Machine Learning Engineer Nanodegree

Capstone Project

- Project overview:

Starbucks Corporation is an American multinational chain of coffeehouses, Starbucks 'goal is to give their customers best service, so Starbucks provide an app for customers to make online purchases in willing to increase sales and customers satisfaction. 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).

This project aims to find the best offer to be send to the customer by predicting if the customer will complete the offer or not.

- Datasets and Inputs:

- **portfolio.json** : containing offer ids and meta data about each offer
- **profile.json** : demographic data for each customer
- **transcript.json** : records for transactions, offers received, viewed, and completed

1- portfolio.json size = (10 rows , 6 features)

- 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)

2- profile.json size = (17000 records,5 features)

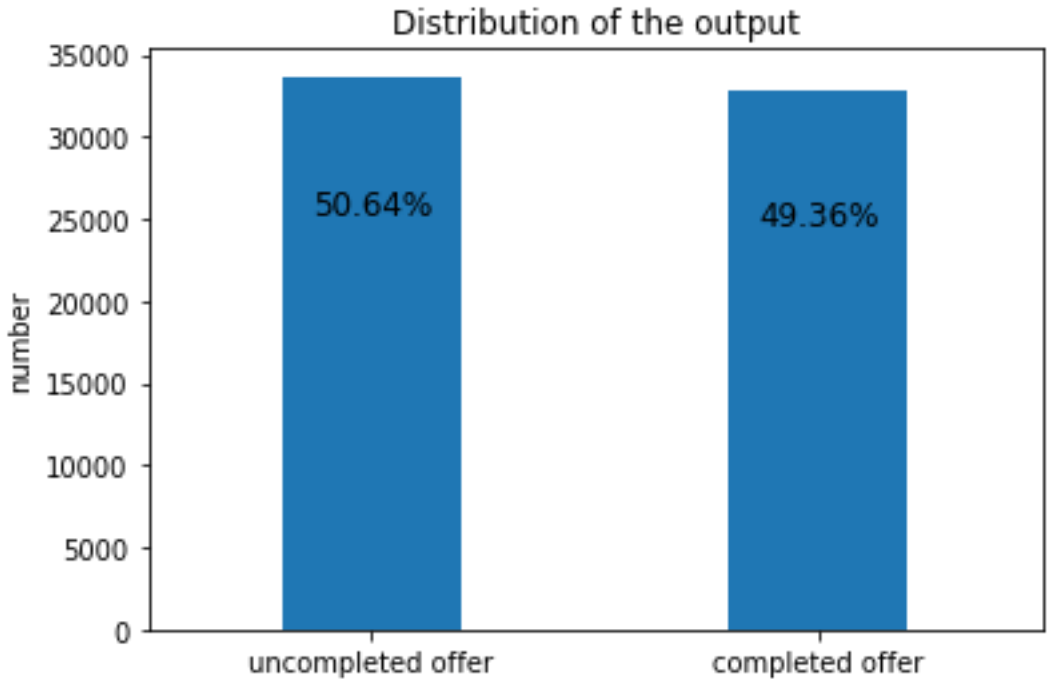
- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- gender (str) - gender of the customer
- id (str) - customer id
- income (float) - customer's income

3- transcript.json size = (306534 records,4 features)

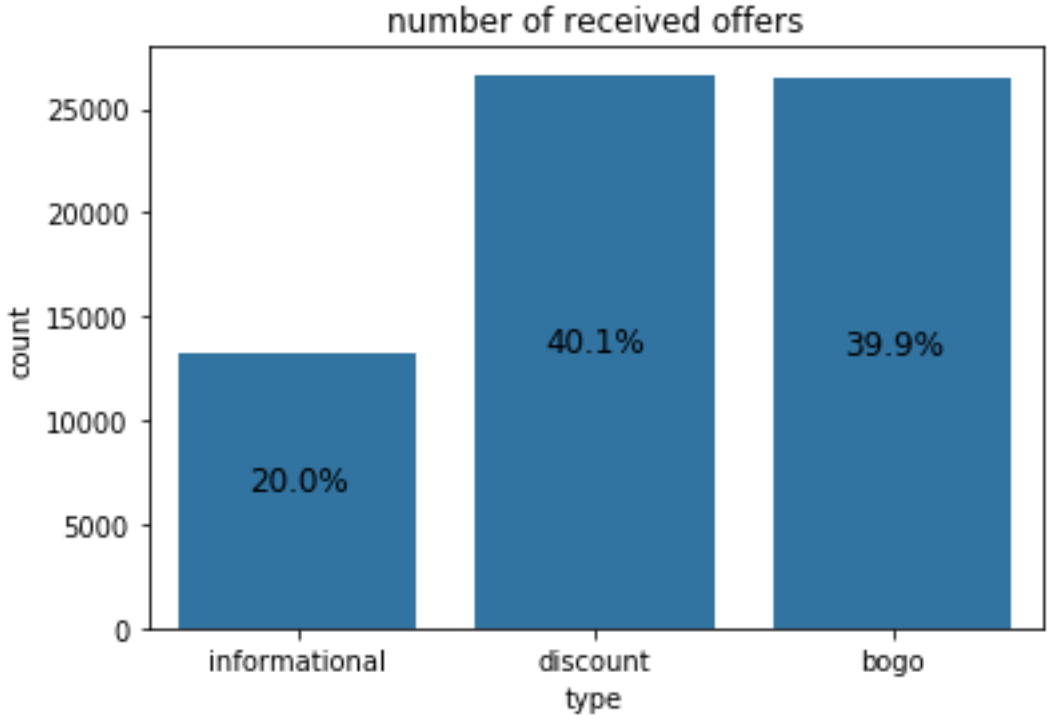
- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record

- Inputs:

- The main data will be transcript.json and I will join all another data on it, the input that I will use will be all records that related to offer receive so we will short our data to only these records and for every user I will aggregate all transaction he/she made before he/she received the offer.



- Types of offers:



- Problem Statement:

Starbucks wants to find the best offer to send to customers based on demographics information and past transactions that user made, so we can do that if we can predict if the user will complete the offer or not and as the output is determined so it will be supervised learning problem and as output is completed or not completed will be a classification problem so our solution will be building a classification model can predict if the customer will complete the offer or not so we could know if the offer is suitable to the customer or not so we can send the best offer to customers.

- Solution Statement:

Mu solution is building a machine learning model that can predict if the user will complete the offer or not, by using multiple algorithm as logistic regression, SVM, gradient boosting, XGBoost and random forest and after tuning them, I build a stack model training on the predicted probability of used models.

- Evaluation Metrics:

The distribution of output is balanced as the number of uncompleted offers is 336771 and completed offer is 32824 so we use accuracy as evaluation metrics with F1 to make sure that the difference between recall and precision is small

- Benchmark Model:

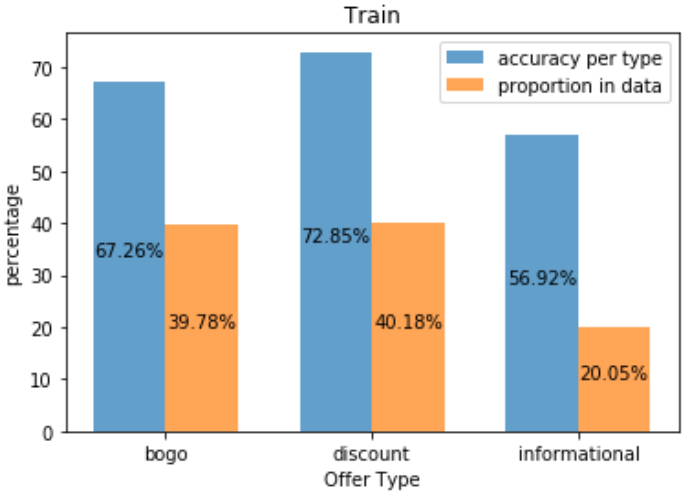
I used a logistic regression model as a benchmark model with regularization L1 and I used it during features engineering step do make sure that every feature was added to the model will be helpful and due to L1 it will be multiple features with coefficient equal zero that helped me to understand if the features is helpless or not and finally coefficient of features helped me in features selection when I tuned the others model to the see the effect of removing these features from input data

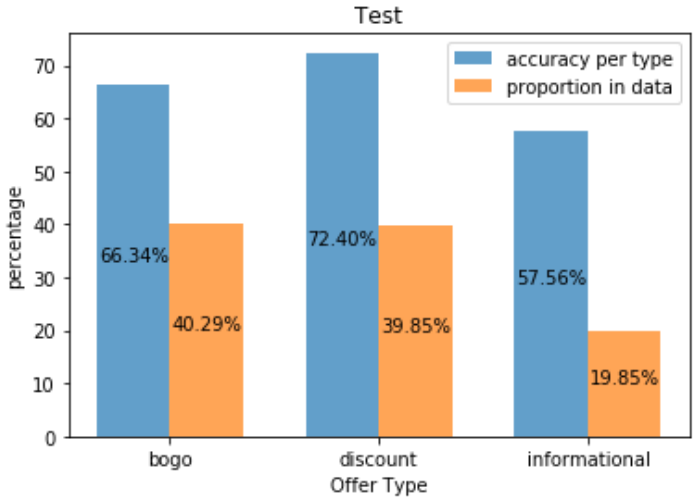
LogisticRegression(random_state=0, penalty='l1', max_iter=100, solver='liblinear', C=.7)

- Number of iterations is quite reasonable and achieved the purpose of using this model
- regularization parameter is equal to 0.7 to avoid overfitting

- Result of benchmark model before feature engineering step:

```
train_acc 0.6743057644110275
test_acc 0.6700950318777817
```

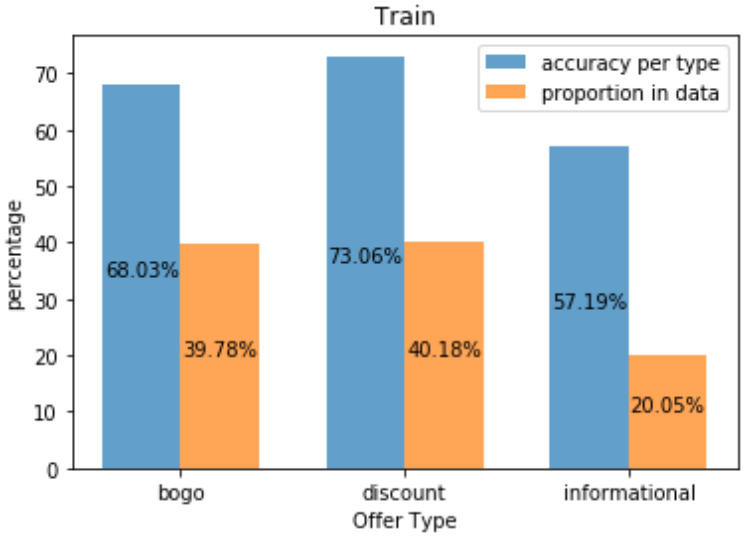


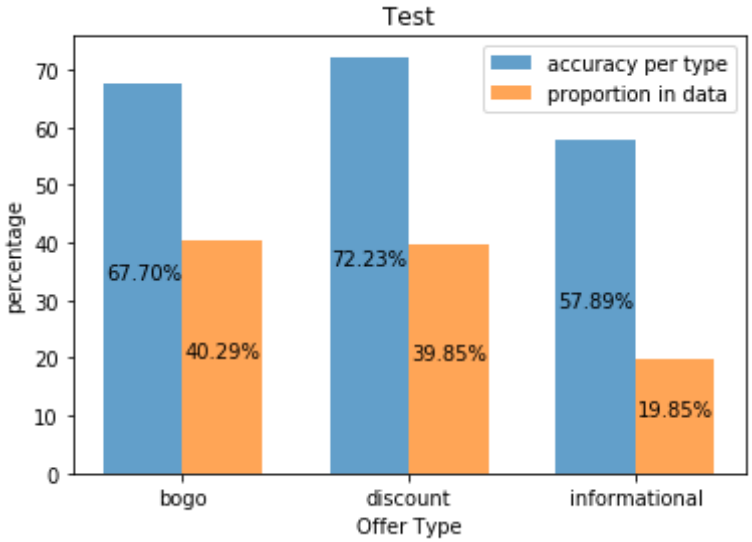


- Result of benchmark model after feature engineering step:

```
train_acc 67.876
precision 0.679
recall 0.679
f1 0.679

test_acc 67.557
precision 0.676
recall 0.676
f1 0.676
```





- Data Exploration & Preprocessing:

1- Profile file:

- There is 2175 null value in income and gender, and it is also associated with unreasonable number for age
- data type of become_member_on is float so I converted it to dataetime

Before

	age	became_member_on	income
count	17000.000000	1.700000e+04	14825.000000
mean	62.531412	2.016703e+07	65404.991568
std	26.738580	1.167750e+04	21598.299410
min	18.000000	2.013073e+07	30000.000000
25%	45.000000	2.016053e+07	49000.000000
50%	58.000000	2.017080e+07	64000.000000
75%	73.000000	2.017123e+07	80000.000000
max	118.000000	2.018073e+07	120000.000000

After

	age	income
count	14825.000000	14825.000000
mean	54.393524	65404.991568
std	17.383705	21598.299410
min	18.000000	30000.000000
25%	42.000000	49000.000000
50%	55.000000	64000.000000
75%	66.000000	80000.000000
max	101.000000	120000.000000

-As we can see the max number in age is 118 and count in income is 14825 as well as gender

- I solved that by dropping all rows with null values

2- Transcript file:

	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0
1	a03223e636434f42ac4c3df47e8bac43	offer received	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}	0
2	e2127556f4f64592b11af22de27a7932	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	0
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}	0
4	68617ca6246f4fbc85e91a2a49552598	offer received	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}	0

- first, I separated the dictionary in value column to offer id and amount

	person	event	time	amount	offer_id
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	a03223e636434f42ac4c3df47e8bac43	offer received	0	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	e2127556f4f64592b11af22de27a7932	offer received	0	NaN	2906b810c7d4411798c6938adc9daaa5
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	0	NaN	fafdcd668e3743c1bb461111dcafc2a4
4	68617ca6246f4fbc85e91a2a49552598	offer received	0	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0

- second, as every record represent offer(receive view complete) or transaction (amount), I melted these columns(amount,offer_id) and dropped records with null value and every offer_id is associated with its type

	person	event	time	type	value
0	0011e0d4e6b944f998e987f904e8c1e5	offer received	0	informational	3f207df678b143eea3cee63160fa8bed
1	0020c2b971eb4e9188eac86d93036a77	offer received	0	discount	fafdcd668e3743c1bb461111dcafc2a4
2	003d66b6608740288d6cc97a6903f4f0	offer received	0	informational	5a8bc65990b245e5a138643cd4eb9837
3	00426fe3ffde4c6b9cb9ad6d077a13ea	offer received	0	informational	5a8bc65990b245e5a138643cd4eb9837
4	005500a7188546ff8a767329a2f7c76a	offer received	0	bogo	ae264e3637204a6fb9bb56bc8210ddfd

- third I joined portfolio (offers properties) with this data frame

	person	event	time	type	value	reward	difficulty	duration
0	0011e0d4e6b944f998e987f904e8c1e5	offer received	0	informational	3f207df678b143eea3cee63160fa8bed	NaN	NaN	4.0
1	0020c2b971eb4e9188eac86d93036a77	offer received	0	discount	fafdcd668e3743c1bb461111dcafc2a4	2.0	10.0	10.0
2	003d66b6608740288d6cc97a6903f4f0	offer received	0	informational	5a8bc65990b245e5a138643cd4eb9837	NaN	NaN	3.0
3	00426fe3ffde4c6b9cb9ad6d077a13ea	offer received	0	informational	5a8bc65990b245e5a138643cd4eb9837	NaN	NaN	3.0
4	005500a7188546ff8a767329a2f7c76a	offer received	0	bogo	ae264e3637204a6fb9bb56bc8210ddfd	10.0	10.0	7.0

- forth I kept only record with event offer received as those represent the offer I want to predict if it will be completed or not

- forth, for every received offer I aggregated all offers and transactions that customer made before he received this offer

- Aggregated past offers and transactions for every customer:
1- I calculated number of every transaction [received, view, completed] related to each offer type and first and last time of this event
'first_bogo_received','first_discount_received','first_informational_received', 'first_bogo_viewed','first_discount_viewed','first_informational_viewed', 'first_bogo_completed','first_discount_completed', 'last_bogo_received' ,'last_discount_received','last_informational_received','last_bogo_viewed', 'last_discount_viewed','last_informational_viewed','last_bogo_completed', 'last_discount_completed','count_bogo_received','count_discount_received', 'count_informational_received','count_bogo_viewed','count_discount_viewed', 'count_informational_viewed','count_bogo_completed', 'count_discount_completed'
2- calculate features related to duration and reward and difficulty for every offer:

'max_duration_bogo','max_duration_discount','max_duration_informational',' min_duration_bogo','min_duration_discount','min_duration_informational', 'mean_duration_bogo','mean_duration_discount', 'mean_duration_informational','median_duration_bogo', 'median_duration_discount','median_duration_informational', 'sum_duration_bogo','sum_duration_discount','sum_duration_informational', 'max_reward_bogo','max_reward_discount','min_reward_bogo', 'min_reward_discount','mean_reward_bogo','mean_reward_discount', 'median_reward_bogo','median_reward_discount','sum_reward_bogo','sum_reward_discount','max_difficulty_bogo','max_difficulty_discount', 'min_difficulty_bogo','min_difficulty_discount','mean_difficulty_bogo', 'mean_difficulty_discount','median_difficulty_bogo','median_difficulty_discount ,'sum_difficulty_bogo','sum_difficulty_discount'

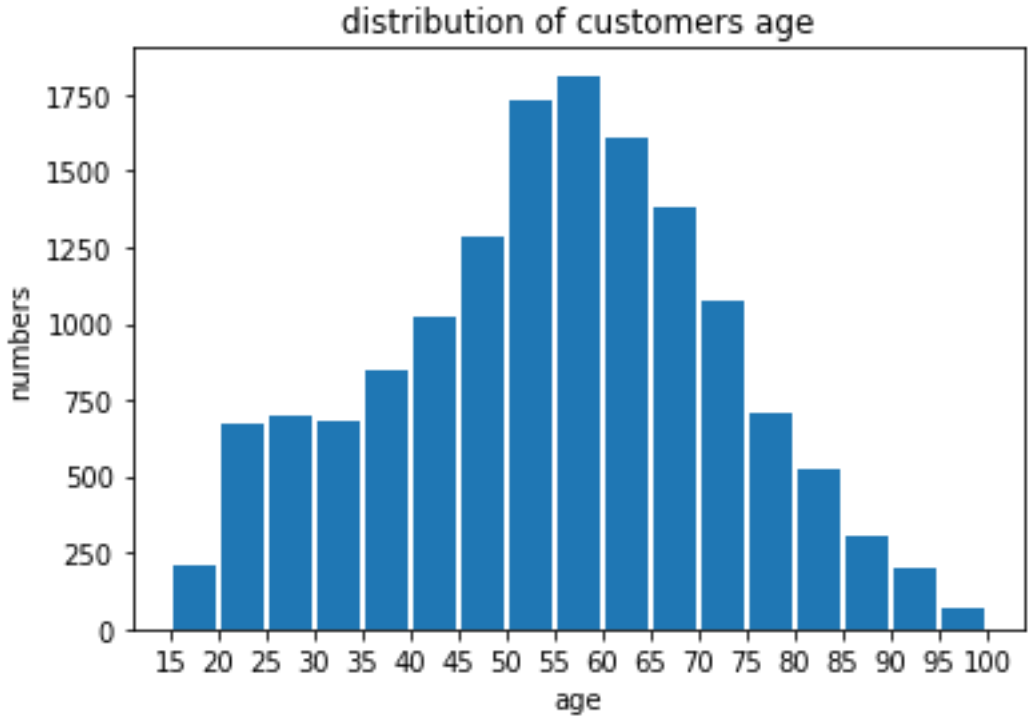
3- calculate all features related to money max, min, mean, median, sum, count, last amount customer has paid and first and last time:
'time_first_transaction','time_last_transaction','count_transaction', 'min_transaction', max_transaction', 'mean_transaction', 'median_transaction', 'sum_transaction','last_transaction'

4- I calculated if the offer will be completed or not:
- for bogo and discount type
the offer to be considered as completed, the customer must do these actions:
offer received -> offer view -> transaction -> offer completed
so for those who didn't view but complete the offer that means he/she made a transaction not to get the offer, they would buy even if they did not receive this offer so I didn't consider this offer as completed
- for informational type as there wasn't a record show if the offer is completed or not, I assumed if the customer made any transaction through the duration of the offer it will be considered as a completed offer.
I quote from project overview written by Starbucks "if an informational offer has 7 days of validity, you can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement."

5- I joined profile data frame (demographic information about every customer) with my modified data frame

- Exploratory Visualization:

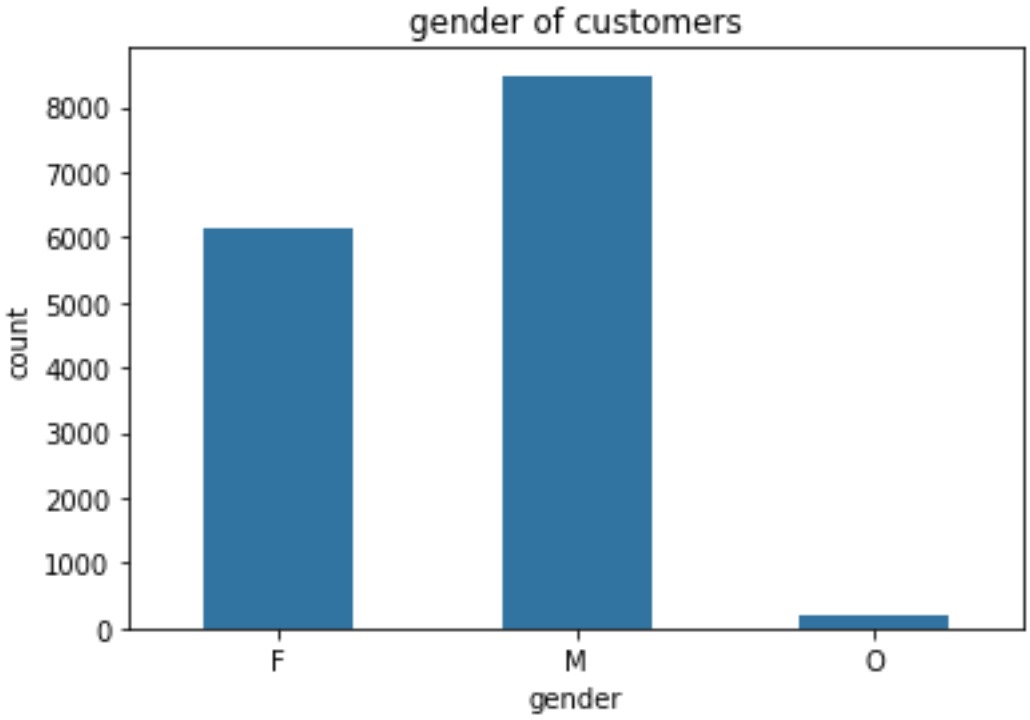
1- age of customers:



- as we can see most of the ages between 45-75 and the distribution is a close to normal one

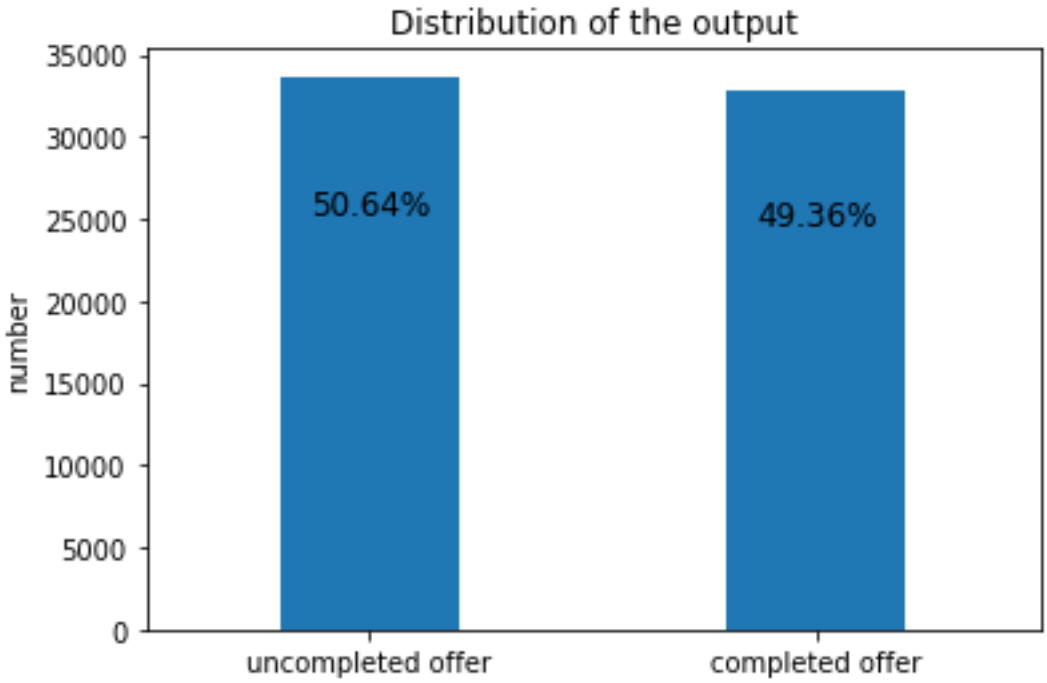
2- distribution of customers gender:

M	57.23
F	41.34
O	1.43



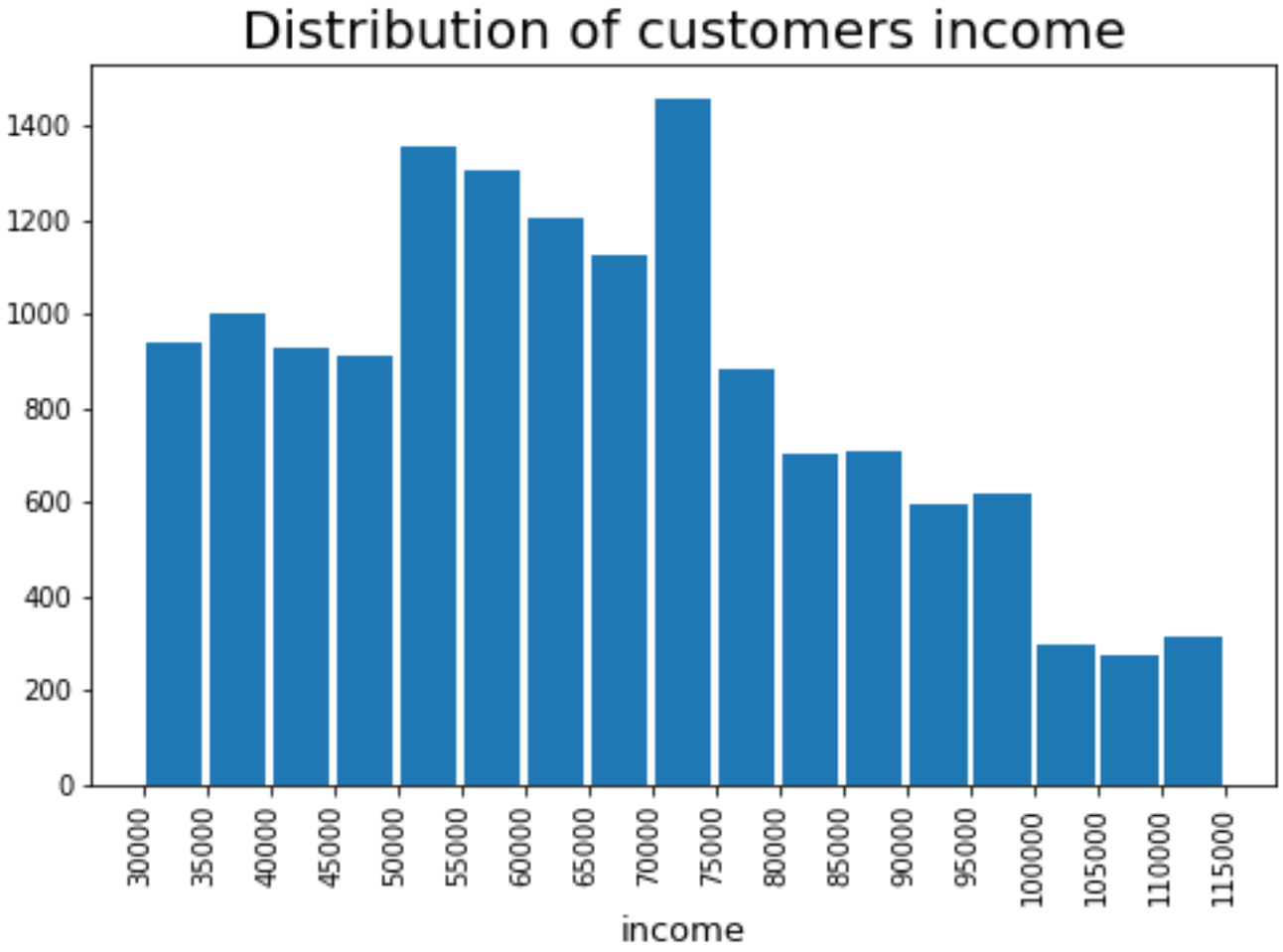
- As we can see 57% of customers is male and 41% is female

- Distribution of completed offer and uncompleted offer:



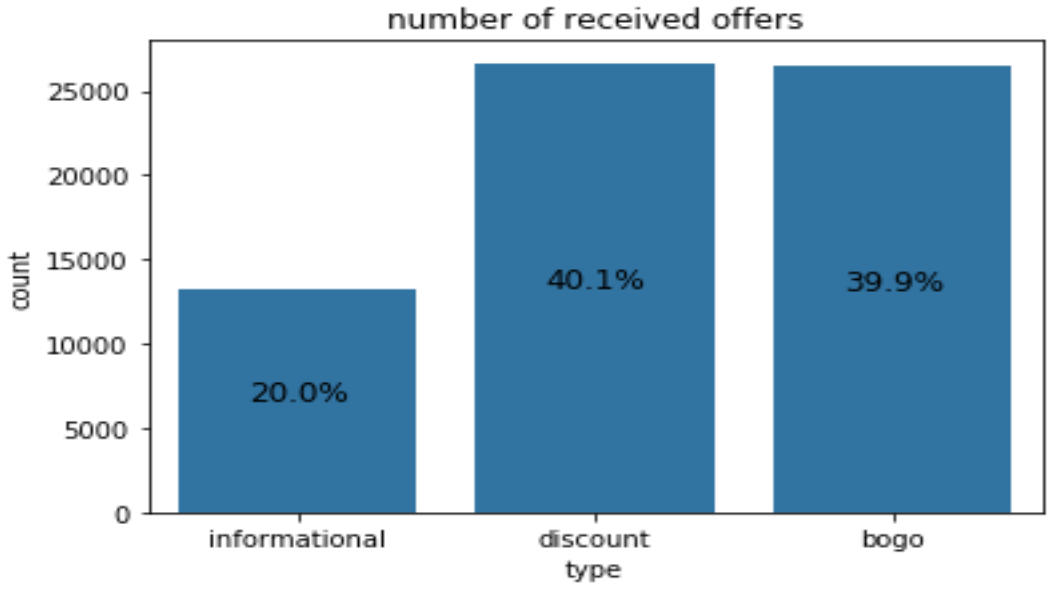
- 49% of received offer is completed so our input data is balanced with respect to output

- Distribution of income:



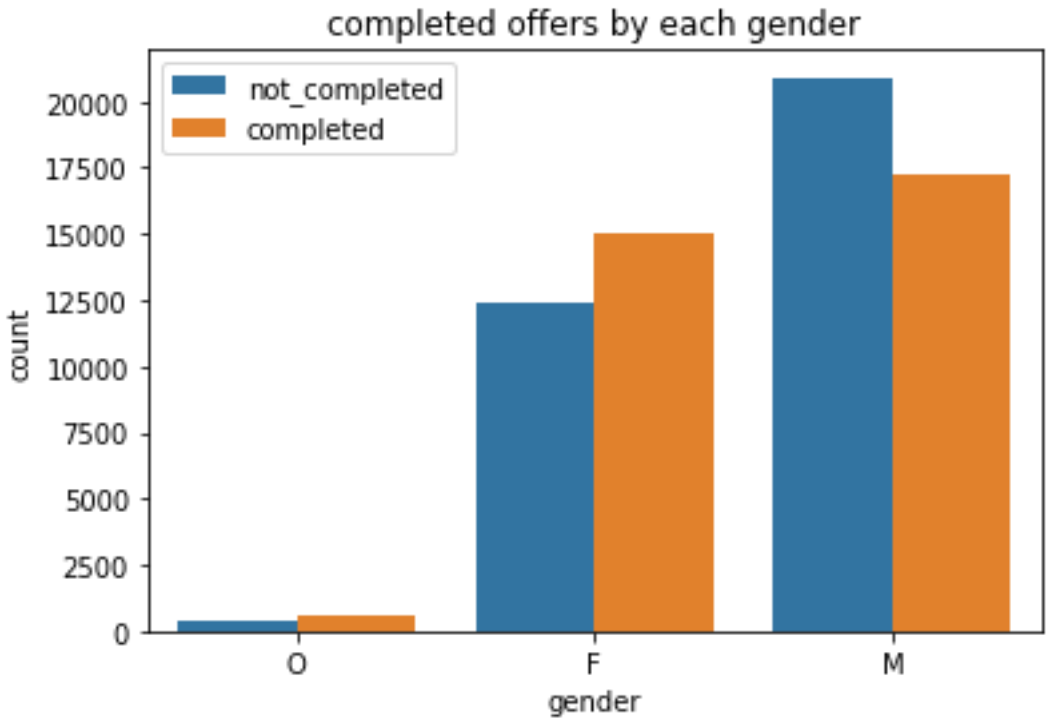
- the distribution is a quite right skew as few customers have a large income

- distribution of received offer types:

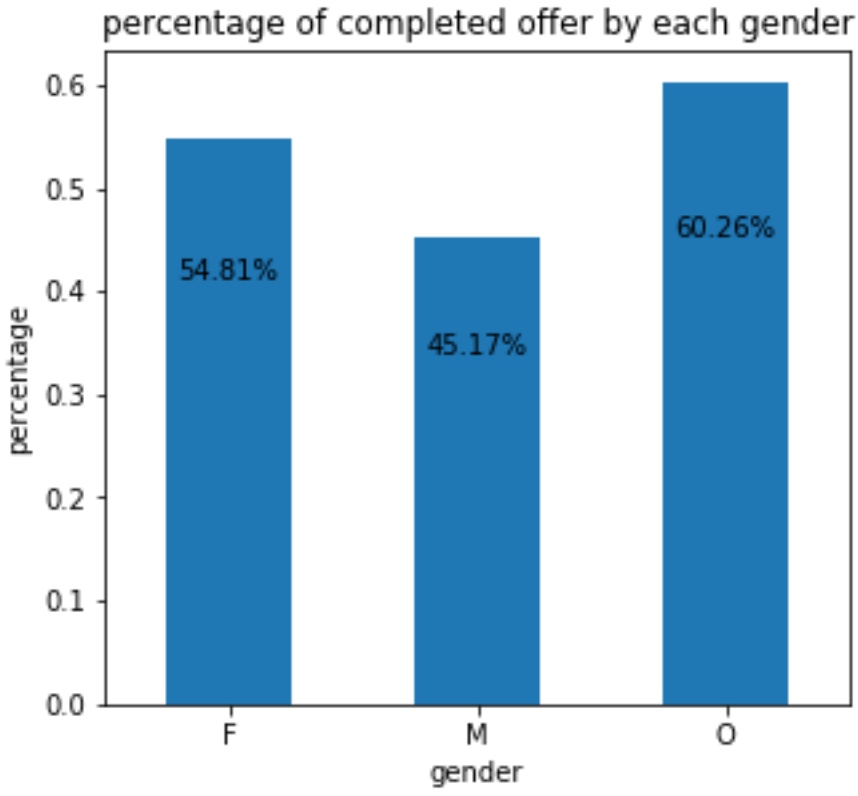


- bogo and discount offers approximately have the same number despite the percentage of informational type is about half of them

- Is the rate of completed offer is different with respect to gender?

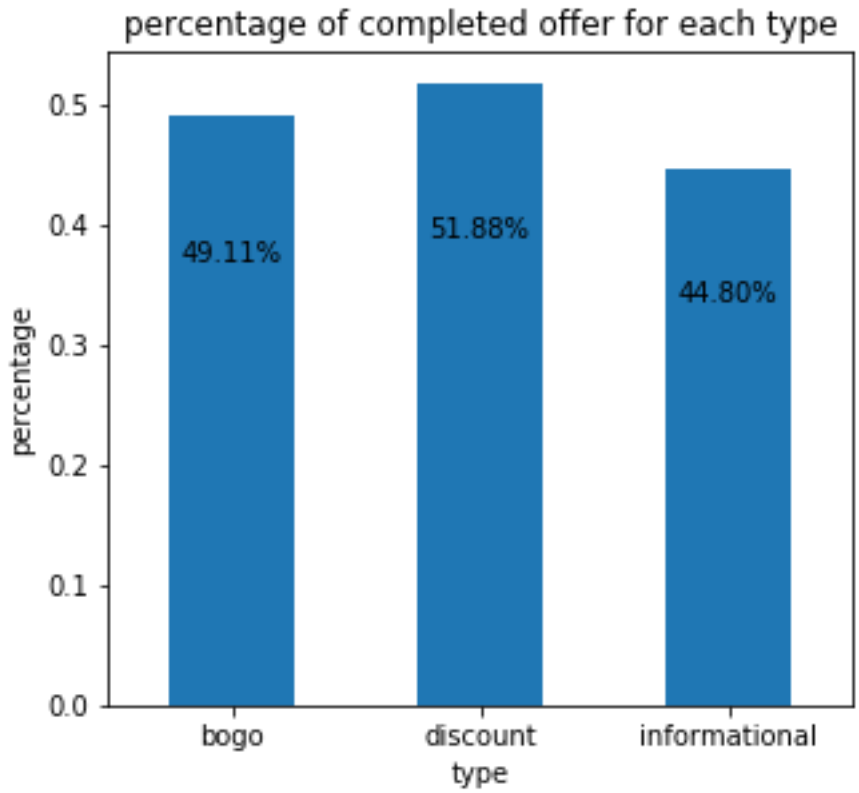


- the number of completed offers for females is larger than uncompleted ones unlike male

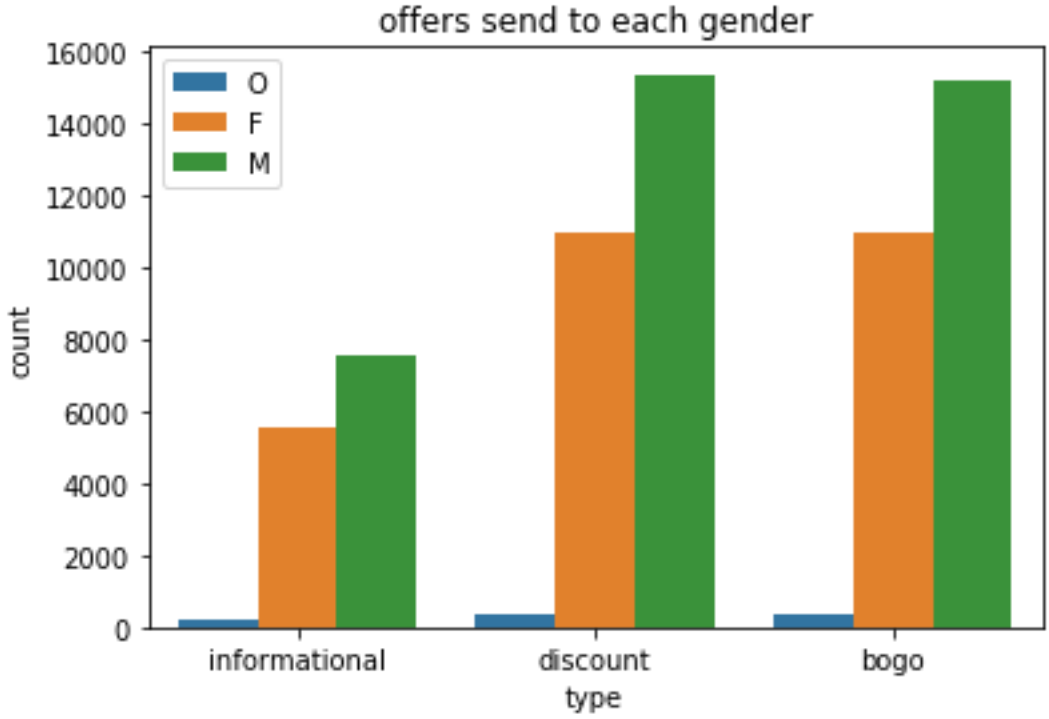


- 54% of females complete their offers and for men is only 45%

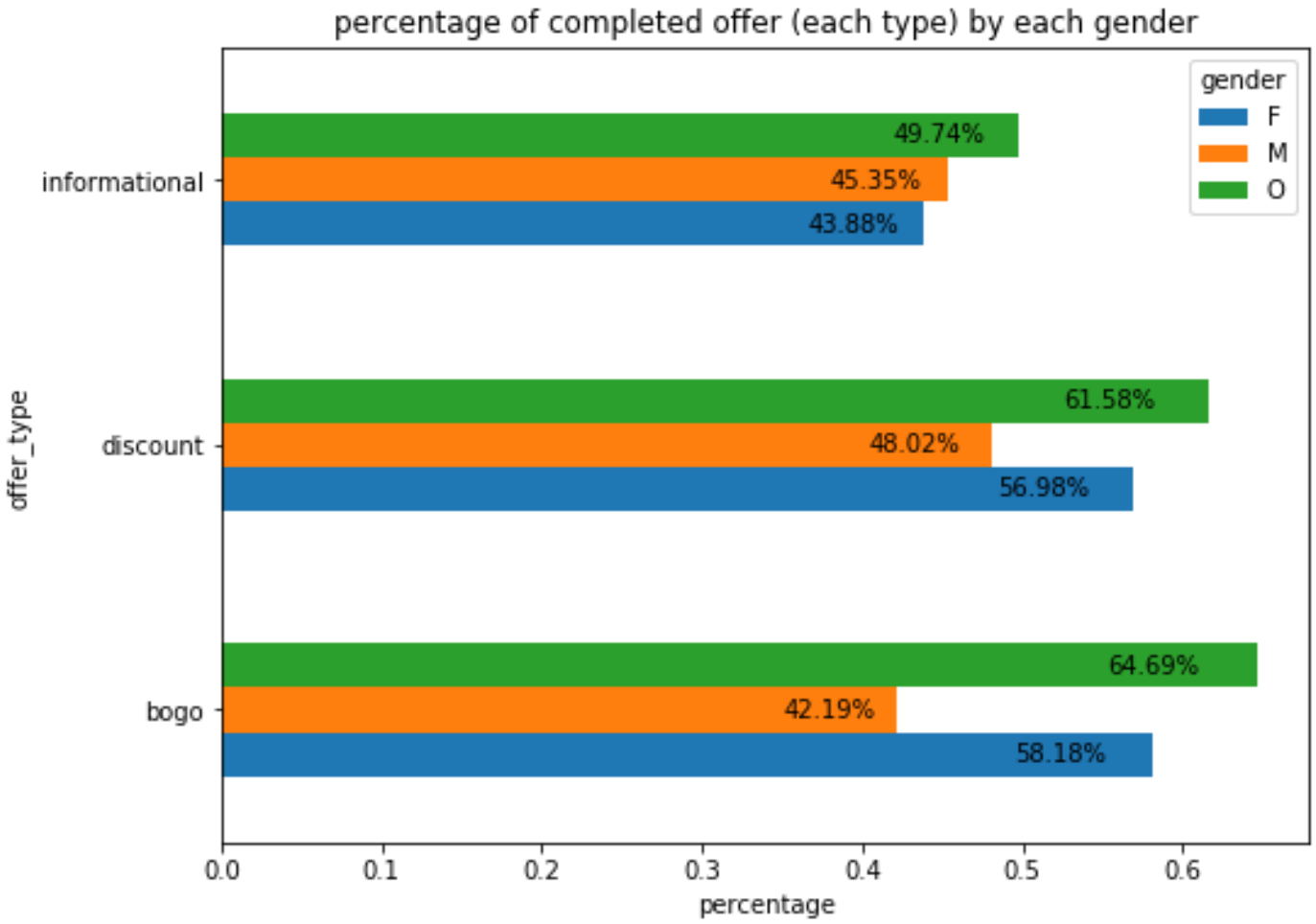
- We saw that proportion of offers types is not balanced but this did not has a large effect of the numbers of completed offers for each type:



- the received offers for each gender with respect to count of them is quite the same:



- I kept investigating more about the relation between gender and offer type and if offer is completed or not to make sure the data is not unbalanced or biased to specific gender of there is a



- 58% of bogo offers received my female is completed unlike males who complete only 42% of bogo offers and the same for discount offers, but informational offers have different characteristics as 45% of males complete the offers and 43% for females, which reveals that females is less affected by informational offers unlike other types

- Algorithms and Techniques:

1- logistic regression (bench model) was described in bench model section.

2- random forest : as our data has 125 features, random forest provide a way to utilize these features by taking fraction of them at each tree and pick the most useful one so I specify the max depth of each tree to prevent over fitting and exclude useless features from being used

RandomForestClassifier (max_depth=9, n_estimators=100 ,max_features=.75)

3- gradientboosting: will be the same as random forest in addition to, I initialized this model with the output of random forest as gradient boosting depend on decreasing the residuals from the initial predicted output and the actual output

GradientBoostingClassifier (max_depth=6,n_estimators=50,
init =rforest_result['model'], max_features=.75,learning_rate=0.1)

4- SVM: as our data has many features SVM provide a way to transfer the data to higher dimension and finding the interaction between them by kernel method and to prevent overfitting, I used a regularized method

5- Xgboost: it was well known for its speed ,high performance and ability to increase accuracy by the complicated way it constructs the samples trees

XGBClassifier('binary:logistic',colsample_bylevel= 0.7,colsample_bytree= 0.75,gamma= 0.01, learning_rate= 0.1,max_depth= 8,min_child_weight= 0.1,n_estimators= 50,reg_lambda= 10.0,subsample= 1.0)

- features engineering: (result represent the result of benchmark model after this step)

1- I calculated the completion rate for each offer type with respect to received ones.
Result :
train_acc 67.425
test_acc 67.13

2- I calculated the difference in time between the last event of every offer type and the time when offer is received (record I want to predict its output)

Result:

train_acc 67.447

test_acc 67.112

3- how often customer make a transaction and the period from last transaction

4- I also added the difference between 'duration' ,'reward' , 'difficulty' of the rec eived offer and min, max, mean of previous offers ('duration' ,'reward' , 'difficult y') for each offer's type

Result:

train_acc 67.308

test_acc 67.01

5- rate of making offers' event (recieve,view,complete) as average of hours when user receives or completes an offer as:

(Time of offer – time of first received offer)/count of received offer

- That was done for each event(recieve,view,complete) per each offer type

Result:

train_acc 67.342

test_acc 67.016

6- I added the number of uncompleted offers for each type

Result:

train_acc 67.31

test_acc 66.961

7- I also added the rate of uncompleted offer with respect to completed ones

Result:

train_acc 67.32

test_acc 67.003

8- I added ratio between mean transaction and income and ratio between sum transaction and income

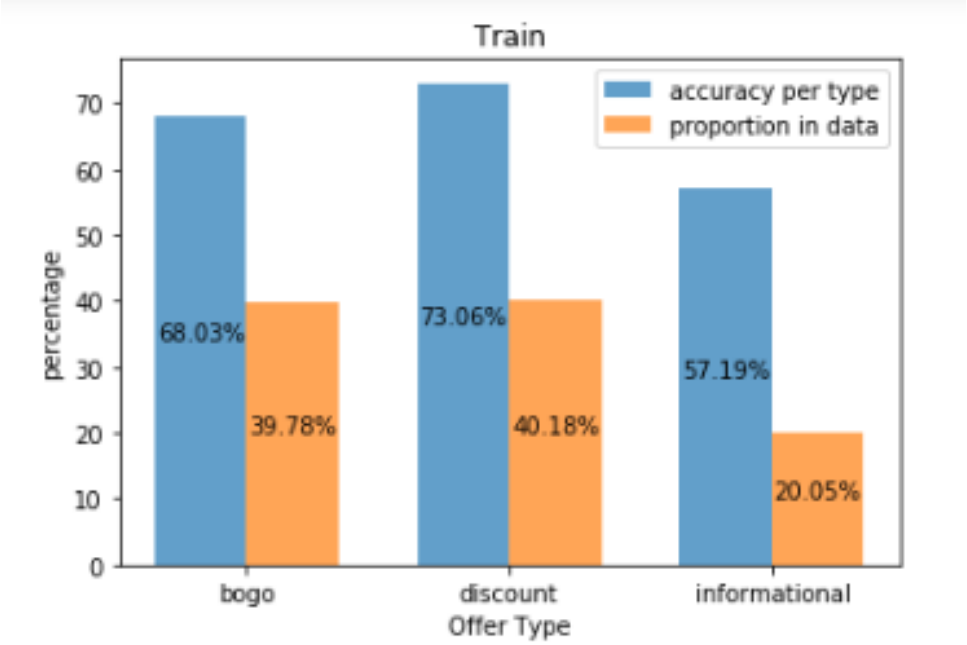
The final result of benchmark model after all these steps:

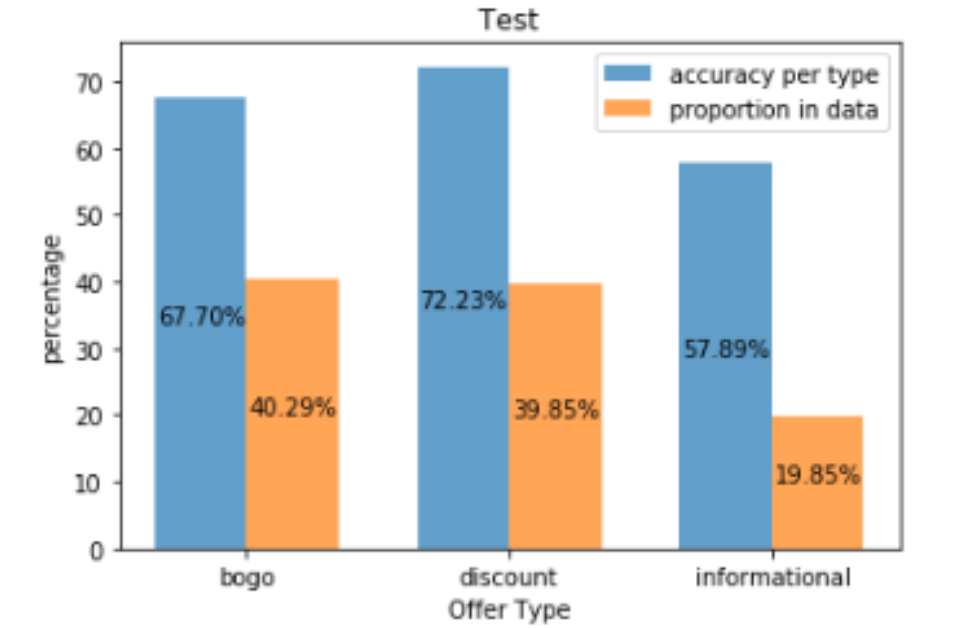
```
train_acc 67.876
precision 0.679
recall 0.679
f1 0.679

test_acc 67.557
precision 0.676
recall 0.676
f1 0.676
```

- Implementation:

- I have discussed the models I used as well as the parameters of these models
- I tuned these models with respect to accuracy result on test data and I did it though using SkLearn package whether using implemented models provide by them and accuracy metric but I implemented a function to help me in understanding the error with respect to offer types so I can understand the accuracy per offer type





- this function was used during trying feature selection and model tuning beside accuracy metric (all steps is well documented in jupyter notebook

- I have tried drop features with zero coefficients in baseline model but that had not any effect on increasing the accuracy

Result of random forest with all features:
train_acc 74.935
test_acc 72.687

result after drop features with zero coefficients in baseline (bench model):
train_acc 74.953
test_acc 72.723

-Refinement:
the result for every model before and after tuning:
1- random forest:

Before:
train_acc 0.7278395989974937
test_acc 0.7085889570552147

After:
train_acc 76.678
precision 0.767
recall 0.767
f1 0.767

test_acc 73.09
precision 0.731
recall 0.731
f1 0.731

2- gradientboosting:
I start with the same parameters of random forest but I there was an overfitting so I had to reduce the max depth even if it will decrease the accuracy by 0.1

Before:
train_acc 79.992
test_acc 73.523

After:
train_acc 76.239
precision 0.762
recall 0.762
f1 0.762

test_acc 73.409
precision 0.734
recall 0.734
f1 0.734

3- SVM:

I tried different types of kernel and tuning the regularization parameters to prevent overfitting

Before:

train_acc 69.562

test_acc 68.447

After:

train_acc 72.509

precision 0.725

recall 0.725

f1 0.725

test_acc 70.684

precision 0.707

recall 0.707

f1 0.707

4- XGboost:

I used random search provides by sklearn to the pick the best parameters

```
xgb_model = xgb.XGBClassifier(objective='binary:logistic')

param_grid = {
        'max_depth': [3,5,7,8],
        'learning_rate': [ 0.1, 0.2, 0,3],
        'subsample': [ 0.8, 0.9, 1.0],
        'colsample_bytree': [ 0.7, 0.75, 0.8],
        'colsample_bylevel': [ 0.7, 0.75, 0.8],
        'min_child_weight': [.001,.01,.1],
        'gamma': [0,.001,.01,.1],
        'reg_lambda': [0.1, 1.0, 10.0],
        'n_estimators': [50,100]}

rs_clf = RandomizedSearchCV(xgb_model, param_grid, n_iter=20,
                            cv=None,n_jobs=2,
                           scoring='accuracy', refit=True, random_state=42)
print("Randomized search..")
search_time_start = time.time()
rs_clf.fit(grad['X_train'], grad['y_train'])
print("Randomized search time:", time.time() - search_time_start)

best_score = rs_clf.best_score_
best_params = rs_clf.best_params_
```

train_acc 78.667

precision 0.787

recall 0.787

f1 0.787

test_acc 73.686

precision 0.737

recall 0.737

f1 0.737

-I tried to build a stack model to all these models but unfortunately the error is overlapped between them

	logistic_train	rforest_train	gradient_train	xgb_train	svm_train
0	0.351929	0.431840	0.518438	0.462488	0.0
1	0.303356	0.183015	0.192708	0.218995	0.0
2	0.718914	0.735711	0.786495	0.821642	1.0
3	0.760404	0.725653	0.767371	0.785522	1.0
4	0.548221	0.333112	0.386247	0.327354	0.0

The result was less than XGboost model:
train_acc 80.63
precision 0.806
recall 0.806
f1 0.806

test_acc 72.832
precision 0.728
recall 0.728
f1 0.728

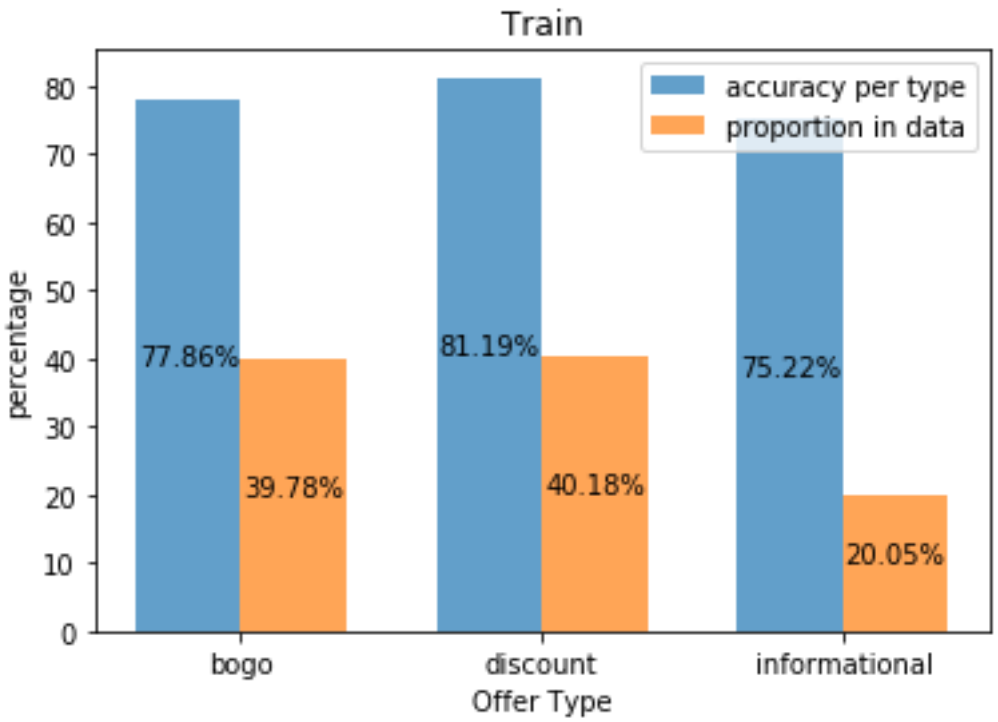
- our final model is XGBoost with these parameters:

XGBClassifier('binary:logistic',colsample_bylevel= 0.7,colsample_bytree= 0.75,gamma= 0.01, learning_rate= 0.1,max_depth= 8,min_child_weight= 0.1,n_estimators= 50,reg_lambda= 10.0,subsample= 1.0)

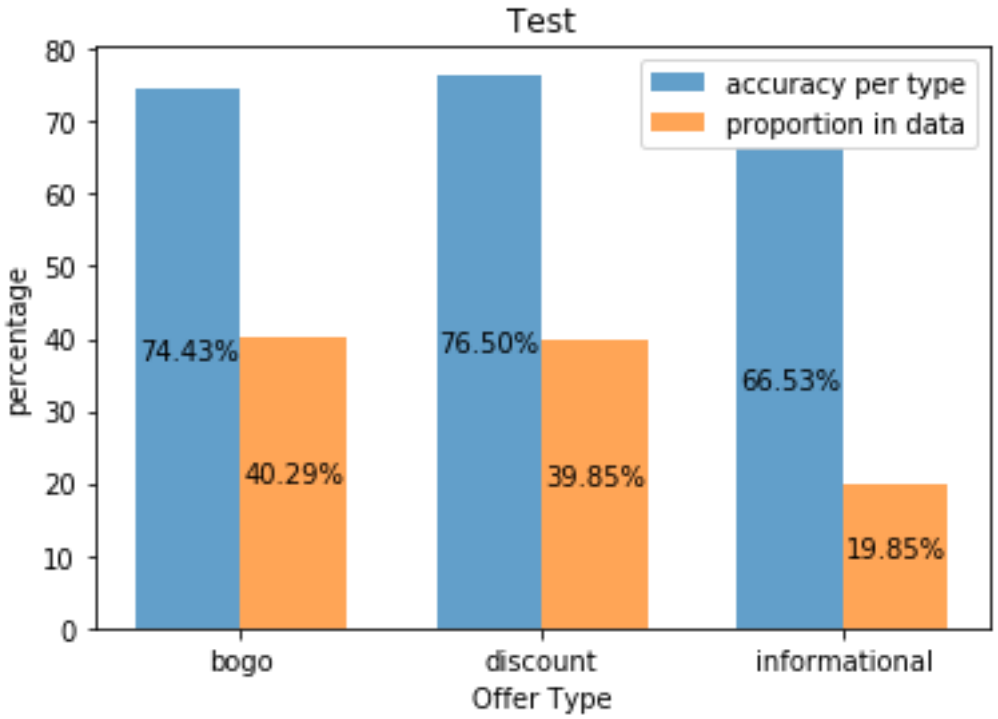
Final Result:

Training Data:

train_acc 78.667
precision 0.787
recall 0.787
f1 0.787



Testing Data:
test_acc 73.686
precision 0.737
recall 0.737
f1 0.737



Validate:
- there are multiple issues in the data:
1- size of data is not large enough
2- time frame of data is quite short
3- as size of data is 66501, I could not drop the first offer received for each customer
As these records is not helpful and do not reveal any historical information about customer s pattern on using the app

- so, our hand was tied, but we could achieve a well result

Benchmark model	XGBoost model
train_acc 67.431 test_acc 67.01	train_acc 78.667 precision 0.787 recall 0.787 f1 0.787 test_acc 73.686 precision 0.737 recall 0.737 f1 0.737

- to make sure of robustness of the model I bootstrapped the testing data for 10000 sample And calculated the accuracy for every sample:

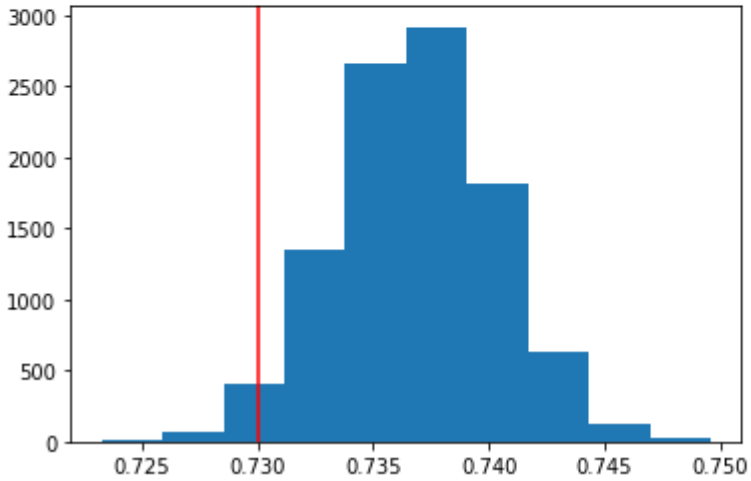
```
df=pd.DataFrame(xgb_result['X_test'])
df['y']=xgb_result['y_test']

accur=[]
for i in range(10000):
    bootsamp = df.sample(df.shape[0], replace = True)
    y_pred=xgb_result['model'].predict(bootsamp.iloc[:,:-1].values)
    accur.append(accuracy_score(bootsamp.iloc[:,-1].values, y_pred))
```

- null hypothesis **accuracy** <= **73**
- alternative hypothesis **accuracy** >**73**

```
null_vales = np.random.normal(np.mean(accur), np.std(accur), 10000)
```

```
plt.hist(null_vales);
plt.axvline(.73 , c= 'r')
plt.show();
```



- P_value = 0.0224 so we can reject the null hypothesis that accuracy is less than or equal to 73%