UDACITY

Machine Learning Engineering Nanodegree

Capstone Project- STARBUCKS

1. Problem Definition:

1.1. Domain Overview

When it comes to coffee, Starbucks is a global and a leading brand. With the advent of the usage of Machine Learning(ML) in todays' world, the company can utilize the strength of ML to increase its revenue. This project is concerned with associating the various offer types to its customers. The reason is that all users should not receive the same offer type. If we can understand the historical customer behaviour/reaction to these offers, we can create personalized offer predictions which can target these customers to increase the revenue. The data provided is collected from the mobile app.

1.2. Problem Statement

Starbucks collects the customer data to understand their behaviour on the rewards and offers sent via the mobile-app. Once every few days, Starbucks sends the personalised offers to its customers. These customers can respond positively/negatively/neutrally. A key thing to note is that not all the customers receive the same offer. The task of this project is to combine transaction, demographic and offer data of the past (which is already provided) to determine which demographic groups respond best to which offer types.

1.3 Data

The dataset simulates the customer behaviour towards the promotional offers. There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels.

Three json files are given. Structure of the data is as follows:

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

- 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: demographic data for each customer.

- 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: records for transactions, offers received, offers viewed, and offers completed.

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

In this project, we created an attribute called offersuccessful which is binary in nature. We will build a classifier to predict whether an offer sent to a customer is successful (has a conversion rate) or not.

Note: We will look into the statistics of the data in detail in the next section.

1.3 Potential Solution

Note: We have taken into account only the offers which have been received and viewed by the customers for simplicity.
If the offer has been received, viewed and completed by any customer, we mark it as success/conversion/offer-successful. This attribute is binary where 0 means that the offer was not successful and 1 means vice versa. Then we have considered the case of building a classifier which can predict which offers can be successful and which can not be. This is also a supervised learning task where the target class is nearly balanced. Since the target variable is binary in nature, we built a Logistic Regression model as a baseline. Then we tried linear state vector classifier and xgboost model. These two are the advanced models. The model which performs the best as per the below definitions of the chosen evaluation metrics was picked and deployed on AWS Sagemaker to generate predictions. In our case the best model is Xgboost.

1.4 Evaluation Metrics used

Note about some terms used:
`If Positive = identified and negative = rejected,`

```
TP- True Positives →  correctly identified positive samples
FP- False Positives →  incorrectly identified
FN- False Negatives → incorrectly rejected
TN- True Negatives →  correctly identified negative samples
```

Precision: It is the fraction of relevant instances among the retrieved instances.

```
Precision = TP/(TP + FP)
```

Recall: It is the fraction of the total amount of relevant instances that were actually retrieved.

```
Recall = TP/(TP + FN)
```

F1: This metric is the harmonic mean of precision and recall. It is a point metric quite popular in evaluating the classifier.

```
F1 = 2 * ( Precision * Recall ) / ( Precision + Recall )
```

ROC-AUC Score: gives a trade-off between the true positive rate and false positive rate. It is the area under the ROC curve (Receiver Operating Characteristic).

2. Exploratory Data Analysis

As mentioned above, we were provided with three datasets: profile, portfolio and transcript. First we performed some structural analysis on the three datasets as mentioned below:

Portfolio- There are 10 records with 6 features given for 10 unique offer ids. These offers can be sent via 4 different channels such as social, web, mobile and email. There are 3 unique offer types such as bogo, discount and informational.

Profile- There are 17000 records with 5 features present. One weird observation was that there were a total 2175 records present where age of the customers was mentioned as 118 with null income values. We filtered out such records.

Transcript- This was the trickiest dataset with 306534 records with 4 features which comprised the information of various offers received, viewed and completed by any customer.

Next, we tried to answer the following questions:

Note: For questions 1-4, please refer to Plot1.
For questions 5-6, please refer to Plot2.

Question 1: What is the distribution of gender in the data?

- 57.2% of the customers are males, 41.34% of the customers are females and a mere share of 1.4% of customers belong to the other gender type. For simplicity, we account for only the male and the female gender groups.

Question 2: What is the mean age of the customer?

- On average a customer is 54 years old.

Question 3: How the distribution of income looks like and what is the average income of the customer?

- Income distribution does not look like a normal distribution (simply by eyeballing the graph). On average a customer earns 65000$ per year. No currency is given and we are assuming it is dollars.

Question 4: In which year most of the customers started their membership with Starbucks?

- Most of the customers started their membership with starbucks from the year 2018 onwards.

Question 5: How much share each offer type hold in this data?

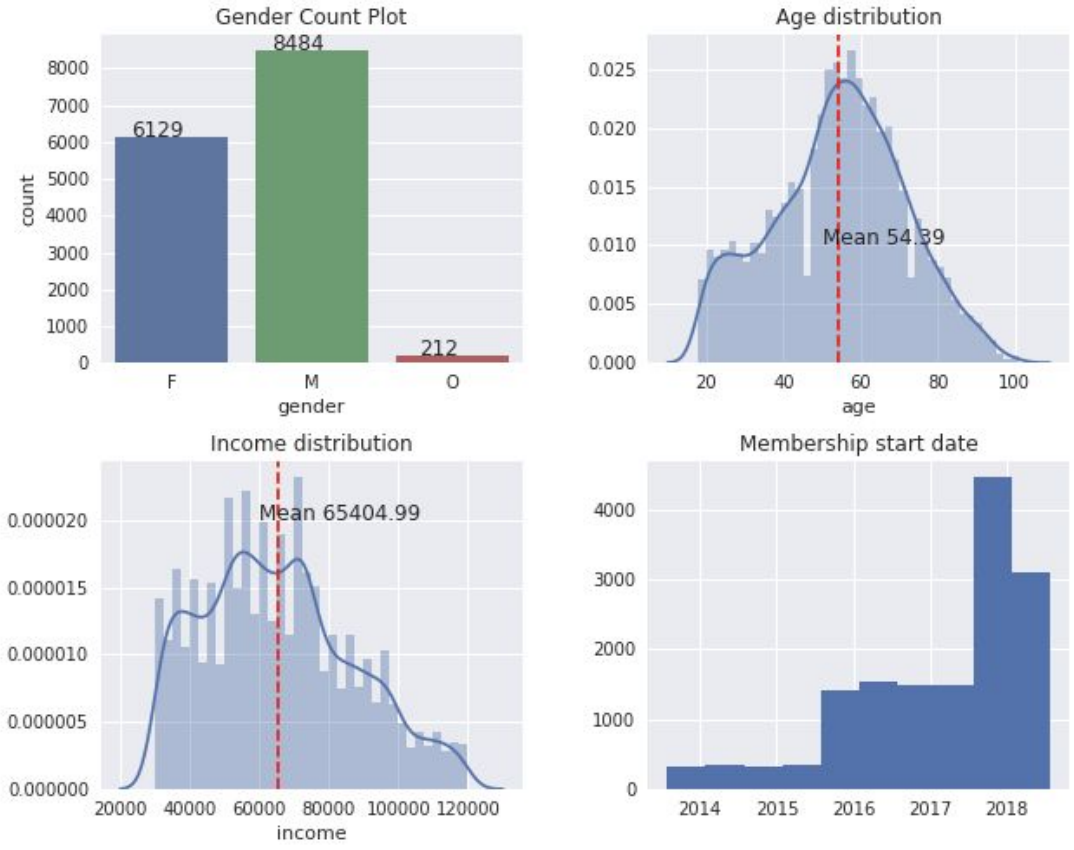
- Discount and bogo roughly share an equal share of approx 40%. Informational offers has a share of 20% in the data. This implies that the customers are generally sent bogo and discount offers.

Question 6: How much share each channel holds in this data?

- All the offers are sent via email. Then, 90% of the offers were sent via email.

Question 7: What is the maximum and minimum duration of any offer?

- Maximum duration is 10 days and minimum duration is 3 days.

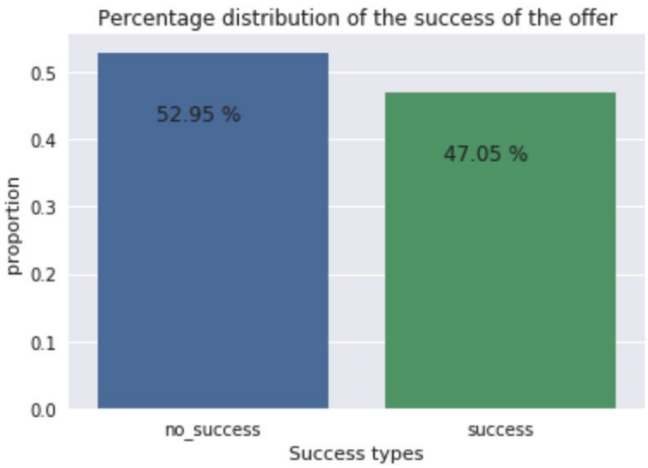


Plot 1: Subplots for profile data

Plot 2: Subplots for portfolio data

Question 7: What is the share of successful offers in the data?

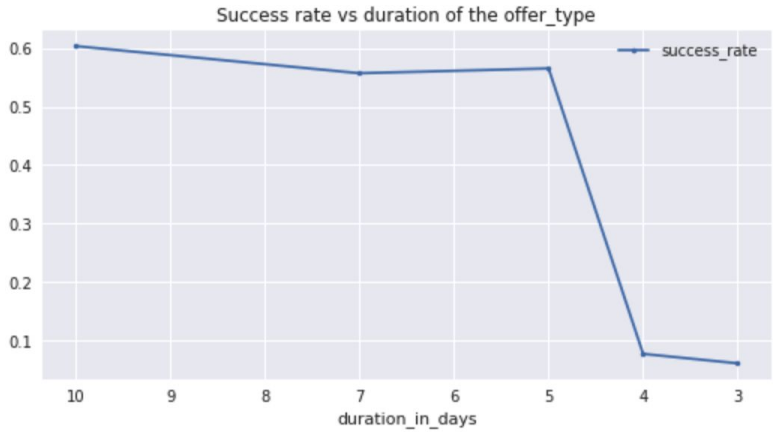
- 47% of the offers were successful. Refer to Plot 3.



Plot 3: Success rate distribution in the data

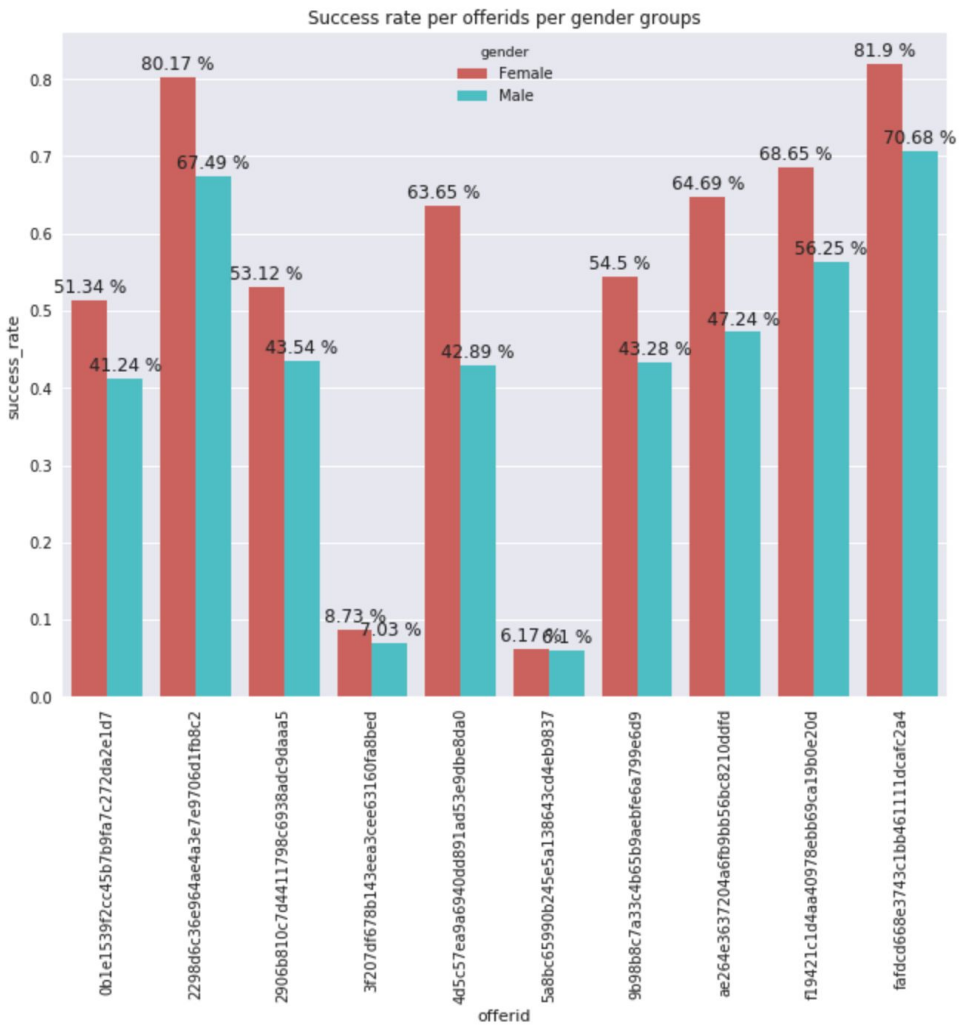
Question 8: Is there any relation of duration of days to the successful offer rate?

- Looking at Plot 4, it seems that more the duration of the offers, the more chance it is for the respective offer to be converted.



Plot 4: Trend in the success rate with that of the duration of the offer.

Plot 5 shows that females have more likelihood of using the various offers as compared to their male counterparts.



Plot 5: Success rate per offer type per gender. Females are more inclined towards using the offer.

Success rate (in %) of offers per channels, gender and offer type

Channel		**Gender**		**Offer type**	
Social	53.63	Female	53.11	discount	60.24
Web	51.21	Male	42.68	bogo	53.87
mobile	47.22			informational	6.94
email	47.05				

- Offers sent via social media have a high conversion rate.
- Females are more likely to convert an offer at Starbucks.
- Customers use offers offering discounts more. Offers which provide mere information have the least likelihood of getting converted.

3. Implementation

3.1 Data Cleaning

In all the three tables, some column names were renamed to be more meaningful such as offer id, customer id. All the categorical variables were one-hot-encoded and the continuous variables were normalized. The transaction history of all the customers which are not present in the profile table were deleted. All the three tables were combined on the fact that the customers first should receive the offer, then he should view it and then if the offer is still valid, either it can be left or can be proceeded for the completion.

In more details:

- Portfolio
    - one-hot encode channels and offer_type.
    - remove underscore in column names.
    - rename id to offer id (more meaningful).
- Profile
    - remove null values.
    - filter the other gender for simplicity.
    - preprocess date column.
    - binarize the gender.
- Transcript
    - rename person to customerid
    - remove all the customer ids which are not in the profile table
    - convert hours into days and rename the column
    - parse the offers as one-hot encoded columns
    - create a separate dataframe for transaction → These transactions are all about the offers which a customer has first received, then viewed and if applicable, used and completed. For this to happen first we check if still the respective offers stands valid. There were some offers which were never viewed by the customer. Such transactions were filtered out.

After the three tables were combined, 'time','customerid','email','informational' attributes were dropped.

Time → we are not concerned when the offer's start date is.

Customerid → no need for this since it does not have any meaning.

email → all the offers were sent via email, hence there is no extra information conveyed by this feature.

informational → all the offers which only convey some information about the offers were also not required because only 20% of offers were informational and out of which only 7% were marked as successfully converted.

For continuous variables such as 'income','totalamount','duration','reward', we used min max scaler function. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. For each value x, the formula is:

$$\frac{x_i - min(x)}{max(x) - min(x)}$$

3.2 Project Structure

This project has been divided into three jupyter notebooks:

1. data_cleaning.ipynb
2. data_analysis.ipynb
3. model_building_and_evaluation.ipynb

Notebook names are self-explanatory.

3.3 Model

Benchmark Model: To setup the baseline, Logistic Regression was used. This is because the target variable is binary(successful offer/ non-successful offer). This model can give the probabilities of the samples belonging to either of the two classes using a logistic regression function which is:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Further models:

Linear-State Vector classifier. This is quite an efficient algorithm when there appears to be a linear separation between the two classes. I also used rbf kernel (results are not reported for simplicity) but it pushed down the performance significantly. The biggest advantage of state vector classifiers are that they use only the subset of training points for decision function and hence are quite memory efficient.

XGBoost. XGBoost is an efficient and easy to use algorithm which delivers high performance and accuracy as compared to other algorithms. It is also known as a regularized version of GBM(Gradient Boosted Machines). It uses multiple CPU cores to execute the model(parallel processing).

XGBoost model was picked(because of its high performance on all the metrics). I then used AWS-Sagemaker's XGBoost model to build, tune and deploy it on the cloud platform.

Following steps were followed for AWS Sagemaker:

Data upload to S3 bucket: The XGBoost classifier requires the dataset to be written to a file and stored using Amazon S3. So we first created training, validation and test set in a csv format with no headers and then uploaded to the s3 bucket. The data format also should have the target variable in front followed by the rest of the feature variables.

Creating XGBoost model:

- Model artifacts: the artifacts are the actual trees that are created during training i.e. the actual model itself.
- Training code (container): uses the training data that is provided and creates the model artifacts.
- Inference code (container): uses the model artifacts to make predictions on new data.

Hyper parameter tuning: Below parameters were set for the tuning purpose.

```
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        early_stopping_rounds=10,
                        num_round=500)
```

Deploy the model: We deploy the model on sagemaker by creating the endpoint. Note that once the endpoint is up and running, it is billable. So never forget to delete it once your work is done.

Make predictions: In this part, we create predictions from the trained and deployed model to evaluate it.

Clear the resources: We delete the endpoint, s3 bucket and stored model.

Please note that the tuning of logistic regression and linear-svm can also be done using cross validation and explicitly passing the hyper-parameters. But this is beyond the scope of this project. I choose to pick the best model and then further tune it to achieve best results.

4. Results

	Measure	Log_Reg	SVC	XGBoost
0	F1	0.841543	0.851419	0.911694
1	Precision	0.871918	0.908318	0.896412
2	Recall	0.813213	0.801228	0.927507
3	ROC-AUC	0.852745	0.864150	0.915427

Note: The results are for the test set.

Clearly from the above results table, both Linear SVC and XGBost outperforms the baseline model. XGBoost has a comparable precision of ~90% to that of SVC's model. But for the rest of the metrics such as recall, ROC-AUC score and F1 score, it clearly outperforms the others.

Please note that these results are for the test set and from the sklearn library.

Refinement

We next tuned the baseline model only for the parameter C. C is the trade-off parameter of logistic regression that determines the strength of the regularization , and higher values of C correspond to less regularization (where we can specify the regularization function). We passed the following range:

```
C_param_range = [0.001,0.01,0.1,1,10,100]
```

The results were as follows:

	C_parameter	F1	Precision	Recall	ROC AUC
0	0.001	0.697571	0.67983	0.716262	0.706036
1	0.010	0.727059	0.701077	0.755042	0.732367
2	0.100	0.769341	0.755328	0.783884	0.777453
3	1.000	0.841543	0.871918	0.813213	0.852745
4	10.000	0.860005	0.899383	0.823931	0.870405
5	100.000	0.86215	0.900477	0.826951	0.872267

As can be seen above, there is not much difference in the results between C=10 and C=100. This also means that not much regularization is needed for this dataset. Chances of this dataset getting over-fitted is low. Next we pick the best scores(the 6th row) and update the results table as below:

	Log_Reg	tuned Log Reg	SVC	XGBoost
0	0.841543	0.862150	0.851419	0.911694
1	0.871918	0.900477	0.908318	0.896412
2	0.813213	0.826951	0.801228	0.927507
3	0.852745	0.872267	0.864150	0.915427

A significant improvement can be seen for the tuned version of the baseline for precision score. However, if we take into account all the four metrics together, XGBoost model performs best.

More results

We picked the XGBoost model to build and deploy on sagemaker. We then obtained the following results on the test set (the results should be the same, we used sagemaker for practice):

	Measure	XGBoost
0	F1	0.911694
1	Precision	0.896412
2	Recall	0.927507
3	ROC-AUC	0.915427

Next, we created a confusion matrix to look into the miss-classification error (false positives and false negatives).

```
[[10268  1113]
 [  678  9585]]
```

```
True positives (TP): 10268
False positives(FP): 1113
False negatives(FN): 678
True negatives(TN): 9585
```

```
Miss-classification rate = (FP + FN) / (TP + FP + FN + TN) = 8.27%
```

XGBoost has quite a low percentage of miss-classification error which is good for the classifier. This model also has an F1 score of 91% which is the harmonic mean of precision and recall. Also, with 91% of cases, this classifier will segregate the positive class and negative class when a sample from both the cases are taken simultaneously(this is the very definition of ROC-AUC score).

5. Conclusion

The problem here was that we wanted to personalise sending the different kinds of offers to the customers because not all the offers should be sent to all the customers randomly. Hence, we turned this problem into a supervised learning task where we trained a classifier given the features related to the customer's profile, offer's portfolio and customer's transaction data, predict whether an offer will be converted/successful or not. To do this, we first implemented a baseline of logistic regression because that is the first and foremost algorithm which comes to the mind if the target data is binary (categorical) in nature. Logistic regression gives the probability of a sample belonging to both the classes. We found that the model performance was good with an average performance of 85% taking all the four metrics

together. However, we then tried two advanced algorithms. First we implemented Linear-SVC and found the performance of the model to be around 87% taking all the metrics together. Then we implemented XGBoost to have a performance jump to 90% taking all the metrics together. This is also because XGBoost is quite an efficient algorithm and known to outclass all other classifiers. It is capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters. System-wise, the library's portability and flexibility allow the use of a wide variety of computing environments like parallelization for tree construction across several CPU cores; distributed computing for large models and Cache Optimization to improve hardware usage and efficiency. With all these advantages, we still face a miss-classification error of around 8%.

Scope of Improvement:

- Better and more efficient data cleaning can be done.
- We can include more edge cases such as taking into account the transaction data of those customers whose profiles are not present in the profile data. Such cases can actually test the learning of the model. This can be a more-realistic setting.
- We can have more data to train the model effectively.
- We can try basic deep learning based models such as feed forward network, LSTM classifier etc.

Reflection:

Personally, the profile and portfolio data were easy to handle. The most time-consuming data was transaction data. It was quite challenging to understand it. It took me several iterations to write code to clean it. Implementing the model on sagemaker was comparatively easy, however accessing sagemaker from external notebook took me quite some time. I had to establish aws cli configuration to set up my access key and pins in .aws file in my local system(the previous projects asked us to use the notebook directly from the sagemaker where we don't have to take care of the credentials). It was good learning.