CAPSTONE PROJECT

FOR

MACHINE LEARNING NANODEGREE

UDACITY

"STARBUCK'S CAPSTONE CHALLENGE"

BY

SHILPA S

30 November 2020

CONTENT

PROJECT DEFINITION

1. PROJECT OVERVIEW

A satisfied customer is the best business strategy of all. The study of customer behaviour and patterns highly influence the leveraging of business. This is often difficulty to analyse a lot of data and to find out the right predictions. But the hottest use of Machine Learning in this problem will solve the issue by developing an appropriate model to analyse the customer influence, thus to attract and retain their customers.

STARBUCKS is one of the well-known coffeehouse chain in the world, who always strive to give their customers the best experience and best service. STARBUCKS have already implemented a free application for customers in order to make online orders, predict the waiting time and to receive offers. This project is an analysis of customer insights to offer them the best reward/promotional offer on the basis of their previous response and transactions in Starbucks app.

2. PROBLEM STATEMENT

The data files provided by STARBUCKS need to be processed to find which offer kind offer should be send to the particular customer. That simply means STARBUCKS need to send customised offer to their customer. The customers are classified on the bases of their transactions recorded in STARBUCKS APP for the last 30 days.

The classification is based on the offer type. On analysing the files we can see, there are 3 types of offers: bogo, discount and informational. These particular offers have different offer id with specific duration, reward, difficulty and noted as 'exact offers'. The bogo offer have 4 different offer id and are mentioned as bogo 1, bogo 2, bogo 3 and bogo 4. Likewise discount also have 4 offer ids mentioned as discount 1, discount 2, discount 3 and discount 4. The informational offers have only 2 kind of offer ids –informational 1 and informational 2.

The transcript file shows the transaction of each customers to theirs offers as well as transaction amount. The events in the transcript file provide how the customer responded to the offers. On analysing that a definition to 'effective offer', 'interested offer' and 'ineffective offer' are derived and explained below.

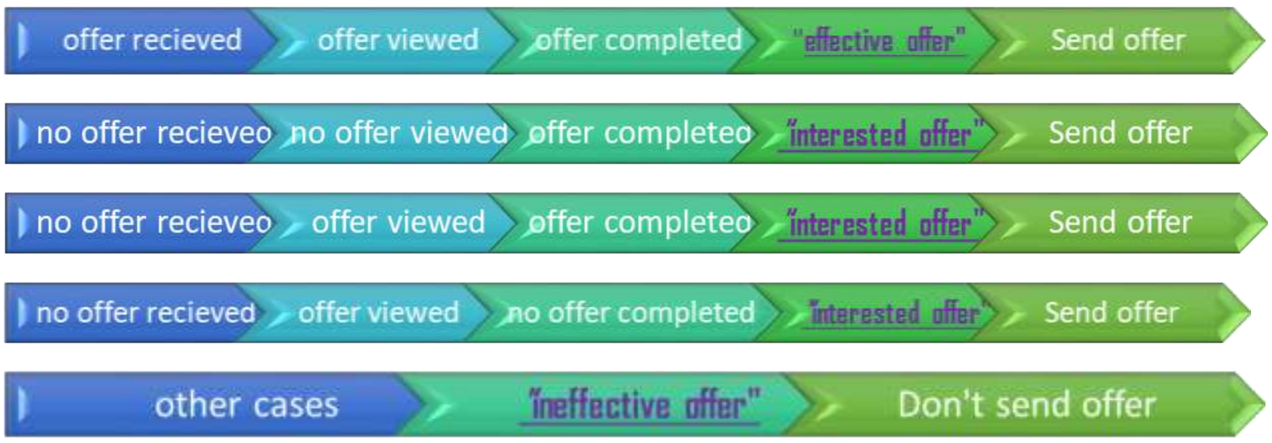


Figure 1: Strategy for defining solution

The response to ‘exact offer’ is encoded to find ‘effective offer’, ’interested offer’ and ‘ineffective offer’. These categorisation of customers will be helpful to arrive the conclusion to whom which offer should be send. The data is processed through various steps and all the steps will be explained in the section ‘DATA PREPROCESSING’. After all these preprocessing we can arrive at a conclusion that which offer (exact offer) should be send to which customer. This is simply explained in the flowsheet below.

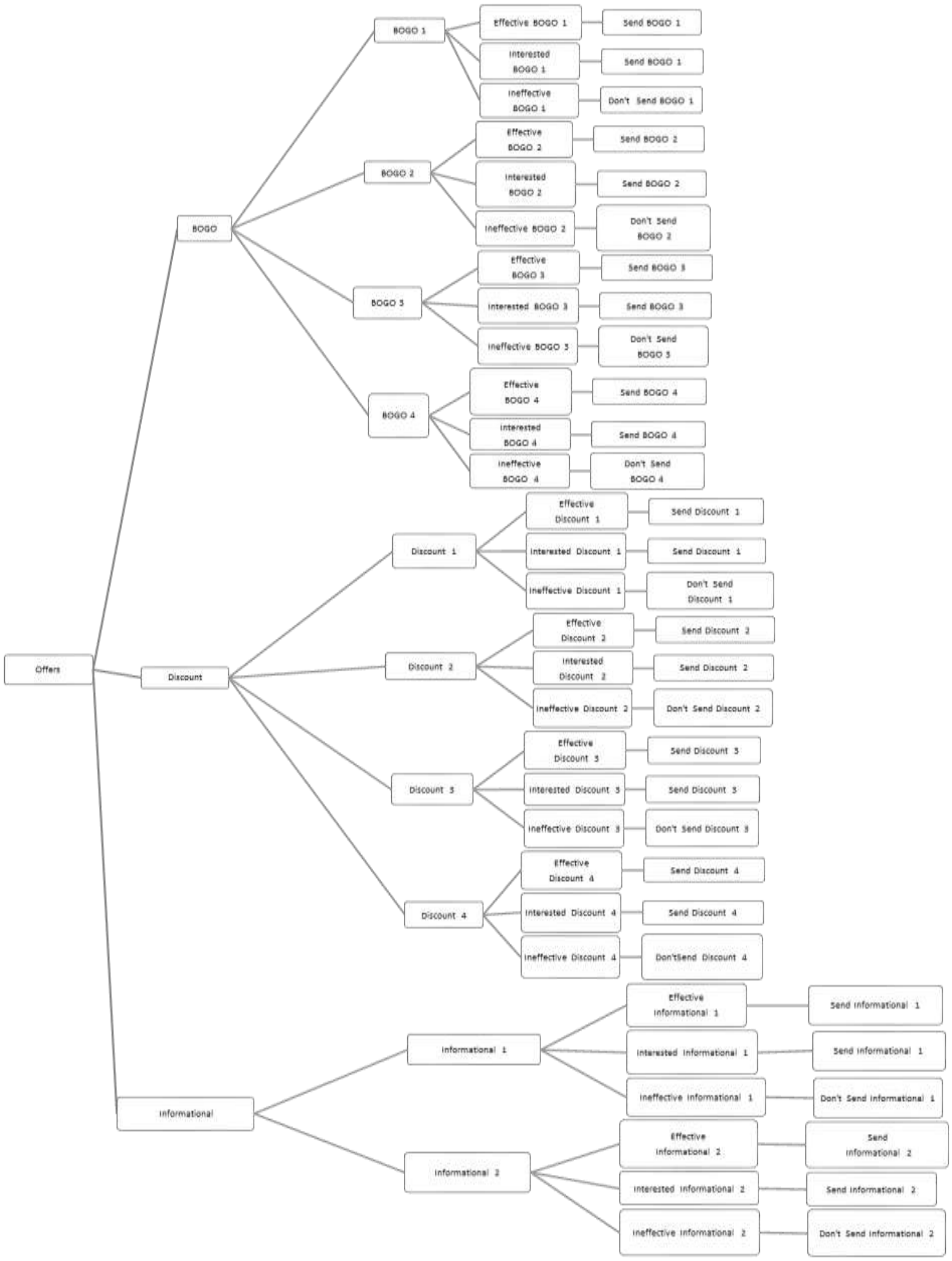
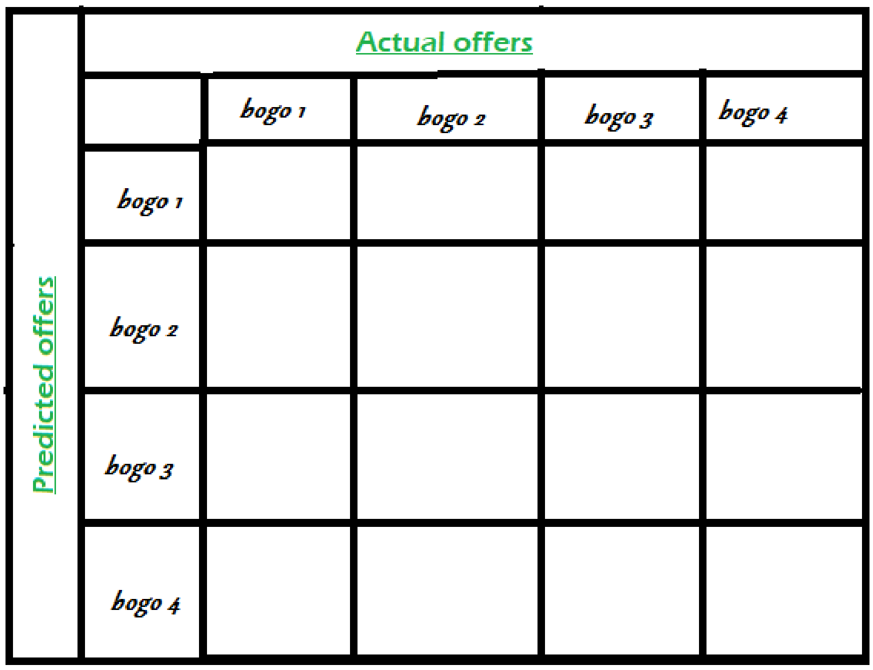


Figure 2: Flowsheet showing the problem solving.

3. METRIC

This is problem of sending customized offer messages to customers is a classification problem. The performance of the model can be evaluated by its accuracy and confusion matrix.

Confusion matrix is a technique for summarizing the performance of a classification algorithm. It is used to illustrate classifier performance based on the four values (True Positive, False Positive, True Negative, False Negative).



		Actual offers			
		bogo 1	bogo 2	bogo 3	bogo 4
Predicted offers	bogo 1				
	bogo 2				
	bogo 3				
	bogo 4				

Multi labelled confusion matrix for bogo offes

4. DATA EXPLORATION

.

4.1 DATASETS AND INPUT

STARBUCKS/UDACITY has provided 3 data sources which contains the simulated data that mimics the customer data on the Starbucks mobile application .The dataset files give are listed below

1. profile.json
2. portfolio.json
3. transcript.json

PROFILE.JSON

This source file contain the demographic data for each customer. (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)

PORTFOLIO.JSON

This file contains the list of all available offers sent during 30-day test period (10 offers x 6 fields)

- 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)

This dataset contains different offers that Starbucks is looking to send to customers.
The offers are explained here:

1.BUY-ONE-GET-ONE(BOGO) : This offer is a reward that enables the customer to obtain an extra or equal product for free.
2. Discounts: This offer gives the customer certain percentage off to the original cost of the product on purchasing.
3. Informational: This is a message or promotion that might contain a piece of content to inform the customers that certain product are available again.

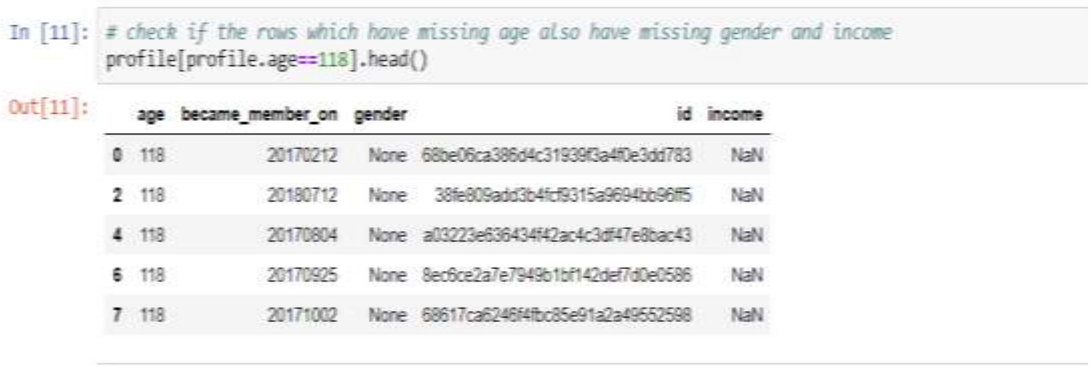
TRANSCRIPT.JSON

This file contains records for transactions, offers received, offers viewed, and offers completed

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 three files provided by STARBUCK: portfolio, profile and transcript are merge and named as ‘customer tracking’.The demographics features of ‘customer tracking’ includes customer id, age, age group (age category encoded), gender (male/female encoded), income, income category (income category encoded), channels, email, web, mobile, social (channels encoded), membership days, offer type, offer id, offer received, offer viewed, offer completed, transaction, amount (events encoded).

On the data exploration of ‘customer tracking’ some null rows are found in income, gender and an abnormality in age value found is 118.These data are eliminated.



```
In [11]: # check if the rows which have missing age also have missing gender and income
         profile[profile.age==118].head()

Out[11]:
   age  became_member_on  gender                                id  income
0  118          20170212    None  68be06ca386d4c31939f3a4f0e3dd783     NaN
2  118          20180712    None  38fe809add3b4fcf9315a9694bb96ff5     NaN
4  118          20170804    None  a03223e636434f42ac4c3df47e8bac43     NaN
6  118          20170925    None  8ec6ce2a7e7949b1bf142def7d0e0586     NaN
7  118          20171002    None  68617ca6246f4fbc85e91a2a49552598     NaN
```

Fig 3. Output displaying rows with missing value

The ‘customer tracking’ file is separated into 4: bogo file, discount file, informational file and transaction file. As far now, we are considering only the offer files not the transaction file.

5. DATA VISUALIZATION

The bogo file, discount file, informational file and transaction file are analysed in separate Jupyter Notebook because of hardware constrains. For each file and its exact offer category, few feature distribution are visualised and given below.

5.1 DATA VISUALIZATION FOR BOGO OFFERS

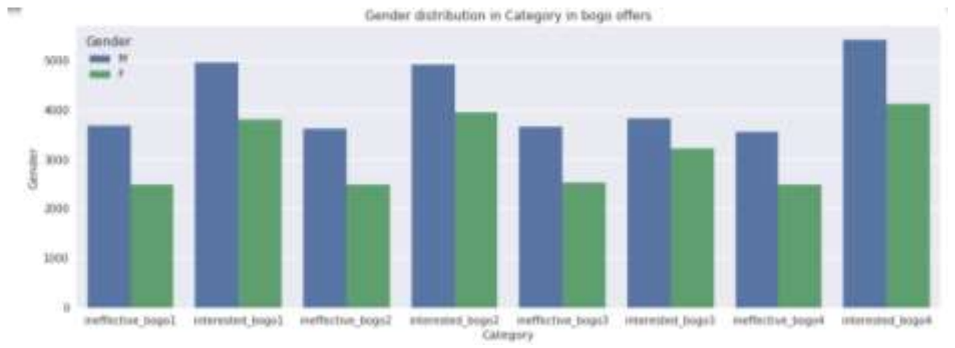


Fig 4. Gender distribution of category in bogo offers

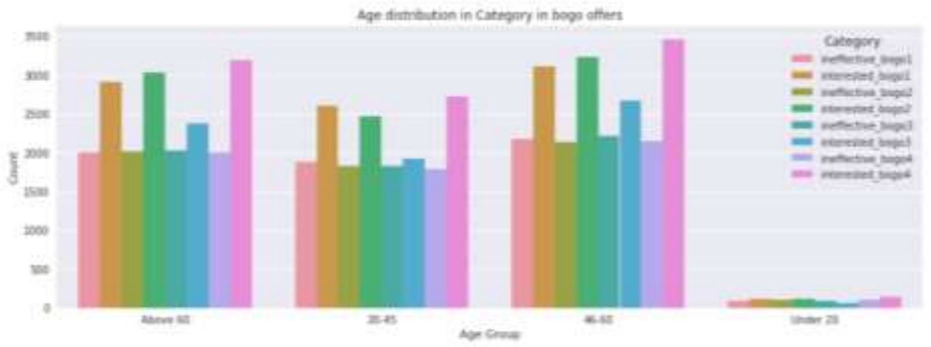


Fig 5. Age distribution of category in bogo offers

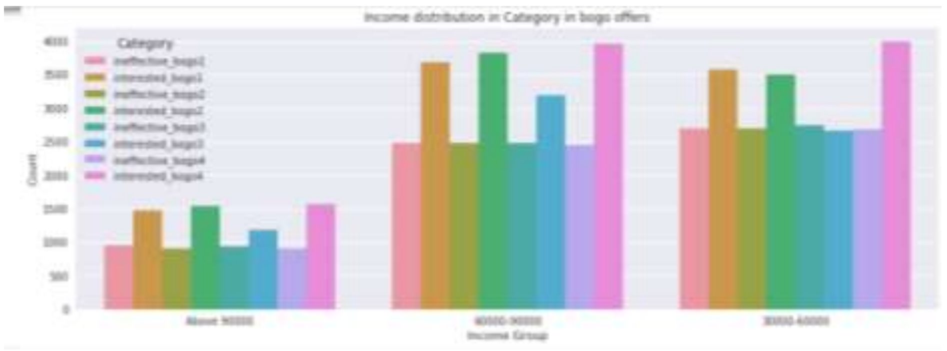


Fig 6. Income distribution of category in bogo offers

5.2 DATA VISUALISATION FOR DISCOUNT OFFERS

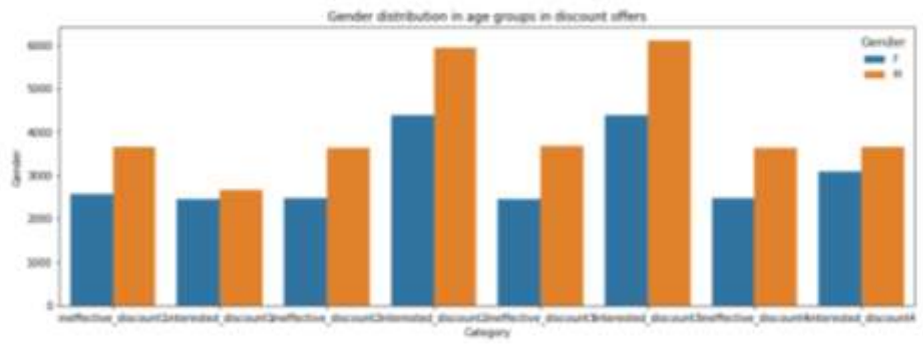


Fig 7. Gender distribution of category in discount offers

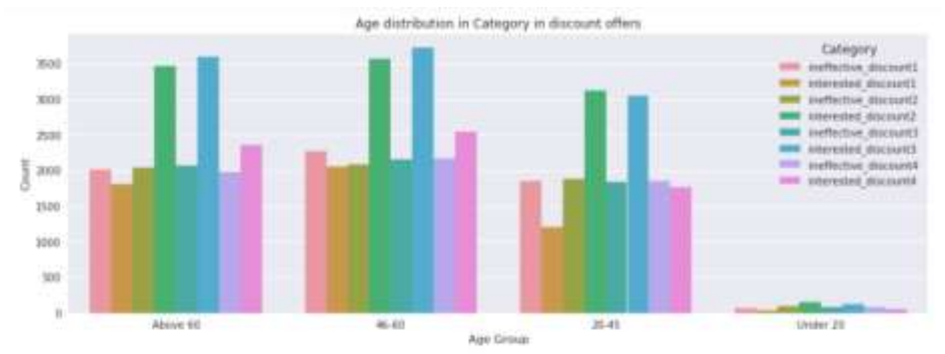


Fig 8. Age distribution of category in discount offers

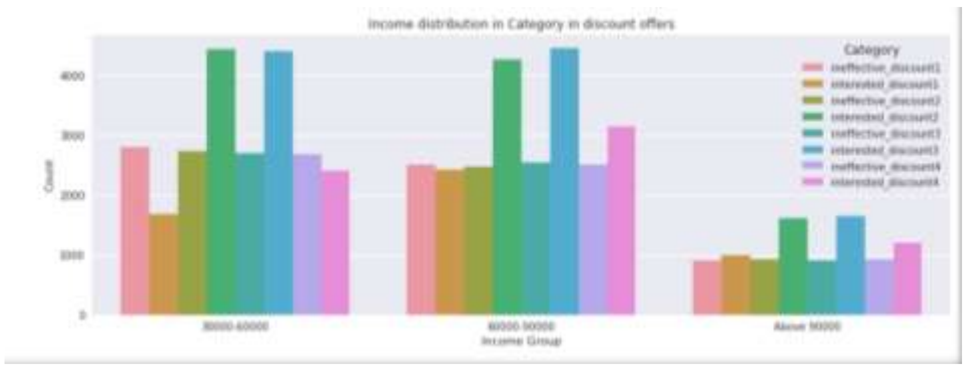


Fig 9. Income distribution of category in discount offers

5.3 DATA VISUALISATION OF INFORMATIONAL OFFERS

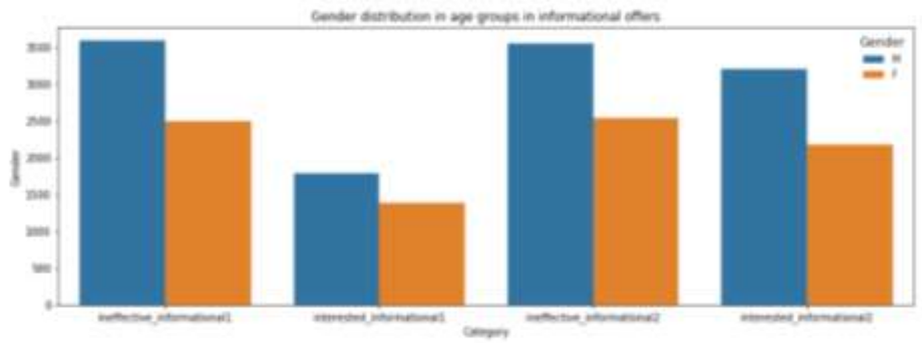


Fig 10. Gender distribution of category in informational offers

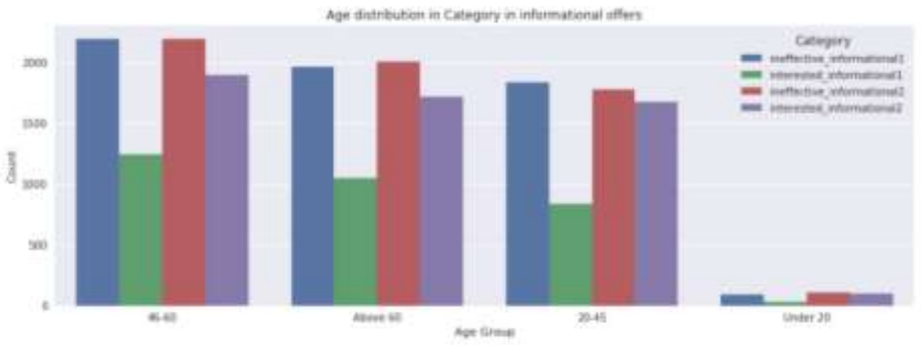


Fig 11. Age distribution of category in informational offers

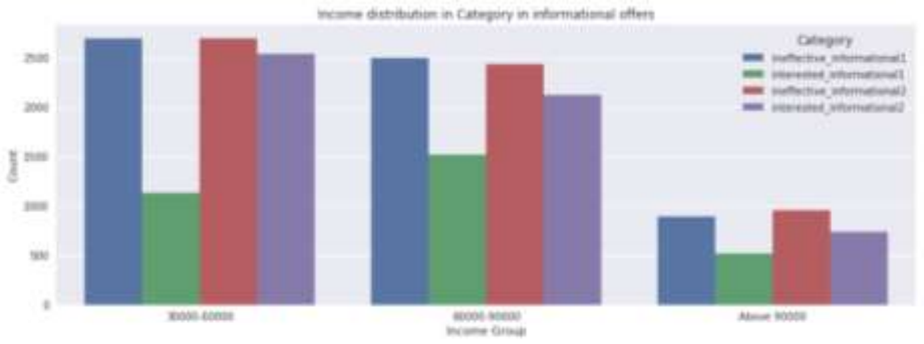
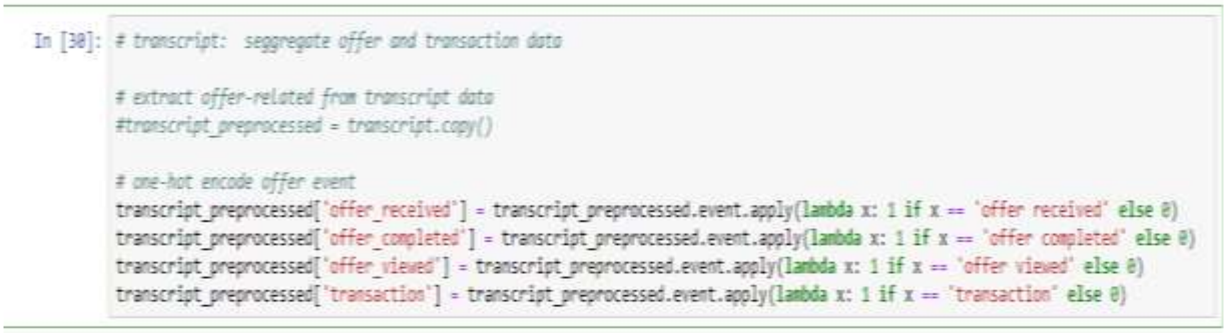


Fig 12. Income distribution of category in informational offers

6. DATA PREPROCESSING AND DATA PREPARATION.

The following points are the way, the data is processed and prepared for training and testing.

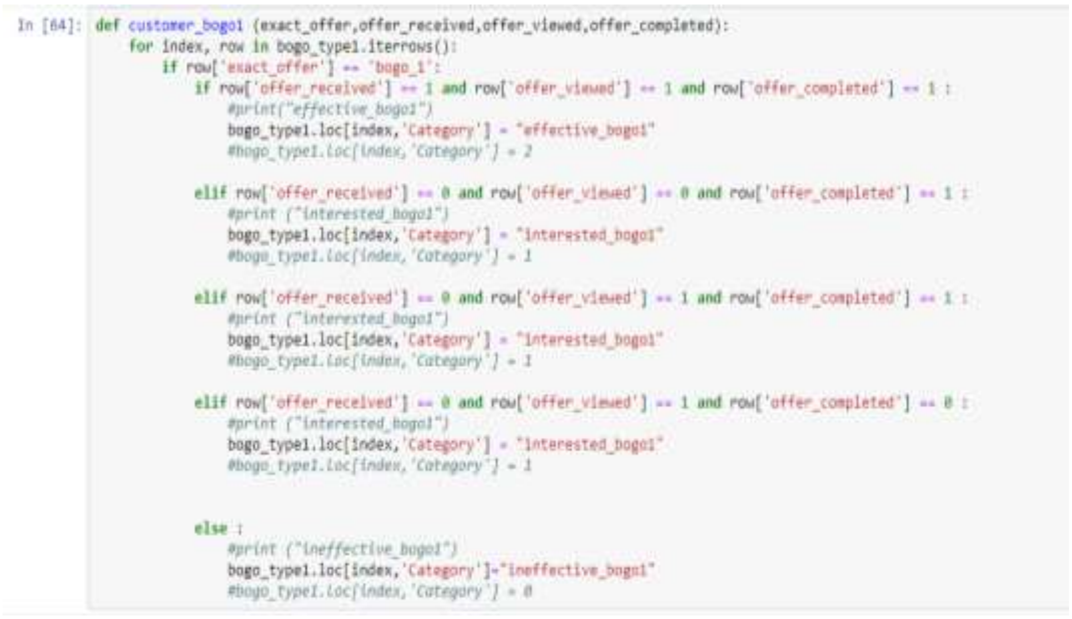
1. The primary analysis of three file given by STARBUCKS is conducted to find the info, null values, and abnormal values.
2. An issue identified in the profile file is few customers with Age = 118 and respective income and gender of the customer is null value. These values are eliminated.
3. The transcript file, portfolio and profile is merged (merge how = left) and a single file is obtained:' customer tracking.
4. The customer tracking file is analysed. This is a classification problem and the features need to be encoded.
5. The Age is categorised and encoded as well as income, gender and channels also encoded.
6. The events are encoded into offer type, offer id, offer received, offer viewed, offer completed, transaction, amount



```
In [38]: # transcript:  seggregate offer and transaction data

         # extract offer-related from transcript data
         #transcript_preprocessed = transcript.copy()

         # one-hot encode offer event
         transcript_preprocessed['offer_received'] = transcript_preprocessed.event.apply(lambda x: 1 if x == 'offer received' else 0)
         transcript_preprocessed['offer_completed'] = transcript_preprocessed.event.apply(lambda x: 1 if x == 'offer completed' else 0)
         transcript_preprocessed['offer_viewed'] = transcript_preprocessed.event.apply(lambda x: 1 if x == 'offer viewed' else 0)
         transcript_preprocessed['transaction'] = transcript_preprocessed.event.apply(lambda x: 1 if x == 'transaction' else 0)
```

Fig 13. Code for one-hot encoding of events

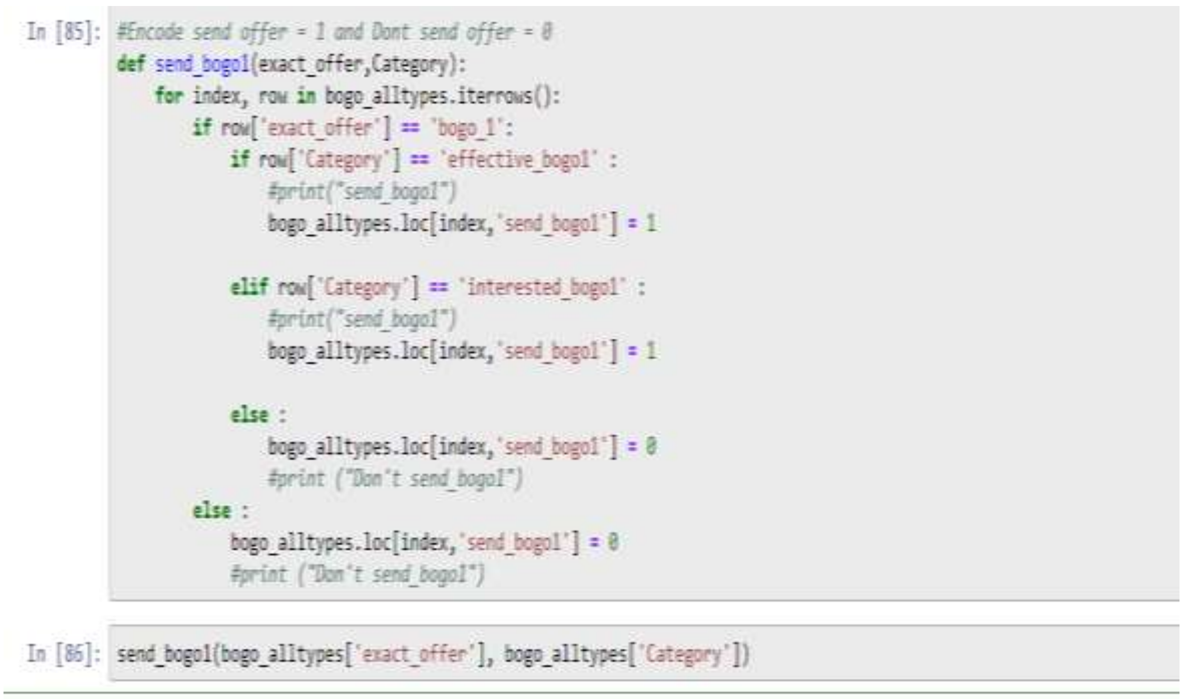
7. The 'customer tracking' file is separated into 4: bogo file, discount file, informational file and transaction file. The bogo file includes the transactions recorded for bogo 1, bogo 2, bogo 3 and bogo 4. Likewise the discount file contains the recorded transactions of discount 1, discount 2, discount 3 and discount 4. Thus the informational file includes the informational 1 and informational 2 details
8. With the problem solving strategy mentioned in the flow sheet (fig.1) 'effective offer' 'interested offer' and 'ineffective offer' are identified.



```
In [64]: def customer_bogo1 (exact_offer,offer_received,offer_viewed,offer_completed):
             for index, row in bogo_type1.iterrows():
                 if row['exact_offer'] == 'bogo_1':
                     if row['offer_received'] == 1 and row['offer_viewed'] == 1 and row['offer_completed'] == 1 :
                         #print("effective_bogo1")
                         bogo_type1.loc[index,'Category'] = "effective_bogo1"
                         #bogo_type1.loc[index,'Category'] = 2

                     elif row['offer_received'] == 0 and row['offer_viewed'] == 0 and row['offer_completed'] == 1 :
                         #print ("interested_bogo1")
                         bogo_type1.loc[index,'Category'] = "interested_bogo1"
                         #bogo_type1.loc[index,'Category'] = 1

                     elif row['offer_received'] == 0 and row['offer_viewed'] == 1 and row['offer_completed'] == 1 :
                         #print ("interested_bogo1")
                         bogo_type1.loc[index,'Category'] = "interested_bogo1"
                         #bogo_type1.loc[index,'Category'] = 1

                     elif row['offer_received'] == 0 and row['offer_viewed'] == 1 and row['offer_completed'] == 0 :
                         #print ("interested_bogo1")
                         bogo_type1.loc[index,'Category'] = "interested_bogo1"
                         #bogo_type1.loc[index,'Category'] = 1


                     else :
                         #print ("ineffective_bogo1")
                         bogo_type1.loc[index,'Category']="ineffective_bogo1"
                         #bogo_type1.loc[index,'Category'] = 0
```

Fig 14. Example code for bogo 1 for classifying the customer category

9. The particular offer should be send to the customer if the response is an effective offer or interested offer and the offers need not send to ineffective offers



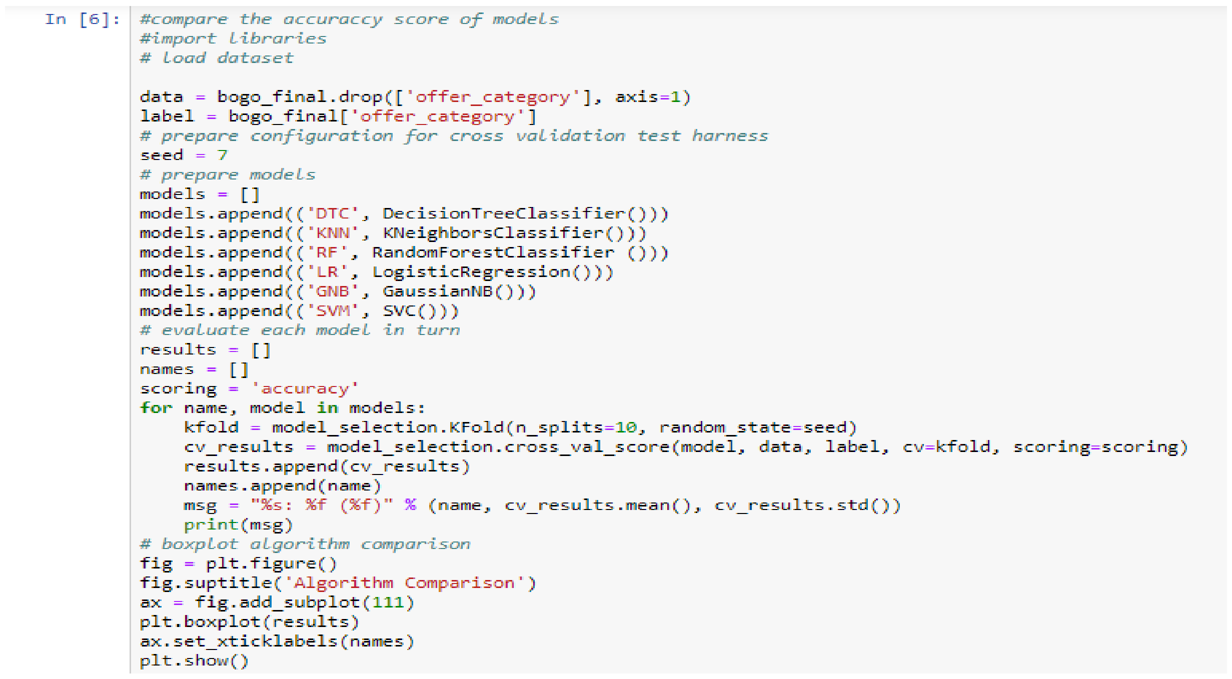
```
In [85]: #Encode send offer = 1 and Dont send offer = 0
         def send_bogo1(exact_offer,Category):
             for index, row in bogo_alltypes.iterrows():
                 if row['exact_offer'] == 'bogo_1':
                     if row['Category'] == 'effective_bogo1' :
                         #print("send_bogo1")
                         bogo_alltypes.loc[index,'send_bogo1'] = 1

                     elif row['Category'] == 'interested_bogo1' :
                         #print("send_bogo1")
                         bogo_alltypes.loc[index,'send_bogo1'] = 1

                     else :
                         bogo_alltypes.loc[index,'send_bogo1'] = 0
                         #print ("Don't send_bogo1")
                 else :
                     bogo_alltypes.loc[index,'send_bogo1'] = 0
                     #print ("Don't send_bogo1")

In [86]: send_bogo1(bogo_alltypes['exact_offer'], bogo_alltypes['Category'])
```

Fig 15. Example code for bogo 1 encoding offer to send and don't send

10. Unwanted columns are dropped and final files for each group is prepared for training and testing.

7. MODEL EVALUATION

The final data is tested and trained in various classifiers : Decision Tree Classifier, KNeighbors Classifier, Random Forest Classifier, Logistic Regression, Gaussian Naïve Base, Support Vector Machine



```
In [6]: #compare the accuraccy score of models
        #import Libraries
        # Load dataset

        data = bogo_final.drop(['offer_category'], axis=1)
        label = bogo_final['offer_category']
        # prepare configuration for cross validation test harness
        seed = 7
        # prepare models
        models = []
        models.append(('DTC', DecisionTreeClassifier()))
        models.append(('KNN', KNeighborsClassifier()))
        models.append(('RF', RandomForestClassifier ()))
        models.append(('LR', LogisticRegression()))
        models.append(('GNB', GaussianNB()))
        models.append(('SVM', SVC()))
        # evaluate each model in turn
        results = []
        names = []
        scoring = 'accuracy'
        for name, model in models:
            kfold = model_selection.KFold(n_splits=10, random_state=seed)
            cv_results = model_selection.cross_val_score(model, data, label, cv=kfold, scoring=scoring)
            results.append(cv_results)
            names.append(name)
            msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
            print(msg)
        # boxplot algorithm comparison
        fig = plt.figure()
        fig.suptitle('Algorithm Comparison')
        ax = fig.add_subplot(111)
        plt.boxplot(results)
        ax.set_xticklabels(names)
        plt.show()
```

Fig 16. Code for the Model Comparison

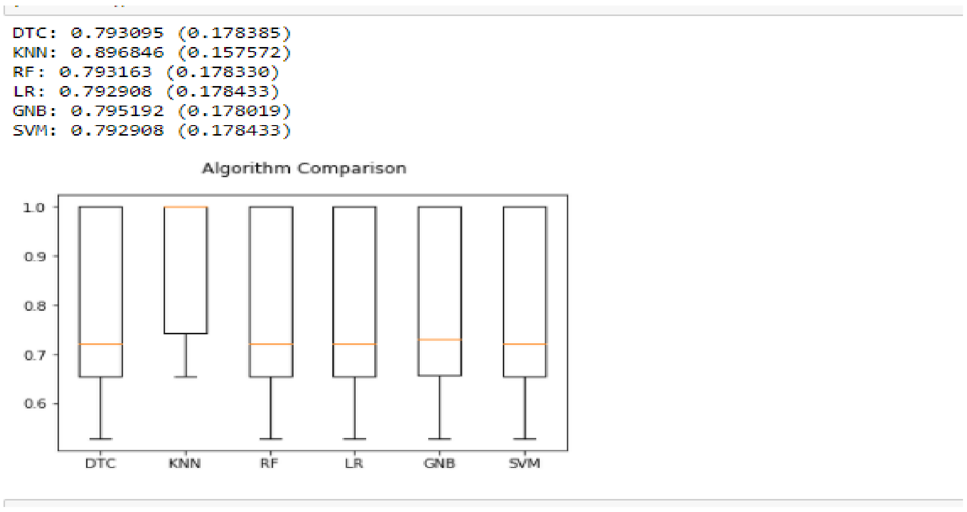
The bogo, discount and informational data are trained and tested separately. The accuracy score and confusion matrix is considered for the evaluation of model. With the above given code recommends the suitable model on comparison with others.

The confusion matrix with less false negative should be recommended to leverage the business. For example if the customer sis interested in bogo 1 but the model predicts it as bogo 4.There will be a loss in business.

8. RESULTS

8.1 BOGO OFFER MODEL

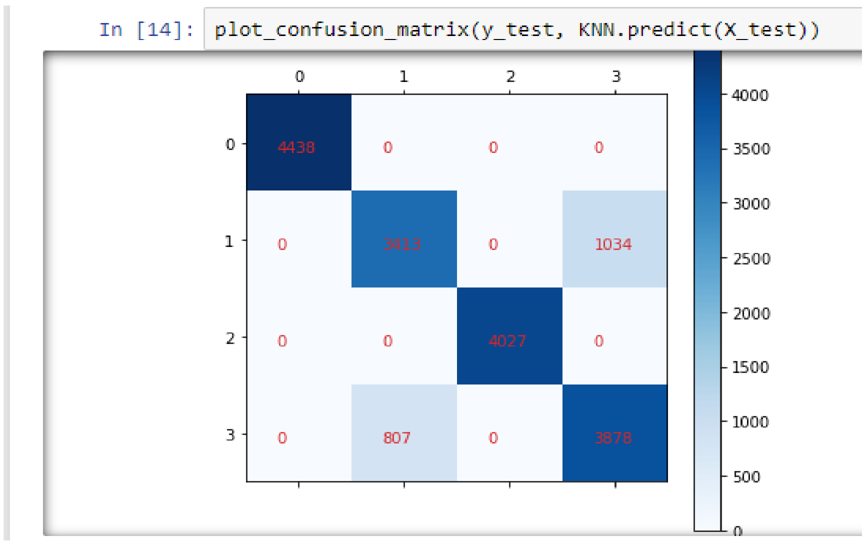
#Comparison of model : output



```
DTC: 0.793095 (0.178385)
KNN: 0.896846 (0.157572)
RF: 0.793163 (0.178330)
LR: 0.792908 (0.178433)
GNB: 0.795192 (0.178019)
SVM: 0.792908 (0.178433)
```

Fig 17: Model comparison output of BOGO offer model

#Confusion matrix of recommended model

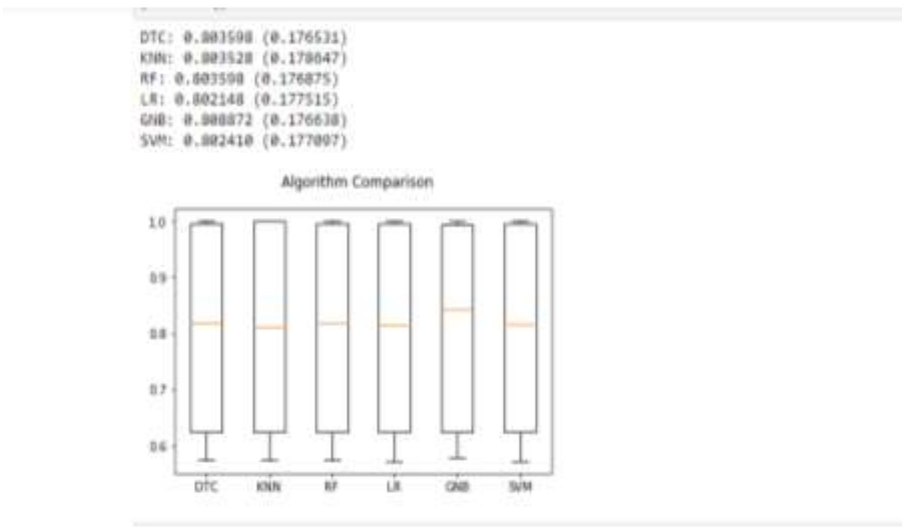


```
In [14]: plot_confusion_matrix(y_test, KNN.predict(X_test))
```

Fig 18: Confusion matrix of KNN – recommended model for BOGO offers

8.2 DISCOUNT OFFER MODEL

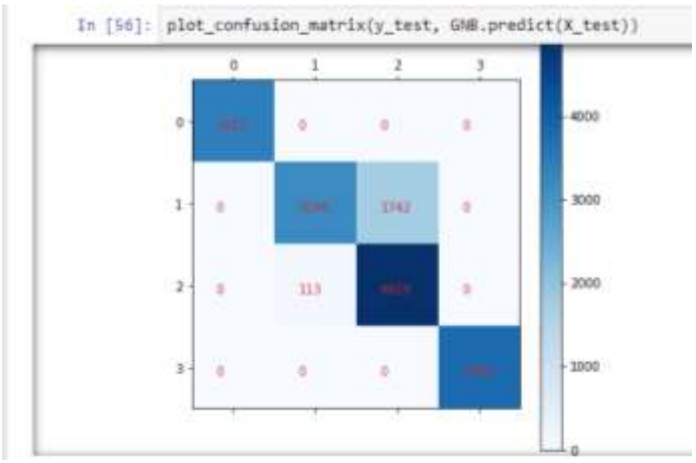
#Comparison of model : output



```
DTC: 0.803598 (0.176531)
KNN: 0.803528 (0.178047)
RF: 0.803598 (0.176875)
LR: 0.802148 (0.177515)
GNB: 0.808872 (0.176638)
SVM: 0.802410 (0.177097)
```

Fig 19: Model comparison output of DISCOUNT offer model

#Confusion matrix of recommended model

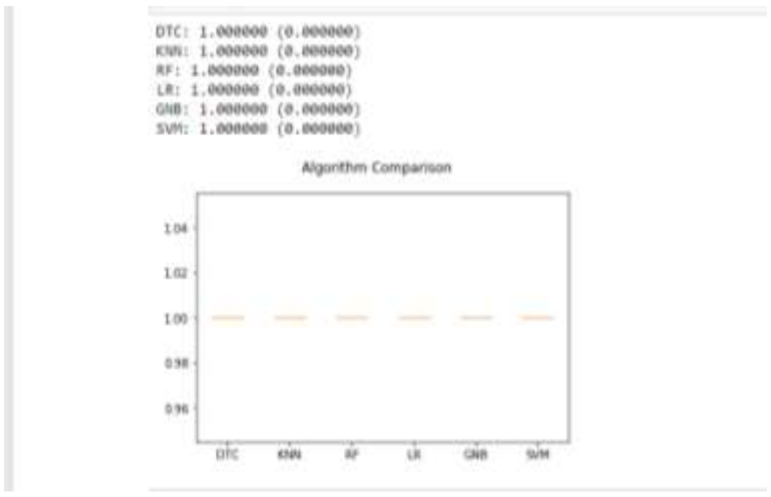


```
In [56]: plot_confusion_matrix(y_test, GNB.predict(X_test))
```

Fig 20 : Confusion matrix of GAUSSIAN NAÏVE BASE – recommended model for DISCOUNT offers

8.3 INFORMATIONAL OFFER MODEL

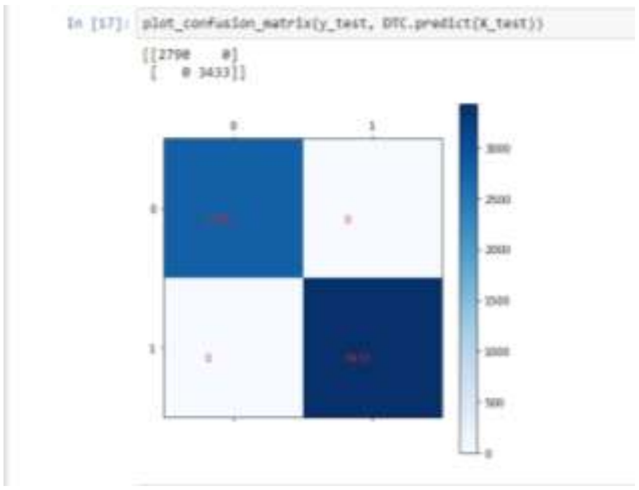
#Comparison of model : output



```
DTC: 1.000000 (0.000000)
KNN: 1.000000 (0.000000)
RF: 1.000000 (0.000000)
LR: 1.000000 (0.000000)
GNB: 1.000000 (0.000000)
SVM: 1.000000 (0.000000)
```

Fig 21: Model comparison output of INFORMATIONAL offer model

#Confusion matrix of recommended model



```
In [17]: plot_confusion_matrix(y_test, DTC.predict(X_test))
[[2798    0]
 [   0 3433]]
```

Fig 22: Confusion matrix of Decision Tree Classifier – recommended model for INFORMATIONAL offers

9. CONCLUSION

The conclusions arrived through this project are listed below:

- On the business point of view, confusion matrix is the right method of evaluation of model
- The offers are tested and trained separately, this will be helpful when we scale up the data set.
- The recommended model algorithm for BOGO offer is KNeighbors Classifier with 89% accuracy
- The recommended model algorithm for DISCOUNT offer is Gaussian Naïve Base Classifier with 80.8% accuracy.
- In the case of INFORMATIONAL offers we have only 20000 data set with two category and the model shows 100% accuracy. This may vary when we scale up the data.
- So the recommended model algorithm for INFORMATIONAL offer is Decision Tree Classifier which is the benchmark model.

10. IMPROVEMENTS

The transaction file is not considered in the offer section and assumed as they are not interested in offers but these customers can also classified to the respective offer category according to their demographic features. Another way of improving this project is to find out which channel is suitable for a particular customer and thus increase the conversion rate of offers.