Capstone Project
Machine Learning Engineer Nanodegree

Richter Lau
Feb 15,2020

# Starbucks Project

## Definition

### Project Overview

Starbucks sells 4 million coffees per day back in 2016[1]. This number is growing year after year, but how did they know what keeps them growing. This is when machine learning comes into place. Starbucks has more than 23.4 million users[2], these users are contributing millions of data when making transactions with Starbucks. Therefore, combining transaction with the user data we could find out lots of information about what makes Starbucks it a better coffee shop.

In this project, we are provided with three simulated datasets that help us to solve our question. The first dataset is portfolio.json which containing offer ids and metadata about each offer. The second dataset is profile.json which containing demographic data for each customer. The last dataset is transcript.json which containing records for transactions, offers received, offers viewed, and offers completed. We will try to use machine learning algorithms to analyze what is the key feature that leads an offer to complete.

### Problem Statement & Metrics

To use the datasets to solve what is the key feature to lead an offer to complete. We will need to do the following steps.

1. Download the dataset and transform it from JSON file to DataFrame
2. Analyze the basic on the raw data by sorting and drawing graphs
3. Preprocessing the data into usable data by Machine Learning Classification Models
4. Find the best solution from these Classification Models

We will use three algorithms in total, Logistic Regression, Random Forest and K Nearest Neighbor. The K Nearest Neighbors will act as a benchmark for this research.

---

[1] Hafner, Josh. “Your Daily Starbucks Latte Won't Keep You from Retiring.” *USA Today*, Gannett Satellite Information Network, 28 Sept. 2016, www.usatoday.com/story/money/nation-now/2016/06/07/your-daily-starbucks-latte-wont-keep-you-retiring/32651191/.

[2] Molla, Rani. “Starbucks's Mobile Payments System Is so Popular in the U.S., It Has More Users than Apple's or Google's.” Vox, Vox, 22 May 2018, www.vox.com/2018/5/22/17377234/starbucks-mobile-payments-users-apple-pay-google.

To evaluate our model, we will use f1-score as a main metric and accuracy as a supporting metric. The reason[3] for using the f1-score as the main metric is because the f1 score seeks a balance between Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric. Also, it can prevent the accuracy paradox if our dataset is unbalanced.

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

$$\text{Accuracy} = \frac{\text{True Positive + True Negative}}{(\text{True Positive + False Positive + True Negative + False Negative})}$$

# Analysis

## Data Exploration and Visualization

First, we will take a look at the portfolio.

```
In [2]: # Read the portfolio samples
        print("Portfolio has %d rows and %d columns."  %portfolio.shape)

        portfolio.head()
```

```
Portfolio has 10 rows and 6 columns.
```

Out[2]:

| | channels | difficulty | duration | id | offer_type | reward |
|---|---|---|---|---|---|---|
| **0** | [email, mobile, social] | 10 | 7 | ae264e3637204a6fb9bb56bc8210ddfd | bogo | 10 |
| **1** | [web, email, mobile, social] | 10 | 5 | 4d5c57ea9a6940dd891ad53e9dbe8da0 | bogo | 10 |
| **2** | [web, email, mobile] | 0 | 4 | 3f207df678b143eea3cee63160fa8bed | informational | 0 |
| **3** | [web, email, mobile] | 5 | 7 | 9b98b8c7a33c4b65b9aebfe6a799e6d9 | bogo | 5 |
| **4** | [web, email] | 20 | 10 | 0b1e1539f2cc45b7b9fa7c272da2e1d7 | discount | 5 |

It's the shortest and easiest to analyze. It only got 10 rows in it and it doesn't get much to fix. We will only need to change it to algorithm friendly and it's good to go.

[3] Huilgol, Purva. "Accuracy vs. F1-Score." *Medium*, Analytics Vidhya, 24 Aug. 2019, medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2.

From the portfolio can we get the number of different offer types.

```
# Count the number of different offer type
portfolio.groupby('offer_type')['id'].count()
```

```
offer_type
bogo             4
discount         4
informational    2
Name: id, dtype: int64
```

Then we will preprocess[4] the dataset by

1. Change the column name ‘id’ into ‘offer_id’ to create a consistency through out the data
2. Split channels into individual type of channel by using One hot encoding[5].
3. Split offer type into individual type of offer by using One hot encoding

| | difficulty | duration | offer_id | reward | web | email | mobile | social | bogo | informational | discount |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10 | 7 | ae264e3637204a6fb9bb56bc8210ddfd | 10 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| **1** | 10 | 5 | 4d5c57ea9a6940dd891ad53e9dbe8da0 | 10 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| **2** | 0 | 4 | 3f207df678b143eea3cee63160fa8bed | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

Second, let’s check with the profile.

```
In [4]: # Read the profile samples
        print("Profile has %d rows and %d columns."  %profile.shape)
        profile.head()
```

```
Profile has 17000 rows and 5 columns.
```

Out[4]:

| | 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 |

It contains 17000 customers’ basic geographic information such as gender, age, and income. Most of these data are already algorithm friendly, but we will still need to check for duplicates and made all the categories algorithm friendly.

[4] Sharma, Mohit. “What Steps Should One Take While Doing Data Preprocessing?” By Mohit Sharma, 20 June 2018, hackernoon.com/what-steps-should-one-take-while-doing-data-preprocessing-502c993e1caa.

[5] Vasudev. “What Is One Hot Encoding? Why And When Do You Have to Use It?” What Is One Hot Encoding? Why And When Do You Have to Use It?, 25 Oct. 2019, hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f.

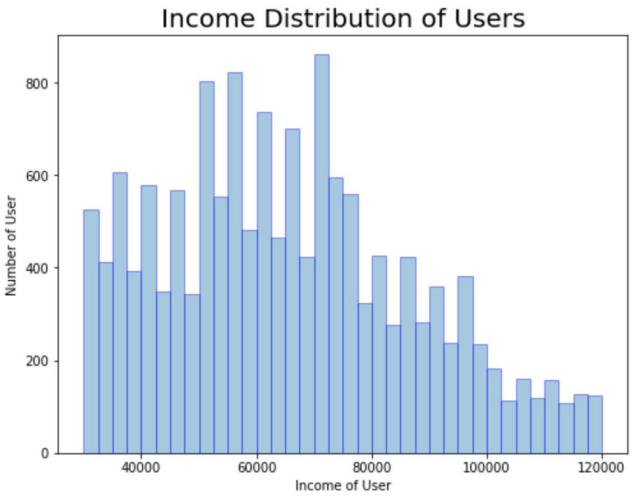
From this dataset, we found out there is some registered customer didn't provide gender and income information. We also found out that some registered customers has an age of 118. We proposed these are the same group of people so we run a test on it, and the result shows that they are the same group of customers. And we decided to remove it from the data for a better outcome.

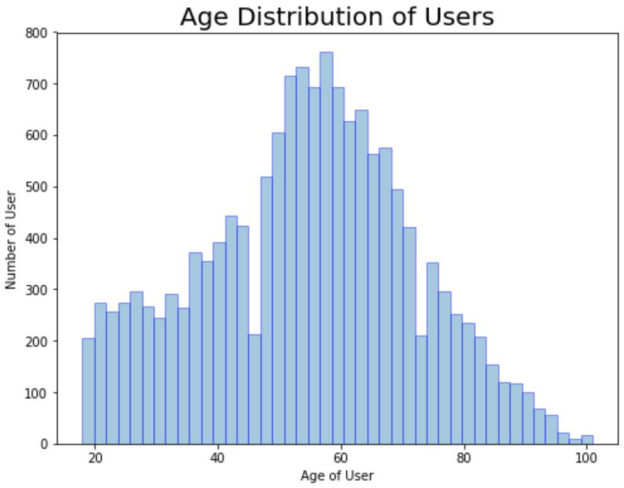
We will preprocess this dataset by:

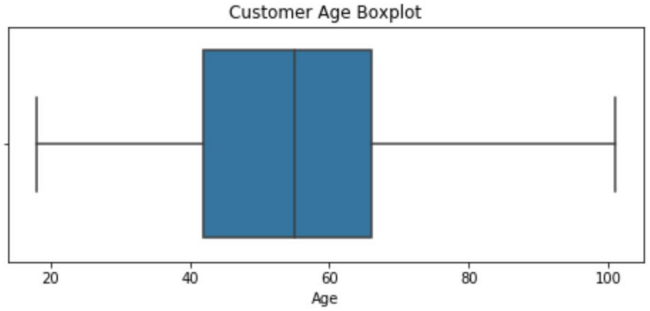
1. Remove all the age of 118 customers
2. Change the format of 'became_member_on' to YYYY-MM-DD format
3. Split the date into individual year, month, and day
4. Split the gender into different gender type by using One hot encoding
5. Change column name 'id' into 'customer_id'

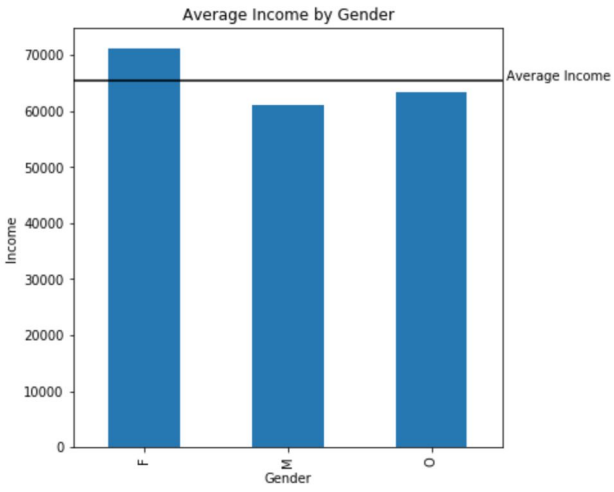
| | age | gender | customer_id | income | member_year | member_month | member_day | member_date | male | female | others |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 55 | F | 0610b486422d4921ae7d2bf64640c50b | 112000.0 | 2017 | 7 | 15 | 2017-07-15 | 0 | 1 | 0 |
| **3** | 75 | F | 78afa995795e4d85b5d9ceeca43f5fef | 100000.0 | 2017 | 5 | 9 | 2017-05-09 | 0 | 1 | 0 |
| **5** | 68 | M | e2127556f4f64592b11af22de27a7932 | 70000.0 | 2018 | 4 | 26 | 2018-04-26 | 1 | 0 | 0 |

These charts shows some inside about this dataset



Last but not least, we will analyze the transcript as well.

```
In [10]: # Read the transcript samples
         print("Transcript has %d rows and %d columns."  %transcript.shape)
         transcript.head()
```

```
Transcript has 306534 rows and 4 columns.
```

Out[10]:

| | 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'} |

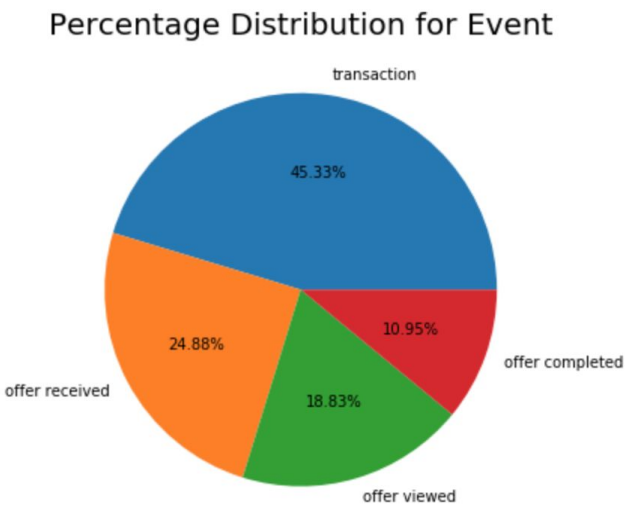
This is the hardest dataset to deal with. It mixed both transaction amount and offer id into one value column. We have to find out some tricks to split it out. Also, it might be a small typo through out the data gathering process. It got ‘offer id’ and ‘offer_id’ in the same column we will also need to merge these as well.

We will preprocess this dataset by:

1. Split the event into individual event by using One hot encoding
2. Change the ‘person’ column name into ‘customer_id’
3. Split the value into key, and value
4. Merge ‘offer id’ and ‘offer_id’ into one type
5. Add an amount column to store the amount if that’s a transaction
6. Change ‘time’ column from hours into days
7. Drop duplicated values

| | event | customer_id | time | offer_id | amount |
|---|---|---|---|---|---|
| 0 | offer received | 78afa995795e4d85b5d9ceeca43f5fef | 0.0 | 9b98b8c7a33c4b65b9aebfe6a799e6d9 | NaN |
| 1 | offer received | a03223e636434f42ac4c3df47e8bac43 | 0.0 | 0b1e1539f2cc45b7b9fa7c272da2e1d7 | NaN |
| 2 | offer received | e2127556f4f64592b11af22de27a7932 | 0.0 | 2906b810c7d4411798c6938adc9daaa5 | NaN |

This chart shows some inside about this dataset




Because there are two types of value and it's completely irrelevant. So we split transcript into two DataFrame.

Transaction_df that contains all the transaction and amount detail

| | customer_id | time | amount |
|---|---|---|---|
| **12654** | 02c083884c7d45b39cc68e1314fec56c | 0.0 | 0.83 |
| **12657** | 9fa9ae8f57894cc9a3b8a9bbe0fc1b2f | 0.0 | 34.56 |
| **12659** | 54890f68699049c2a04d415abc25e717 | 0.0 | 13.23 |

Offers_df that contains all the offer progress (We also split the event into individual event here)

| | customer_id | time | offer_id | received | completed | viewed |
|---|---|---|---|---|---|---|
| **0** | 78afa995795e4d85b5d9ceeca43f5fef | 0.0 | 9b98b8c7a33c4b65b9aebfe6a799e6d9 | 1 | 0 | 0 |
| **1** | a03223e636434f42ac4c3df47e8bac43 | 0.0 | 0b1e1539f2cc45b7b9fa7c272da2e1d7 | 1 | 0 | 0 |
| **2** | e2127556f4f64592b11af22de27a7932 | 0.0 | 2906b810c7d4411798c6938adc9daaa5 | 1 | 0 | 0 |

Now we need to create a new DataFrame that contain all of these DataFrame for further process

1. Identify whether a transaction is related to an offer
2. Match the time and amount with the transcript
3. Match the customer id with the portfolio

```
final.head()
```

| | bogo | customer_id | difficulty | discount | duration | email | informational | mobile | offer_id | offer_successful | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 78afa995795e4d85b5d9ceeca43f5fef | 5 | 0 | 7 | 1 | 0 | 1 | 9b98b8c7a33c4b65b9aebfe6a799e6d9 | 1 | ... |
| **1** | 0 | 78afa995795e4d85b5d9ceeca43f5fef | 0 | 0 | 3 | 1 | 1 | 1 | 5a8bc65990b245e5a138643cd4eb9837 | 0 | ... |
| **2** | 1 | 78afa995795e4d85b5d9ceeca43f5fef | 10 | 0 | 7 | 1 | 0 | 1 | ae264e3637204a6fb9bb56bc8210ddfd | 1 | ... |
| **3** | 1 | 78afa995795e4d85b5d9ceeca43f5fef | 5 | 0 | 5 | 1 | 0 | 1 | f19421c1d4aa40978ebb69ca19b0e20d | 1 | ... |
| **4** | 0 | e2127556f4f64592b11af22de27a7932 | 10 | 1 | 7 | 1 | 0 | 1 | 2906b810c7d4411798c6938adc9daaa5 | 0 | ... |

5 rows × 24 columns

# Methodology

We will use three types of classification algorithms to solve the problem, one of them will be acted as a benchmark. Based on the charts above, we have an unbalanced dataset. To avoid accuracy paradox we will focus on the F1-score for all these algorithms and use accuracy as supporting metrics.

1. Logistic Regression[6] - The goal of logistic regression is to find the best fitting model to describe the relationship between the dependent variable and a set of independent variables, which is very close to our goal in this research.

2. Random Forest[7] - One advantage of random forest is that the trees protect each other from their individual errors. This method can decrease the overfitting issue that often happened in decision trees.

3. K - Nearest Neighbor[8] - KNN is a supervised classification algorithm that will give new data points accordingly to the k number or the closest data points, which is different than the k-means clustering that is an unsupervised clustering algorithm that gathers and groups data into k number of clusters. We will decide the optimal K value by the Elbow Method[9]. This method will be acted as a benchmark for this research

Before running all these algorithmns we need to scale our data to make sure everything works fine. We will also need to drop some of unneccesary columns for more accuray outcome.

We use MinMaxScaler to scale our data.

[6] Brownlee, Jason. “Logistic Regression for Machine Learning.” Machine Learning Mastery, 12 Aug. 2019, machinelearningmastery.com/logistic-regression-for-machine-learning/.
[7] Yiu, Tony. “Understanding Random Forest.” Medium, Towards Data Science, 14 Aug. 2019, towardsdatascience.com/understanding-random-forest-58381e0602d2.
[8] Umamaheswaran, Venkatesh. “Comprehending K-Means and KNN Algorithms.” Medium, Becoming Human: Artificial Intelligence Magazine, 14 Nov. 2018, becominghuman.ai/comprehending-k-means-and-knn-algorithms-c791be90883d.
[9] Doumbia, Moussa. “Elbow Method in Supervised Learning(Optimal K Value).” Medium, Medium, 24 Aug. 2019, medium.com/@moussadoumbia_90919/elbow-method-in-supervised-learning-optimal-k-value-99d425f229e7.

```
from sklearn.preprocessing import MinMaxScaler

# Initialize a scaler
scaler = MinMaxScaler()
features_scaled = pd.DataFrame(scaler.fit_transform(features.astype(float)))
features_scaled.columns = features.columns
features_scaled.index = features.index
```

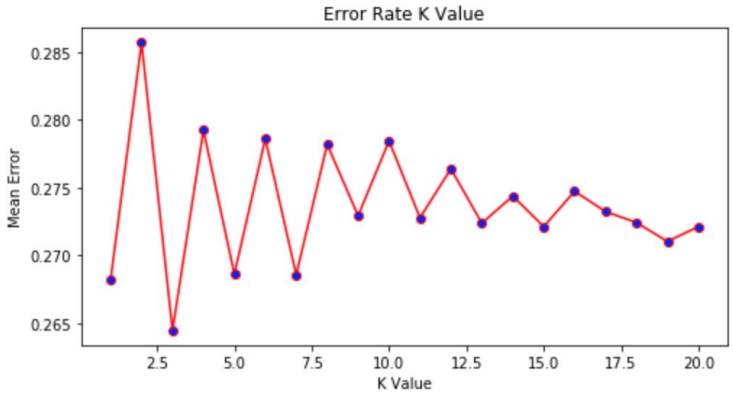
Then we used train_test_split to Split arrays into random test and train subsets, which is 30% versus 70% in this case.

```
X_train, X_test, y_train, y_test = train_test_split(features_scaled.values, target.values, test_size=0.3, random_state=
```

```
print("Training set has %d" %(X_train.shape[0]))
print("Testing set has %d" %(X_test.shape[0]))
```

```
Training set has 46550
Testing set has 19951
```

First, we will use the k-nearest neighbors algorithm as a benchmark for this research. In this algorithm we will need to find the optimal k value, k value is used to indicate the amount of nearest points we want to find to our value.




As we see on the chart, K = 3 has the lowest mean error. So we will use 3 as our K value.

```
classifier_refined = KNeighborsClassifier(n_neighbors=3)
KNN_classifier = classifier_refined.fit(X_train, y_train)
knn_output_train, y_pred_train = train_predict(classifier, X_train, y_train)
knn_output, y_pred_test = test_predict(classifier, X_test, y_test)
```

```
KNeighborsClassifier
{'F1-Score: training dataset': 0.7431832593532023, 'Accuracy: training dataset': 0.7563909774436091}
KNeighborsClassifier
{'F1-score: testing dataset': 0.7121573617517628, 'Accuracy: test dataset': 0.727883314119593}
```

The f1-score is about 74% for the training dataset and 71% for the test dataset, which is a good number for a benchmark.

Now, let's move on to the logistic regression. Logistic regression is the method we suggested in the proposal because this method helps find the best fitting model to describe the relationship between the dependent variable and a set of independent variables.

First we will try the default setup for Logistic Regression, we select liblinear as our solver because it's better for small dataset.

```
# Initiate LogisticRegression
lr_clf = LogisticRegression(solver='liblinear')

# Fit train data to the classifier
LR_classifier = lr_clf.fit(X_train, y_train)
```

```
# Print the result for LogisticRegression
lr_output_train = train_predict(LR_classifier, X_train, y_train)
lr_output_test = test_predict(LR_classifier, X_test, y_test)
```

```
LogisticRegression
{'F1-score: training dataset': 0.8530961473389084, 'Accuracy: training dataset': 0.8694307196562836}
LogisticRegression
{'F1-score: testing dataset': 0.8516201117318436, 'Accuracy: test dataset': 0.86687384091023}
```

The f1-score is about 85% for both the training and test dataset, which is a better score compare to the benchmark.

Then we will try to improve our model by tuning some hyperparameters. We change the random_state to 42, change the penalty from l2 to l1, and change the c value from 1.00 to 100.

```
# Tuning for the hyperparameters
lr_clf = LogisticRegression(random_state=42, penalty = 'l1', C=100, solver='liblinear')
LR_classifier = lr_clf.fit(X_train, y_train)
lr_output_train = train_predict(LR_classifier, X_train, y_train)
lr_output_test = test_predict(LR_classifier, X_test, y_test)
```

```
LogisticRegression
{'F1-score: training dataset': 0.8555505058961418, 'Accuracy: training dataset': 0.8702685284640171}
LogisticRegression
{'F1-score: testing dataset': 0.8559265811587793, 'Accuracy: test dataset': 0.8693799809533357}
```

After tuning the hyperparameters we can get a slightly better result from this method. However, this is not enough we will try another method for a better result.

Finally, let's check out the random forest. This method can decrease the overfitting issue that often happened in decision trees. So hopefully we can get a better result from this method.

Now let's start with a default setup.

```
# Initiate RandomForestClassifier
rf_clf = RandomForestClassifier()

# Fit train data to the classifier
RF_classifier= rf_clf.fit(X_train, y_train)
```

```
/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: I
e default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
# Print the result for RandomForestClassifier
rf_output_train = train_predict(RF_classifier, X_train, y_train)
rf_output_test = test_predict(RF_classifier, X_test, y_test)
```

```
RandomForestClassifier
{'F1-score: training dataset': 0.9930220320757739, 'Accuracy: training dataset': 0.9934479054779807}
RandomForestClassifier
{'F1-score: testing dataset': 0.8985446108415945, 'Accuracy: test dataset': 0.9035637311412962}
```

The f1-score is higher than 99% on the training dataset and higher than 89% on the test dataset, which is a very decent score. However, we will try to improve it by tuning the hyperparameters. We change the random_state to 42change the n_estimators  from 10 to 100.

```
# Tuning for the hyperparameters
rf_clf = RandomForestClassifier(random_state=42, n_estimators=100)
RF_classifier= rf_clf.fit(X_train, y_train)
rf_output_train = train_predict(RF_classifier, X_train, y_train)
rf_output_test = test_predict(RF_classifier, X_test, y_test)
```

```
RandomForestClassifier
{'F1-score: training dataset': 0.9999771840562185, 'Accuracy: training dataset': 0.9999785177228786}
RandomForestClassifier
{'F1-score: testing dataset': 0.9077356130108424, 'Accuracy: test dataset': 0.9112826424740614}
```

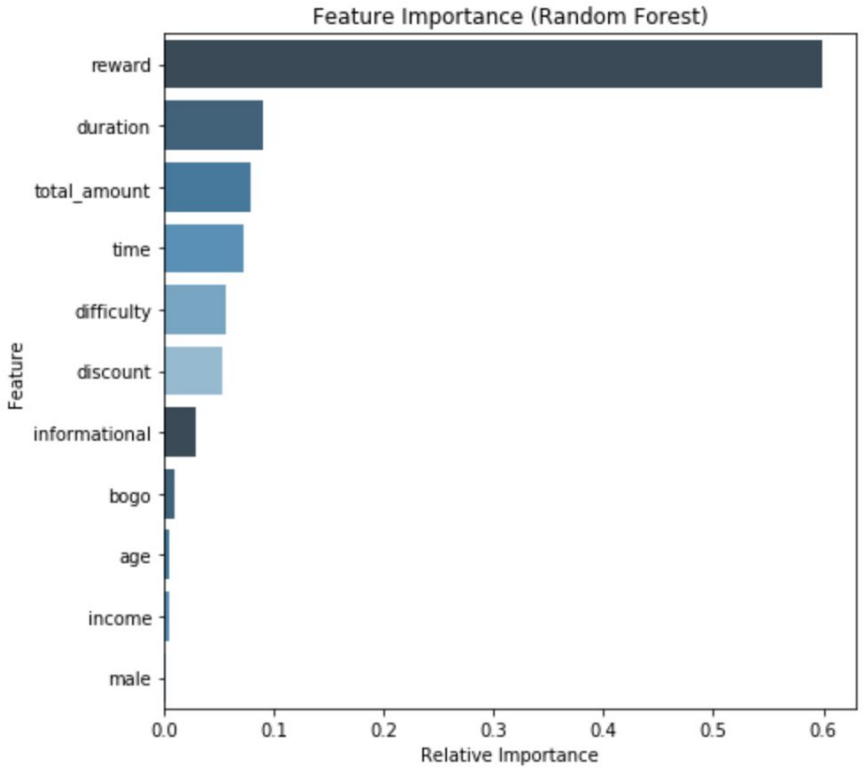
After turning the hyperparameters, we get the f1-score is close to 100% on the training dataset and higher than 90% on the test dataset, this is the best score amount these three models.

# Results

## Conclusion

Based on the models we ran, these are the result for it. As we can see Random Forest got the highest f1-score and accuracy amount all the training and testing. So we will focus on its prediction.

| | K - Nearest Neighbor | Logistic Regression | Random Forest |
|---|---|---|---|
| F1-score_Train | 0.743183 | 0.853096 | 0.999977 |
| Accuracy_Train | 0.756391 | 0.869431 | 0.999979 |
| F1-score_Test | 0.712157 | 0.851620 | 0.907736 |
| Accuracy_Test | 0.727883 | 0.866874 | 0.911283 |




This bar chart shows a significant amount of information about what makes a offer attractive. The reward given to acustomer for completing an offer makes a tremedous different. Also, Discount offers are more attractive than the bogo offers. Male is more react to the offer compare to other genders.

## Things to Improve

I remember back in Udacity classroom, there is an interview saying that people spend 80% of the time to organized or labeling the data, and spend 20% of the time complaining about the data. That is the exact case here as well. We hope to have a more efficient way to preprocess the data and that will save a lot of time. Back to this research, I believe the f1-score and accuracy can be improved by testing the hyperparameters, for example, the penalty and c value for the logistic regression, the n_estimators and the max_features for the random forest, etc. After this project, I believe I can solve more problem by using machine learning in the future.