visualization of the classification evaluation metrics.



Conversion rate is defined as the percentage of users who have completed an offer. Our goal is to predict the probability of a user taking up an offer, hence the business shows success when the **conversion rate increases**. The project development is successful when there is a **lift** of conversion rate when using our model.

To obtain success in business and an increase in revenue, we also calculated the conversion rate for each promotional offer. This can help business to focus on that specific offer.

| Sl No | Promotional Offer | Benchmark Conversion Rate | Model Response Conversion Rate | Lift |
|---|---|---|---|---|
| 1 | 0b1e1539f2cc45b7b9fa7c272da2e1d7 | 14.27% | 29.91% | 2.096189 |
| 2 | 9b98b8c7a33c4b65b9aebfe6a799e6d9 | 19.45% | 35.85% | 1.84332 |
| 3 | ae264e3637204a6fb9bb56bc8210ddfd | 21.94% | 37.23% | 1.696928 |
| 4 | 3f207df678b143eea3cee63160fa8bed | 26.23% | 43.97% | 1.676696 |
| 5 | 4d5c57ea9a6940dd891ad53e9dbe8da0 | 21.90% | 34.36% | 1.569301 |
| 6 | 2906b810c7d4411798c6938adc9daaa5 | 20.61% | 32.17% | 1.560848 |
| 7 | f19421c1d4aa40978ebb69ca19b0e20d | 28.94% | 39.80% | 1.375269 |
| 8 | 5a8bc65990b245e5a138643cd4eb9837 | 44.36% | 56.83% | 1.281037 |
| 9 | fafdcd668e3743c1bb461111dcafc2a4 | 39.93% | 46.58% | 1.166461 |
| 10 | 2298d6c36e964ae4a3e7e9706d1fb8c2 | 36.76% | 42.25% | 1.149256 |
|  | **OVERALL CONVERSION RATE** | **27.41%** | **44.38%** | **1.62** |

## V.Conclusion

From the results drawn, the **cumulative model conversion rate achieved is higher than the benchmark** - which in fact is **almost twice** the benchmark conversion rate.

- Cumulative Benchmark Conversion Rate: 27.41%
- Cumulative Model Conversion Rate: 44.38%
- Cumulative lift: 1.62

Also, we could notice that the conversion rate segregated by offer achieved a **maximum lift of 2.096 and a minimum lift of 1.149**.

Hence, we could conclude that the model was able to obtain an increase in conversion rate. This shall **benefit the business**, in terms of sending personalized promotional offers to targeted customers **with minimum resource utilization** and **marketing efforts**.