Machine Learning Engineer Nano Degree

Capstone Project

Sameerna Joshi
25-04-2021

Definition

- Starbucks the coffee place, Starbucks Corporation is an American multinational chain of coffeehouses and roastery reserves headquartered in Seattle, Washington. As the world's largest coffeehouse chain, Starbucks is seen to be the main representation of the United States' third wave of coffee culture. Starbucks has a mobile application where the user can order coffee online or place an order online and go pick it up from the store.
- "My Starbucks Rewards™ membership", after paying through the app the user receives free Stars/Bonus points that can be used to redeem a free drink of the user's choice. This app also offers various promotions to the users which includes Discount in a discount, a user gains a reward equal to a fraction of the amount spent on drinks, BOGO (Buy One Get One Free), in a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount and Informational offer which basically includes any release of new product and there is no reward, but neither is there a requisite amount that the user is expected to spend.

Problem Statement

- With the Starbucks dataset I am trying to make an effort to understand the Starbucks customer's buying behaviour.
- What influences their purchasing decisions? Do they respond to promotional offers.
- Which demographic groups respond to which offer types.
- As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product. Some of the users might not receive any offer during certain weeks. Not all users receive the same offer, and that is the challenge to solve with this data set.

Metrics

With F1 Score as an evaluation metrics we can determine which model will be the best fit and which model can perform better. The F1 Score is the 2*((precision*recall) / (precision + recall)). It is also called the F Score or the F Measure ,the F1 score conveys the balance between the precision and the recall.

- lowest possible value of F1 score is 0 and Maximum value is 100, more the F1 score better the accuracy

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

- TP = Number of True Positives
- FP = Number of False Positives
- FN = number of false Negatives

Analysis

Data Exploration

The data is contained in three files:

1) portfolio.json - containing offer ids and meta data about each offer(duration, type, etc.)

2) profile.json - demographic data for each customer

3) transcript.json - records for transactions, offers received, offers viewed,and offers complete

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

portfolio.json

- id (string) - offer id
- offer_type (string) - type of offer ie 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)

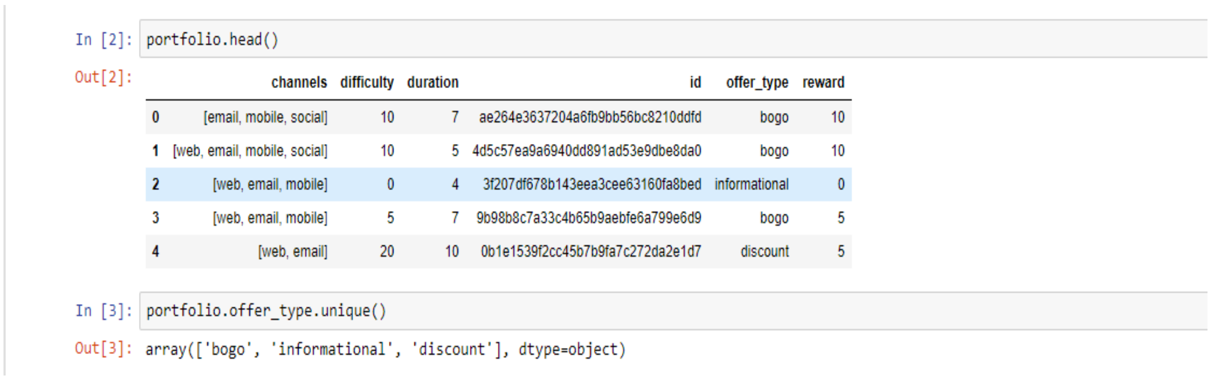
profile.json

- 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' forother rather than M or F)
- id (str) - customer id
- income (float) - customer's income

transcript.json

- event (str) - record description (ie 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 dependingon the record

PORTFOLIO

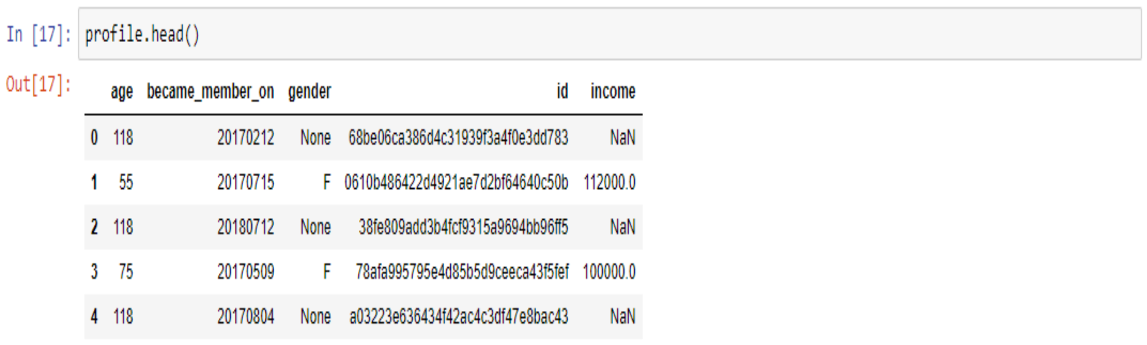


```
In [2]: portfolio.head()
Out[2]:
```

	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

```
In [3]: portfolio.offer_type.unique()
Out[3]: array(['bogo', 'informational', 'discount'], dtype=object)
```

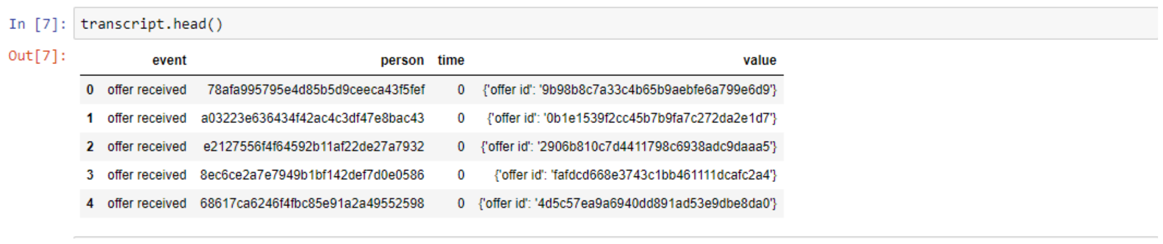
Profile



```
In [17]: profile.head()
Out[17]:
```

	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

Transcript



In [7]:
```
transcript.head()
```

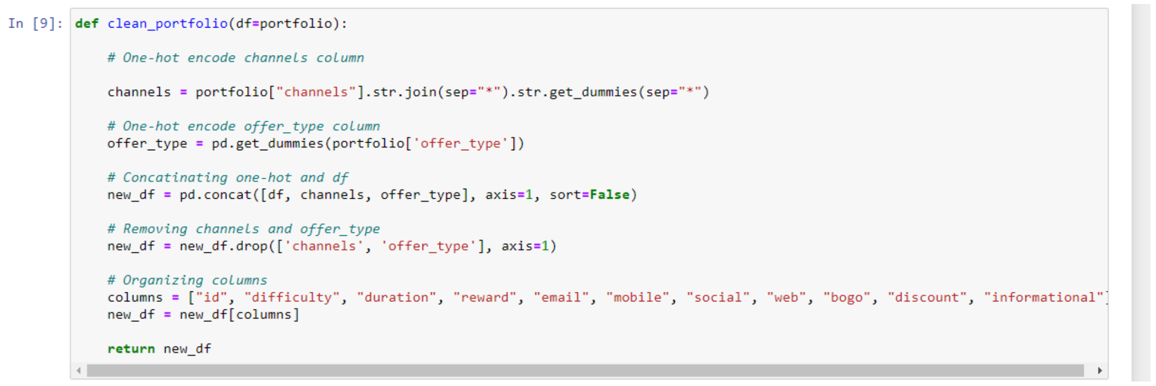
Out[7]:

	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'}

Data cleaning

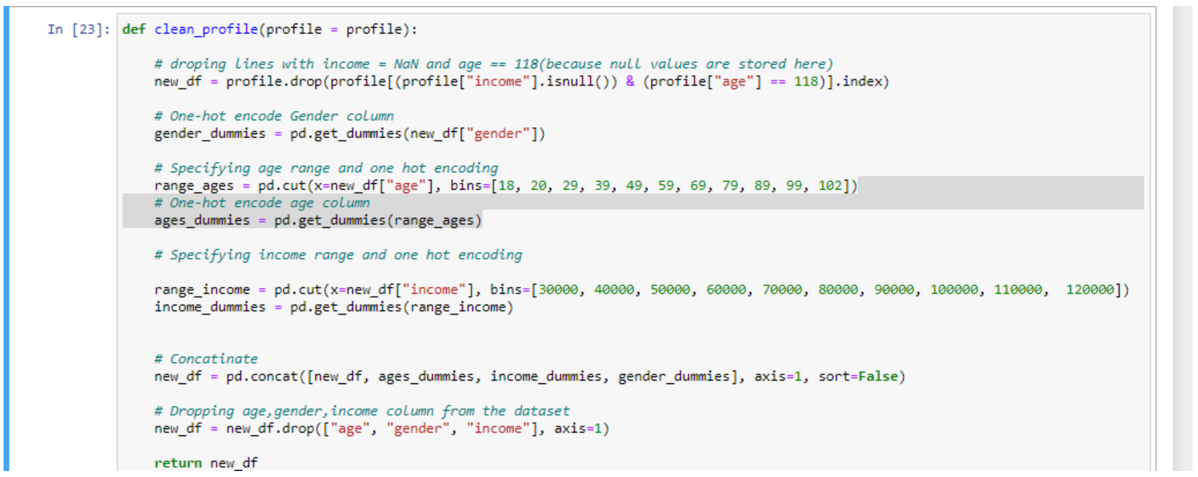
Firstly we pass the data to a function to transform the data so that it can be fed to the machine learning model, with one hot encoding we assign the values to each feature.

Here we re- organise the columns appropriately.



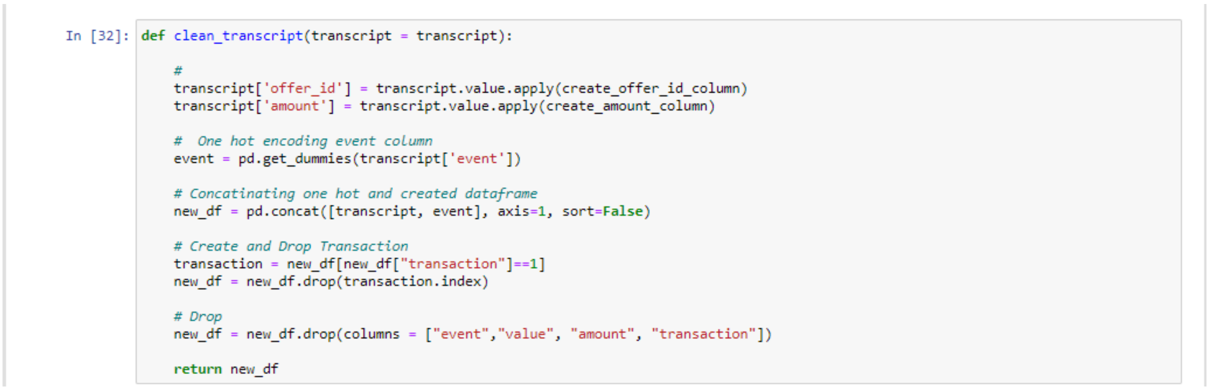
In [9]:
```
def clean_portfolio(df=portfolio):

    # One-hot encode channels column

    channels = portfolio["channels"].str.join(sep="*").str.get_dummies(sep="*")

    # One-hot encode offer_type column
    offer_type = pd.get_dummies(portfolio['offer_type'])

    # Concatinating one-hot and df
    new_df = pd.concat([df, channels, offer_type], axis=1, sort=False)

    # Removing channels and offer_type
    new_df = new_df.drop(['channels', 'offer_type'], axis=1)

    # Organizing columns
    columns = ["id", "difficulty", "duration", "reward", "email", "mobile", "social", "web", "bogo", "discount", "informational"]
    new_df = new_df[columns]

    return new_df
```

Cleaning profile dataset



In [23]:
```
def clean_profile(profile = profile):

    # droping lines with income = NaN and age == 118(because null values are stored here)
    new_df = profile.drop(profile[(profile["income"].isnull()) & (profile["age"] == 118)].index)

    # One-hot encode Gender column
    gender_dummies = pd.get_dummies(new_df["gender"])

    # Specifying age range and one hot encoding
    range_ages = pd.cut(x=new_df["age"], bins=[18, 20, 29, 39, 49, 59, 69, 79, 89, 99, 102])
    # One-hot encode age column
    ages_dummies = pd.get_dummies(range_ages)

    # Specifying income range and one hot encoding

    range_income = pd.cut(x=new_df["income"], bins=[30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000,  120000])
    income_dummies = pd.get_dummies(range_income)


    # Concatinate
    new_df = pd.concat([new_df, ages_dummies, income_dummies, gender_dummies], axis=1, sort=False)

    # Dropping age,gender,income column from the dataset
    new_df = new_df.drop(["age", "gender", "income"], axis=1)

    return new_df
```

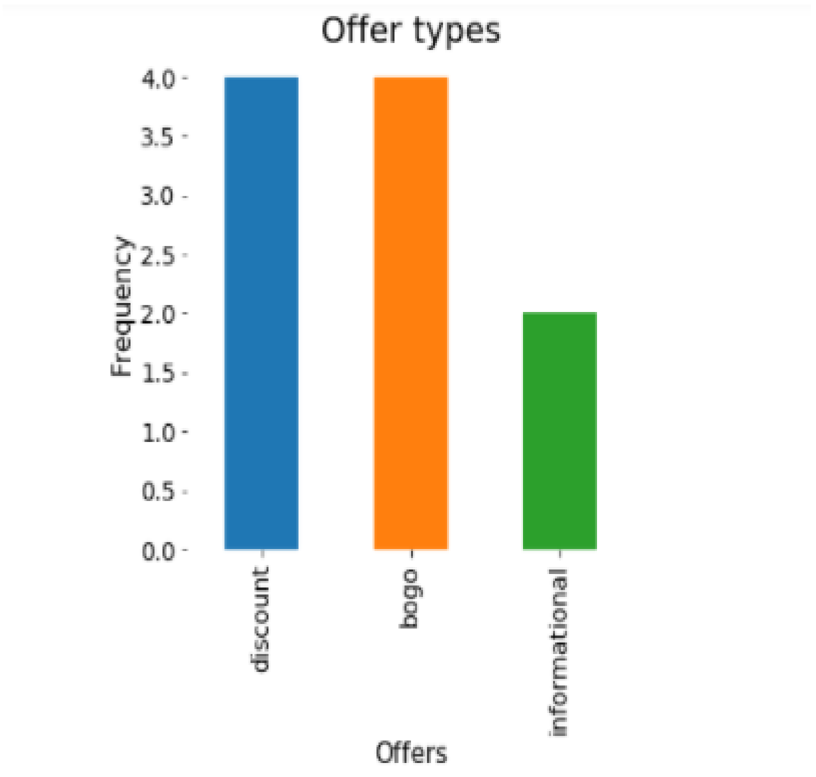
Cleaning Transcript dataset



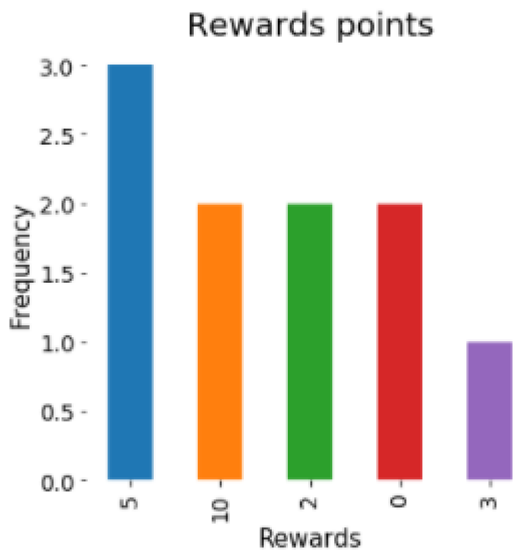
```
In [32]: def clean_transcript(transcript = transcript):

             #
             transcript['offer_id'] = transcript.value.apply(create_offer_id_column)
             transcript['amount'] = transcript.value.apply(create_amount_column)

             #  One hot encoding event column
             event = pd.get_dummies(transcript['event'])

             # Concatinating one hot and created dataframe
             new_df = pd.concat([transcript, event], axis=1, sort=False)

             # Create and Drop Transaction
             transaction = new_df[new_df["transaction"]==1]
             new_df = new_df.drop(transaction.index)

             # Drop
             new_df = new_df.drop(columns = ["event","value", "amount", "transaction"])

             return new_df
```

Data Analysis and Visualization.

Firstly, we observe the Offers Types.

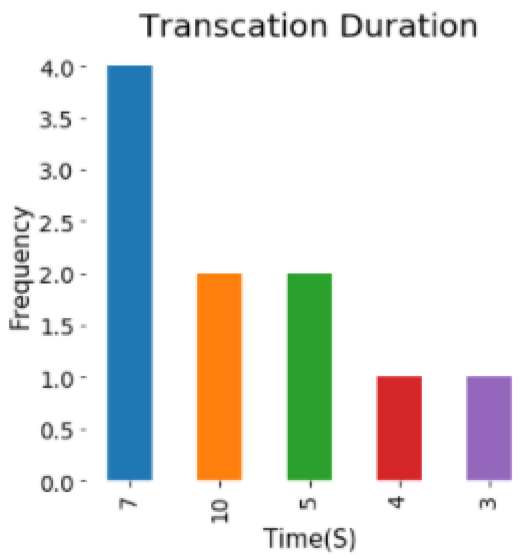


Reward points.



Transaction Duration.

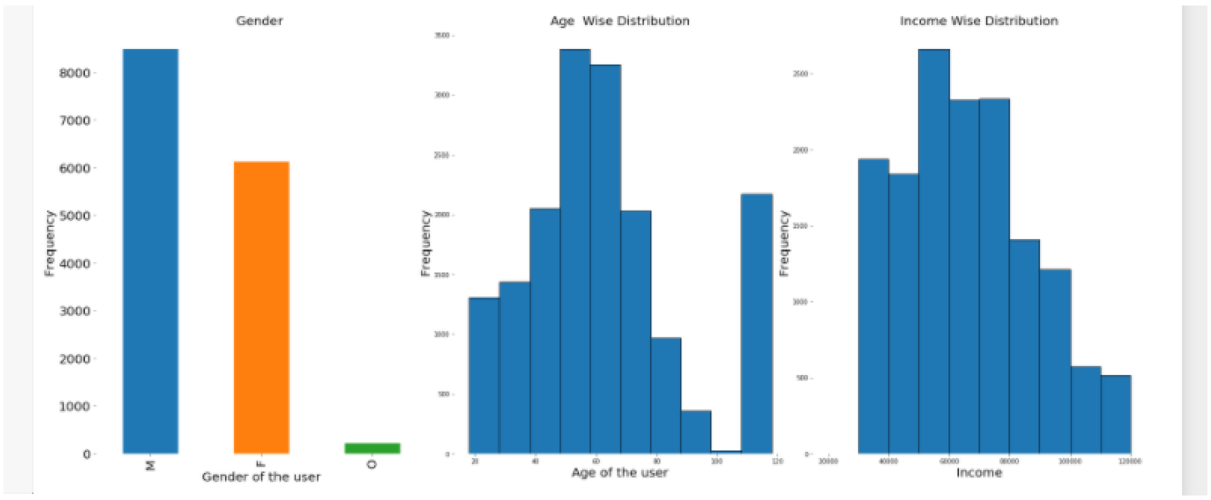
The average time taken by the customer to complete the transaction.



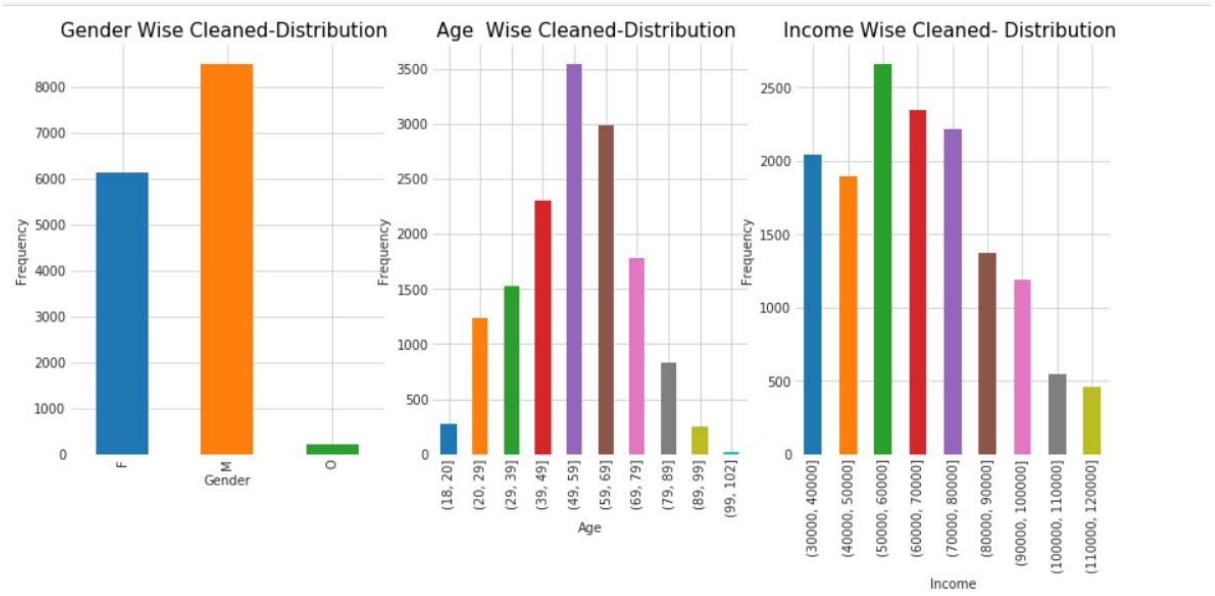
Exploring Profile dataset

With different parameters we infer different outcomes

- with gender we infer that more of the profile registered are male
- With Age we can infer that People around 40 that is 30-50 have registered the most to the application.
- Distribution with respect to the Income of an individual we can infer that Income with around 50- 60k have registered the most.



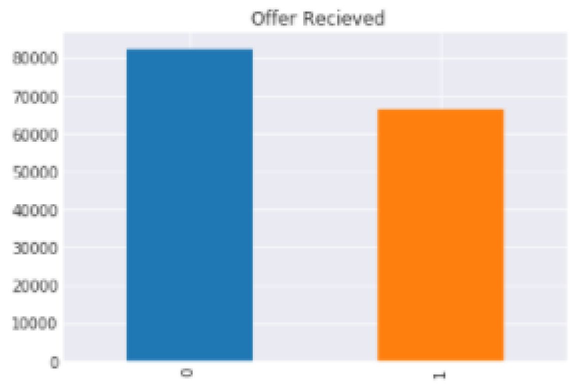
Now after the cleaning process we observe the data set with respect to offers



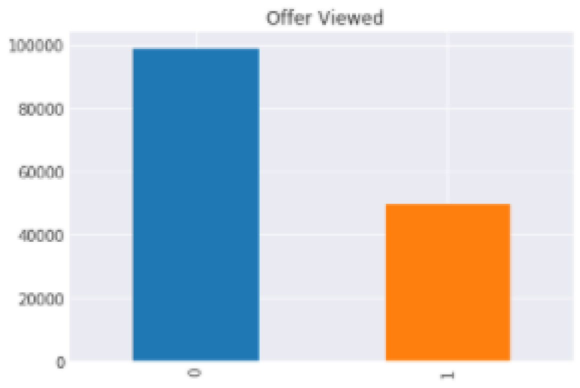
- We observe that men have ordered coffee more than Women

Exploring Offer data.

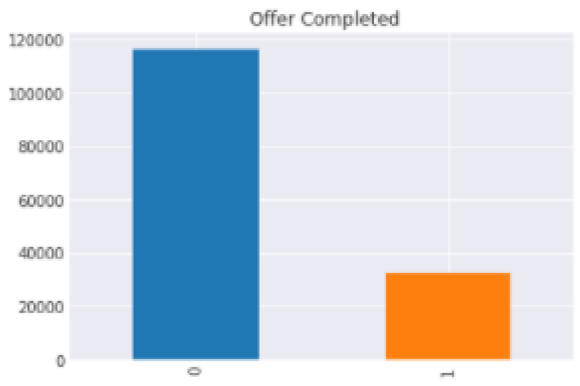
Offer Received



Offer Viewed

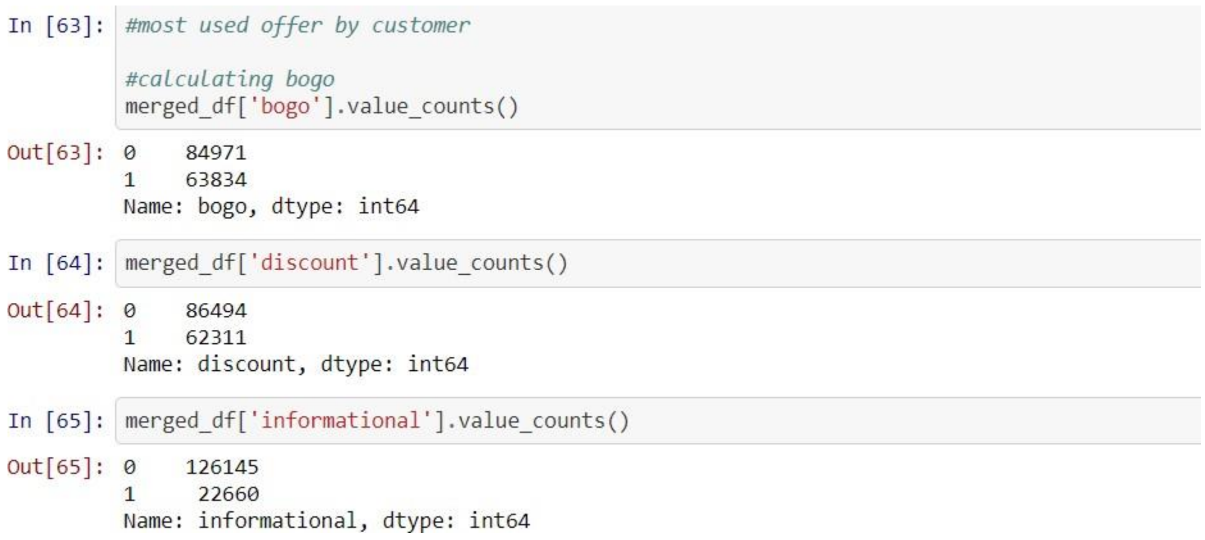


Offer completed



We can conclude that most of the offers are not viewed by the customers.

The offer that works.



```
In [63]: #most used offer by customer

         #calculating bogo
         merged_df['bogo'].value_counts()

Out[63]: 0    84971
         1    63834
         Name: bogo, dtype: int64

In [64]: merged_df['discount'].value_counts()

Out[64]: 0    86494
         1    62311
         Name: discount, dtype: int64

In [65]: merged_df['informational'].value_counts()

Out[65]: 0    126145
         1     22660
         Name: informational, dtype: int64
```

We can conclude that Buy one get one free is the offer that is redeemed the most.

Algorithms

RandomForest Algorithm

Random forest is one of the best algorithm for classification and is widely used in day to day problems. It is an ensemble classifier that uses multiple Decision Trees to obtain better accuracy and performance.

It uses many classification trees and bootstrap sample technique is used to train each tree from the set of training data. This method only searches for a random subset of variables in order to obtain a split at each node.

The basic concept is that ‘weak learners’ come together to form a ‘strong learner’. It chooses the most voted prediction as the result. It also has the ability to handle larger input datasets when compared with other methods.

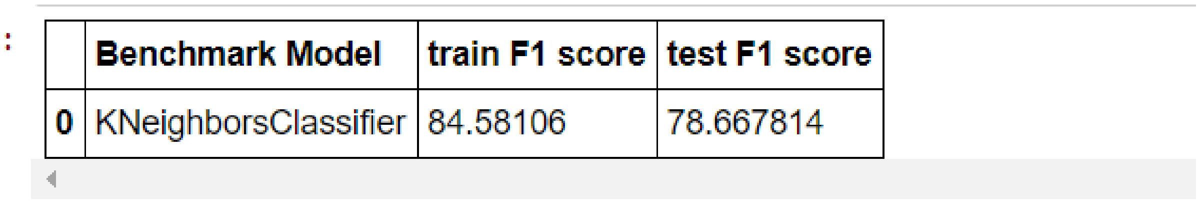
Decision Tree Algorithm:

One of the most widely used and easily understandable algorithms which is easy to implement in machine learning. Decision tree works on the principle of splitting the dataset into small parts until the dataset is no longer splitable or the target variable is the same.
The algorithm first assigns all training instances to the root of the tree and then splits the values based on the split feature with the criteria(In decision tree algorithm, gini index and information gain methods are used to calculate these nodes.). After which data is partitioned at nodes based on criteria and threshold, these partitioned values are called nodes and the above method is repeated until the tree is grown to full length or stopped forcefully.

Benchmark

As the data provided has both input and output this type of model comes in supervised learning, the model best suited for benchmark is KNeighbors Classifier as it is fast and accurate for this type of problem.

As we can see that Benchmark Model has shown a pretty good result as F1 on test score is around 80 out of 100. More the F1 score better the model.

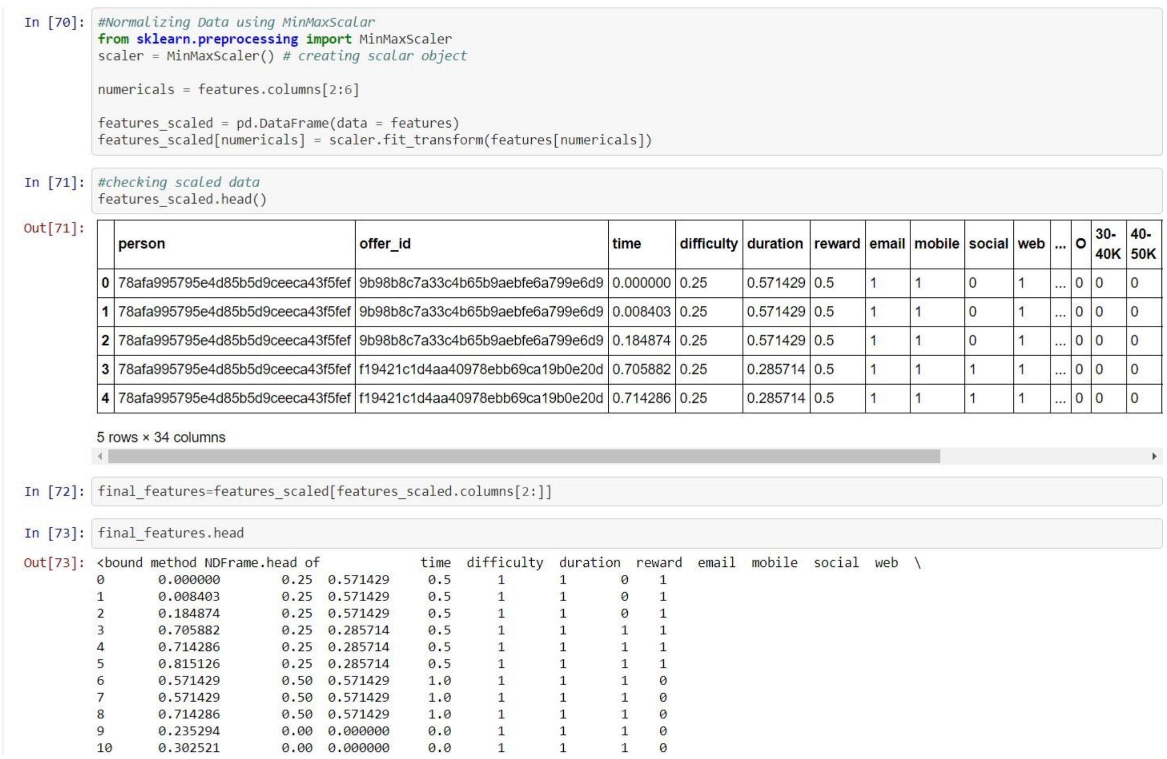


	Benchmark Model	train F1 score	test F1 score
0	KNeighborsClassifier	84.58106	78.667814

Methodology

Data Pre-processing.

- One hot encoding – For all the three data sets with one hot encoding I have removed the categorical variable in order to feed the machine learning model and for better understanding of the dataset. Every unique value in the category will be added as a feature. One-Hot Encoding is the process of creating dummy variables.
- Train and Test split - The train test split function is for splitting a single dataset for two different purposes: training and testing. The testing subset is for building your model. The testing subset is for using the model on unknown data to evaluate the performance of the model
- Normalization - Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.



```
In [70]: #Normalizing Data using MinMaxScalar
         from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler() # creating scalar object

         numericals = features.columns[2:6]

         features_scaled = pd.DataFrame(data = features)
         features_scaled[numericals] = scaler.fit_transform(features[numericals])
```

```
In [71]: #checking scaled data
         features_scaled.head()
```

Out[71]:

	person	offer_id	time	difficulty	duration	reward	email	mobile	social	web	...	O	30-40K	40-50K
0	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.000000	0.25	0.571429	0.5	1	1	0	1	...	0	0	0
1	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.008403	0.25	0.571429	0.5	1	1	0	1	...	0	0	0
2	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	0.184874	0.25	0.571429	0.5	1	1	0	1	...	0	0	0
3	78afa995795e4d85b5d9ceeca43f5fef	f19421c1d4aa40978ebb69ca19b0e20d	0.705882	0.25	0.285714	0.5	1	1	1	1	...	0	0	0
4	78afa995795e4d85b5d9ceeca43f5fef	f19421c1d4aa40978ebb69ca19b0e20d	0.714286	0.25	0.285714	0.5	1	1	1	1	...	0	0	0

5 rows × 34 columns

```
In [72]: final_features=features_scaled[features_scaled.columns[2:]]
```

```
In [73]: final_features.head
```

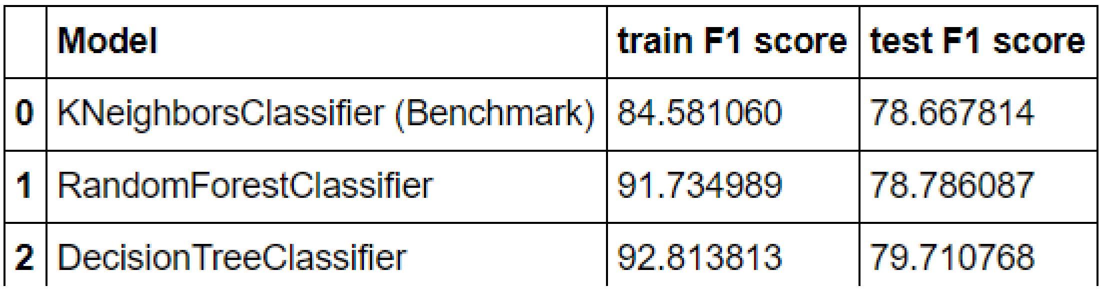
```
Out[73]: <bound method NDFrame.head of           time  difficulty  duration  reward  email  mobile  social  web  \
         0       0.000000        0.25  0.571429     0.5      1       1       0    1
         1       0.008403        0.25  0.571429     0.5      1       1       0    1
         2       0.184874        0.25  0.571429     0.5      1       1       0    1
         3       0.705882        0.25  0.285714     0.5      1       1       1    1
         4       0.714286        0.25  0.285714     0.5      1       1       1    1
         5       0.815126        0.25  0.285714     0.5      1       1       1    1
         6       0.571429        0.50  0.571429     1.0      1       1       1    0
         7       0.571429        0.50  0.571429     1.0      1       1       1    0
         8       0.714286        0.50  0.571429     1.0      1       1       1    0
         9       0.235294        0.00  0.000000     0.0      1       1       1    0
         10      0.302521        0.00  0.000000     0.0      1       1       1    0
```

Implementation

1) First the Data was cleaned individually
2) Data was merged and analysed.
3) Conclusions were drawn on the basis of EDA 4)Data was prepared for the modelling-
Data was sent for Scaling in this case Normalization.
5) Splitting the dataset into test and train data set. The data set was divided into 25% Test and 75% Training dataset.
6) Creating the Benchmark model and checking it's F1 Score.
7) Passing the data to the model RandomForest and Decision Tree and calculate the result.
8) Compare the results and Select the best model.

Results

- The reason for the model to perform better is that Splitting the data set in the 25%-75% Test Train has helped and provided better result because it had given the model the adequate amount of data for training and thus model has performed better.
- Random forest is an ensemble of decision tree algorithms. It is an extension of bootstrap aggregation (bagging) of decision trees, both of these are advanced algorithms and use better algorithms and thus overstands the benchmark model



	Model	train F1 score	test F1 score
0	KNeighborsClassifier (Benchmark)	84.581060	78.667814
1	RandomForestClassifier	91.734989	78.786087
2	DecisionTreeClassifier	92.813813	79.710768

Improvements

With models like XG Boost and Light GBM the model can be more efficient.

This project felt more real and challenging.

Thank you😊