Machine Learning Engineer Nanodegree Capstone Project(Starbucks) Report
Abhay Mahajan

Introduction

This data set contains simulated data that mimics customer behaviour 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 that is the challenge to solve with this data set.

Data Sets

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

- 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' for other 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 depending on the record

Cleaning The Datasets:

Portfolio.json:-

Renaming 'id' to 'offer_id'

Profile.json:-

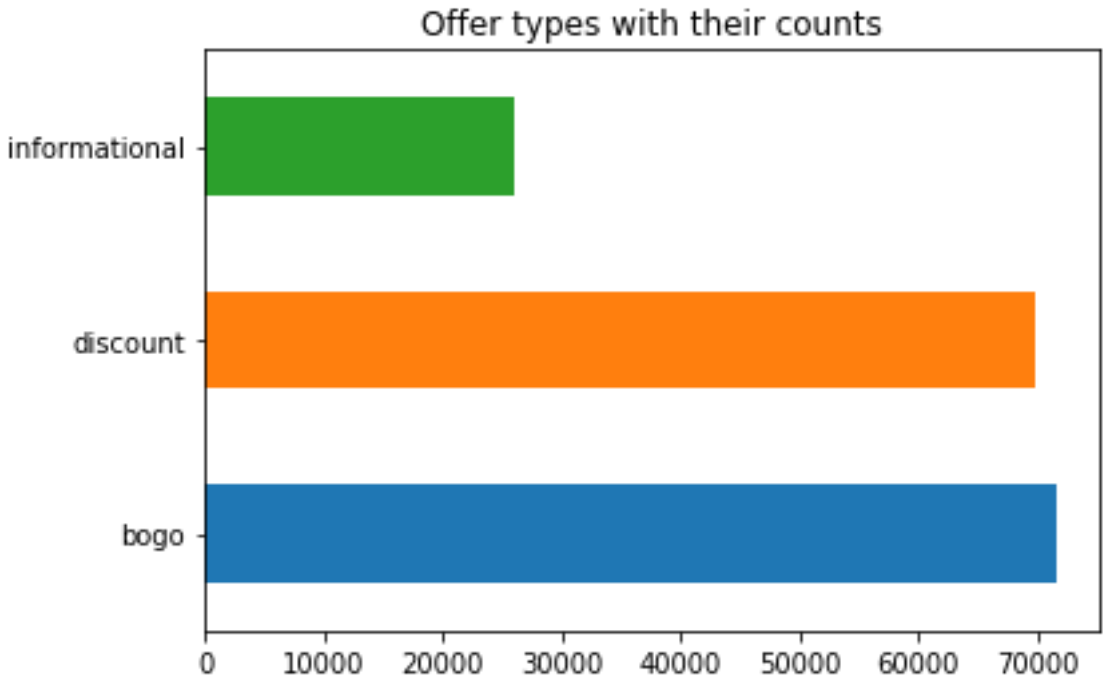
Renaming 'id' to 'customer_id' , filling the missing values of age and income with mean value , filling the missing values of gender with mode.

Transcript.json:-

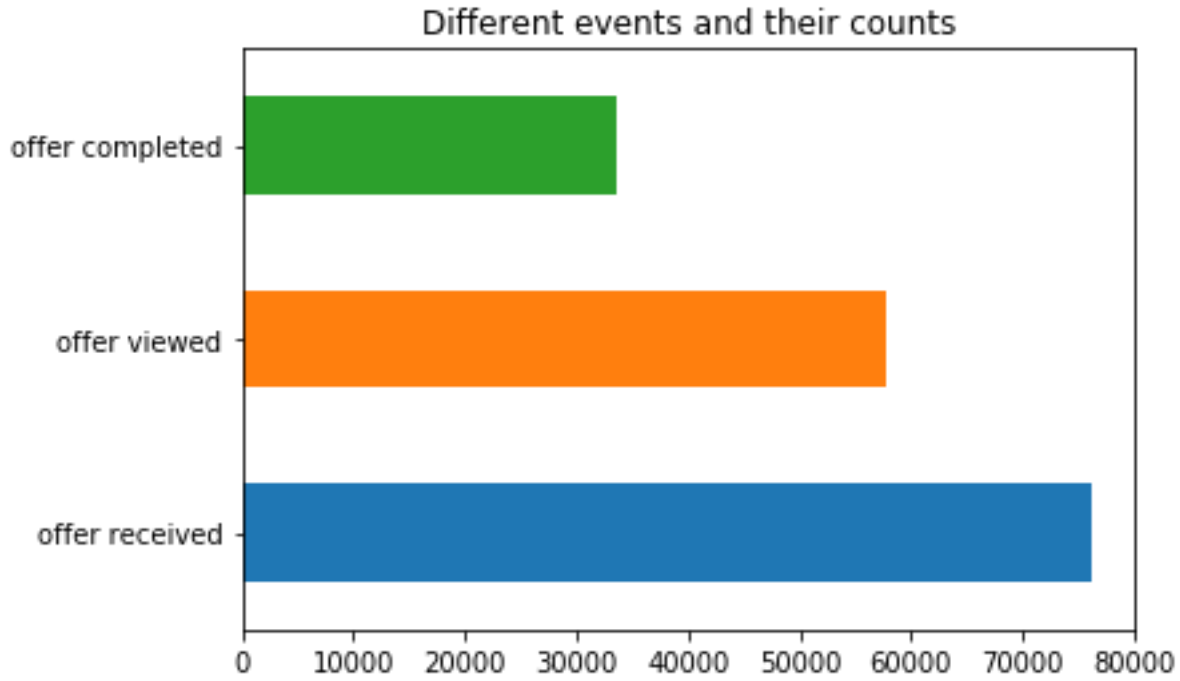
Renaming 'person' to 'customer_id' , splitting the 'value' column based on its keys and dropping the unnecessary columns and filling the missing values with 0

Exploratory Data Analysis:

Different offer types with counts:-

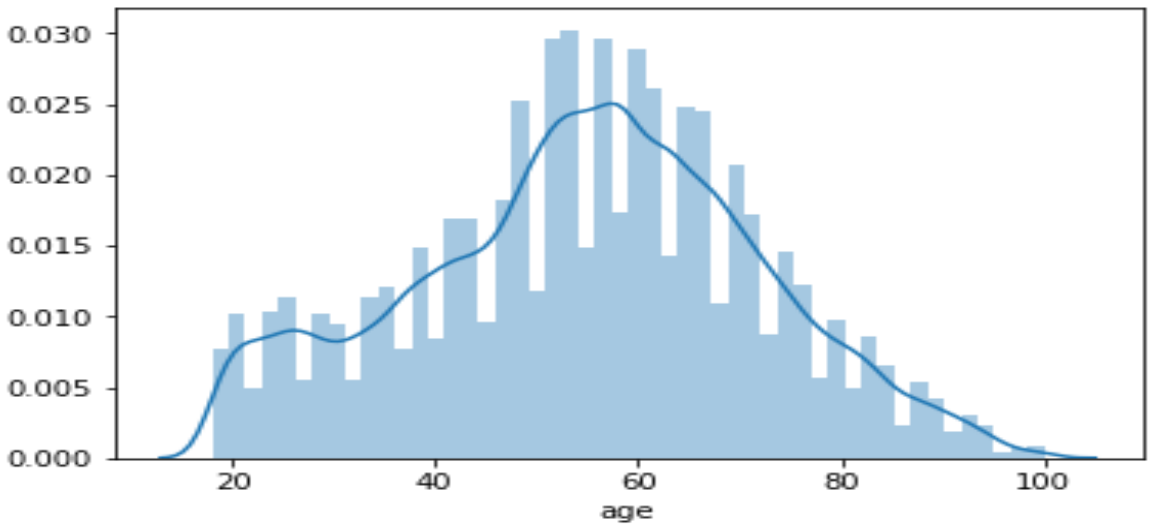


Different events with counts:-



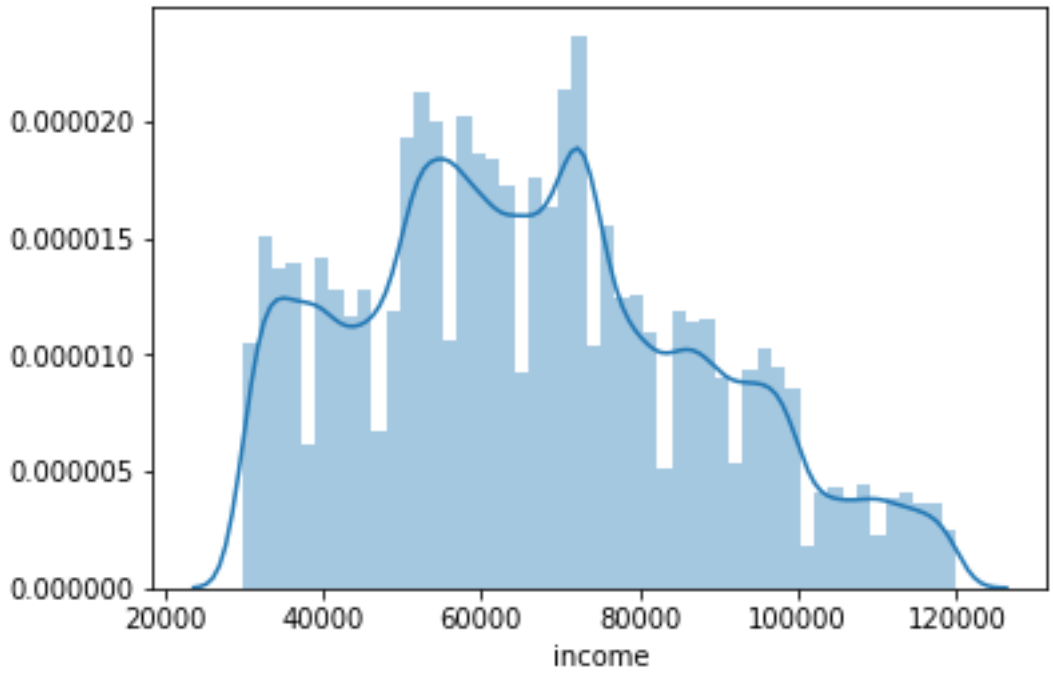
So,in most of the cases offer is received by the user and it is not completed by him/her, means most of the people just ignore the offers they receive.

Distortion plot of age:-

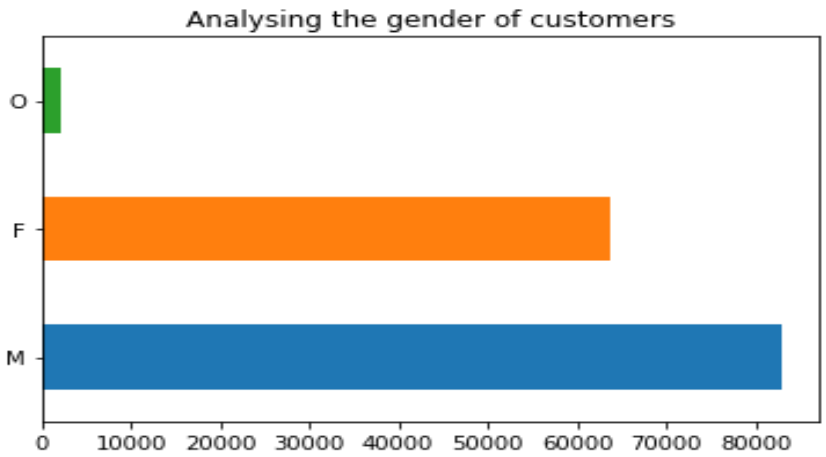


We can observe that most of the customers are within the age group of 45-60 are the most frequent customers and more than any other group,this is quite interesting.

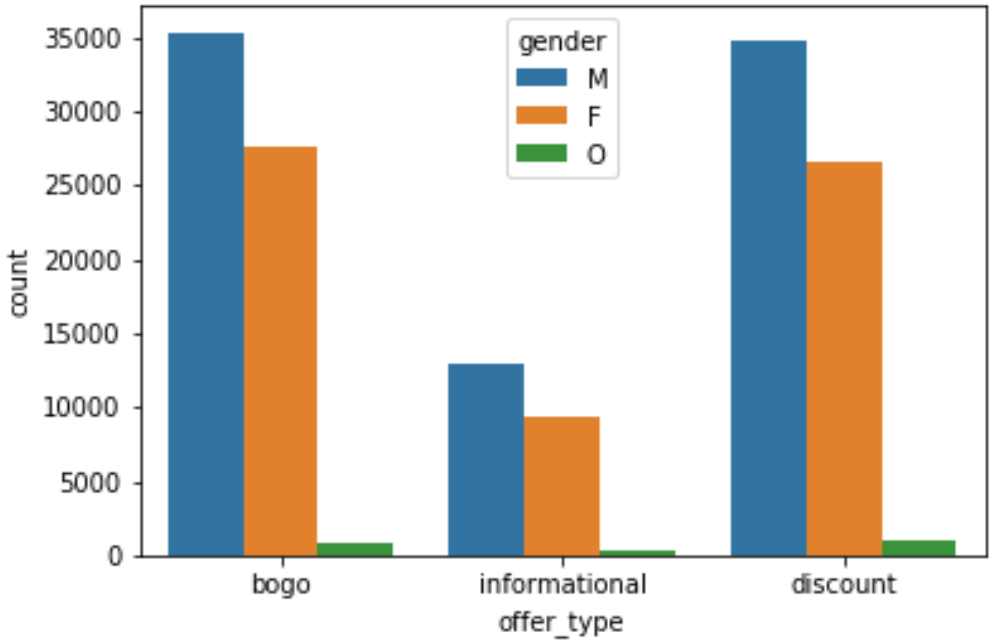
Distortion plot of income:-



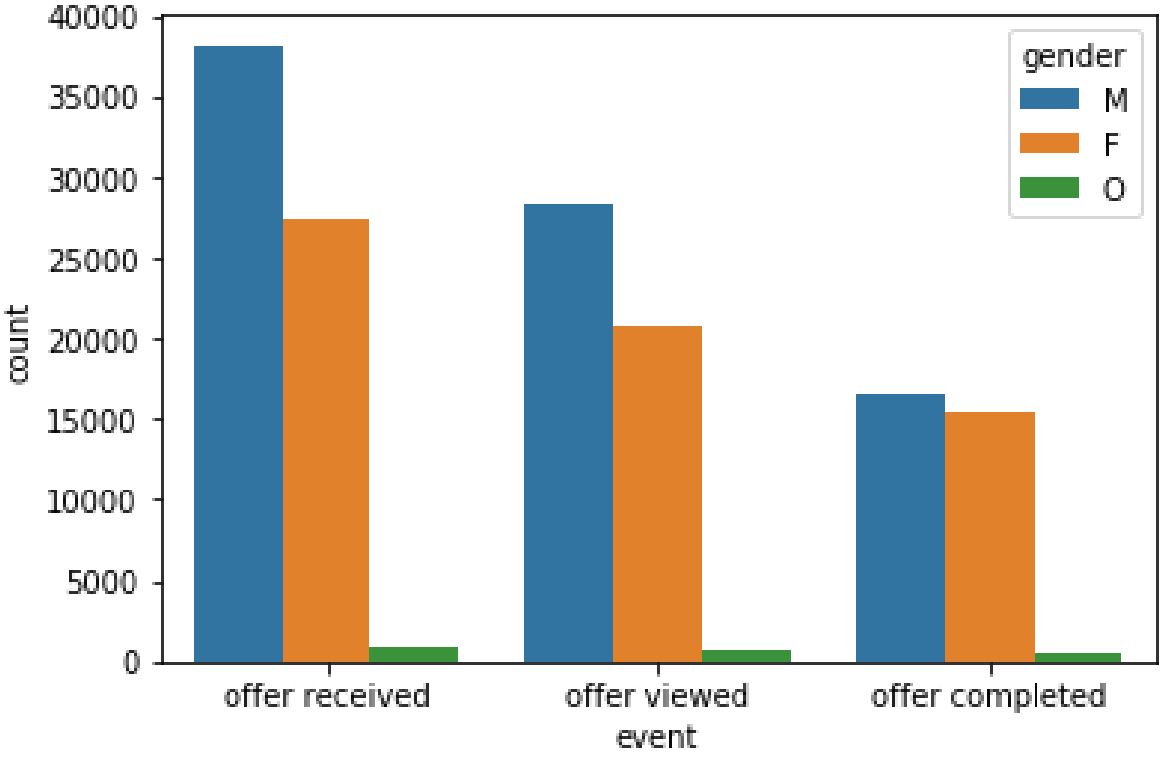
Bar plot of customers on the basis of gender:-



Count plot of offer_type on the basis of gender:-



Count plot of event on the basis of gender:-



So,from the exploratory data analysis we can see that most of the customers just receive the offers and they do not view them and the people who complete the offers they receive is quite less and most of the offers made by Starbuks are BOGO and Discount and most of the people that are the customers are within the age group of 45-60 and the most common gender is male and the people who are the customers of Starbucks have their income within the range of 55k - 75k

Making a Machine Learning Model(Preprocessing the final dataset):

We will now encode the categorical features like 'offer_type' , 'gender' , 'age'

We will encode the offer_id and customer_id

we will scale the numerical data including 'income' , 'difficulty' , 'duration' and many more...

We will encode the values in the 'event' column

Now encoding the channels column

Training Our Dataset:-

Now splitting our 'final_df' into training and test set

In [67]:
```
independent_variables = final_df2 #our dataset containing all the independent variables excluding the 'event'
dependent_variable = final_df['event'] #our final dataset containing the 'event'
```

In [68]:
```
from sklearn.model_selection import train_test_split
# splitting our dataset into training and test set and the test set being the 30% of the total dataset
x_train , x_test, y_train , y_test = train_test_split(independent_variables , dependent_variable , test_size = 0.3 , random_stat
```

In [69]:
```
x_train.shape
```

Out[69]: (104127, 104)

In [70]:
```
x_test.shape
```

Out[70]: (44627, 104)

Testing Our Dataset:-

Here we will test our training and the testing dataset on several machine learning models and will find out the best model for our dataset on the basis of F1 score,so for the purpose of calculating the predicted values and calculating the F1 scores of different models we have build a function named train_test_f1()

In [71]:
```
# We will implement a number of classification machine learning methods and will determine which method is best for our model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [72]:
```
#We will test the quality of the predicted output on a number of metrics,i.e. accuracy score,f1 score
#We will use f1 score because it considers the class imbalance pretty well as compared to the accuracy score and is the best met
from sklearn.metrics import confusion_matrix , accuracy_score , fbeta_score
def train_test_f1(model):
    """
    Returns the F1 score of training and test set of any particular model
    model : model name
    Returns
    f1_score_train : F1 score of training set
    f1_score_test : F1 score of test set
    """
    predict_train = (model.fit(x_train , y_train)).predict(x_train)
    predict_test = (model.fit(x_train , y_train)).predict(x_test)
    f1_score_train = fbeta_score(y_train , predict_train , beta = 0.5 , average = 'micro')*100
    f1_score_test = fbeta_score(y_test , predict_test , beta = 0.5 , average = 'micro')*100
    return f1_score_train , f1_score_test
```

Implementing Various Models:-

Implementing the KNN Model

In [73]:
```
knn = KNeighborsClassifier()
f1_score_train_knn , f1_score_test_knn = train_test_f1(knn)#calculating the F1 scores
```

Implementing the Logistic Regression

In [74]:
```
logistic = LogisticRegression()
f1_score_train_logistic , f1_score_test_logistic = train_test_f1(logistic)#calculating the F1 scores
```

Implementing the Random Forest Classifier

In [75]:
```
random_forest = RandomForestClassifier()
f1_score_train_random , f1_score_test_random = train_test_f1(random_forest)#calculating the F1 scores
```

Implementing the Decision Tree Classifier

In [76]:
```
decision_tree = DecisionTreeClassifier()
f1_score_train_decision , f1_score_test_decision = train_test_f1(decision_tree)#calculating the F1 scores
```

Conclusion from the Different Models and F1 Scores:-

	model_name	Training set F1 Score	Test set F1 Score
0	KNeighborsClassifier	52.274626	30.564456
1	LogisticRegression	66.573511	66.571806
2	RandomForestClassifier	93.668309	64.803818
3	DecisionTreeClassifier	94.893736	86.003989

So,from the above dataframe we can conclude that when we trained our training dataset according to the KNeighborsClassifier our model performed worst,on training the model on RandomForestClassifier ,the training set F1 score is quite good i.e. 93.58 but it

performed badly on the test set with a F1 score of 64.266 and when we trained our model on DecisionTreeClassifier our model's performance was best with the training set F1 score of 94.89 and the test set F1 score of 86.02,which means our model was able to classify between the events of offers upto a great extent.As this is a practical case study with real world dataset,we can say that our model has performed successfully.