Machine Learning Engineering Nanodegree

Capstone Project

Mahlong Makwele Wishbert
03 March 2022

Starbucks

Definition

Project Overview

Being able to tell how customers will react to products or offers is an important part of business, StarBucks has set out to figure out just how to do this. In this project I analyze simulated StarBucks data provided by Udacity to try and figure out how customers would respond to the different offers given by StarBucks.

Problem Statement

The aim is to determine how certain customers will respond to certain offers.
The data can be tricky because you have to keep in account that some customers can make purchases through the app without having received or seen the offer and thus these customers are not influenced by the offer. From a profit perspective it is also good to look at what customers will purchase still, without any offers available. You can assume that when a customer sees an offer the offer has an effect on the customer for the duration of its validation time and from this we can also see how customers would react due to the offer.

How I will deal with the problem is as follows:

- I will create a jupyter-notebook workspace

- I will import and try to merge to data files
- I will do an exploratory data analyses to try and find useful and meaningful insights
- I will train the benchmark model and other models
- I will evaluate the models

Metrics

All the models used in this problem classify data with regards to offer completion.
A simple way to evaluate the models is by calculating for precision, recall and F1 score.

$$\text{Precision}(class = a) = \frac{TP(class = a)}{TP(class = a) + FP(class = a)}$$

$$\text{Recall}(class = a) = \frac{TP(class = a)}{TP(class = a) + FN(class = a)}$$

$$\text{F-1 Score}(class = a) = \frac{2 \times \text{Precision}(class = a) \times \text{Recall}(class = a)}{\text{Precision}(class = a) + \text{Recall}(class = a)}$$

The above equations are for calculating the metrics by which we will evaluate. The python code does the calculations for us.

Analysis

Data Exploration

The Starbucks dataset comes in three files, files that need to be joined together to be able to gain insights from. The files are portfolio.json, profile.json and transcript.json. The portfolio.json file contains all the different offers that can be made and details about the offers, the data columns are as follows:

- 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)

	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

The profile.json file contains information about the customer demographic, the data columns are as follows:

- 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

	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

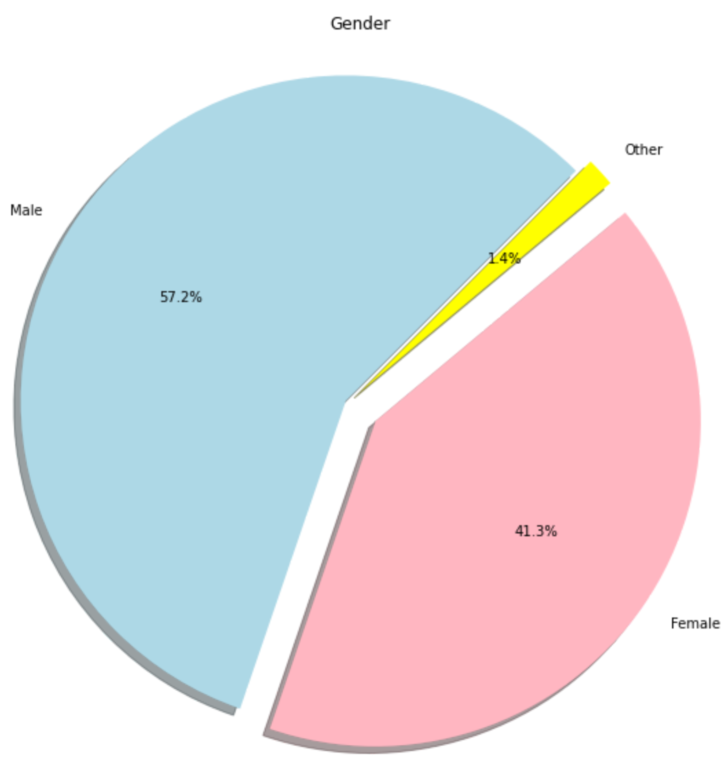
The transaction.json file contains information about events when offers were received and completed, the times of completion. The transaction data columns are as follows:

- event (str) - record description (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

	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'}

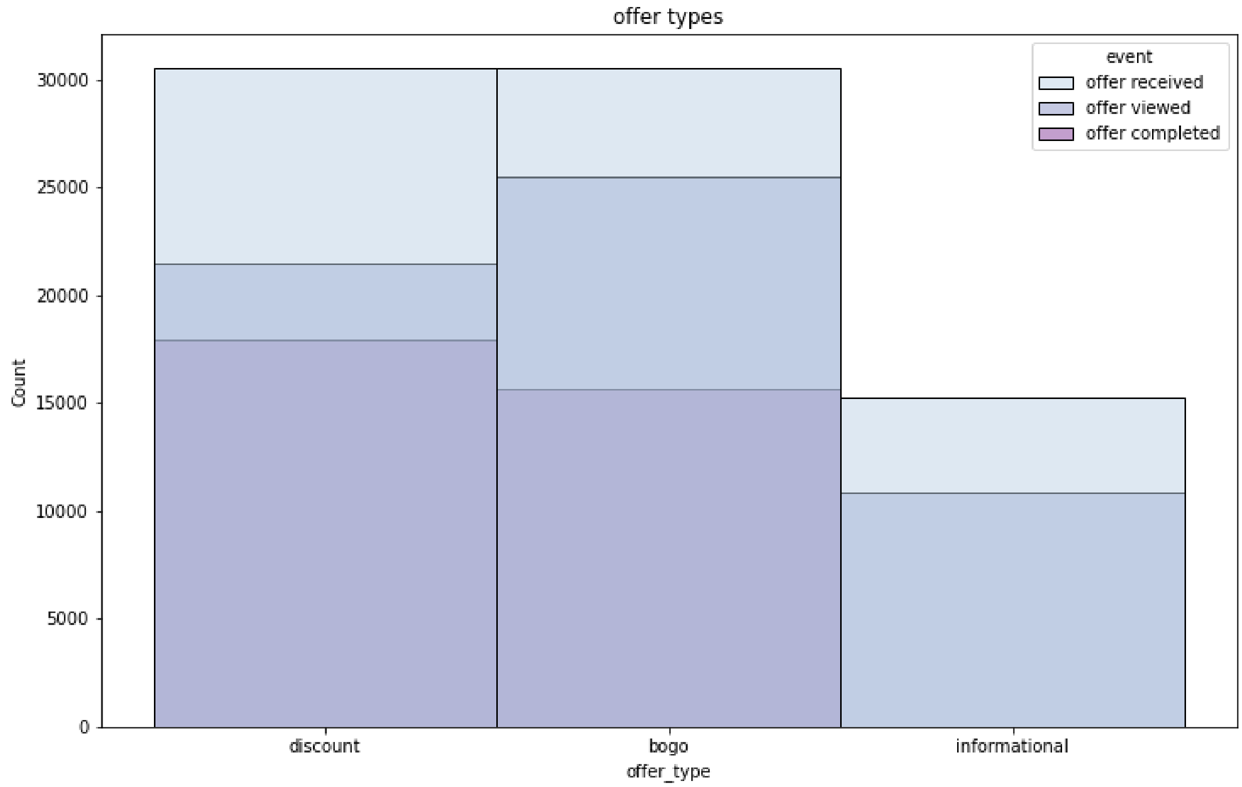
Exploratory Visualization

The chart below shows a gender distribution of the customers at StarBucks.



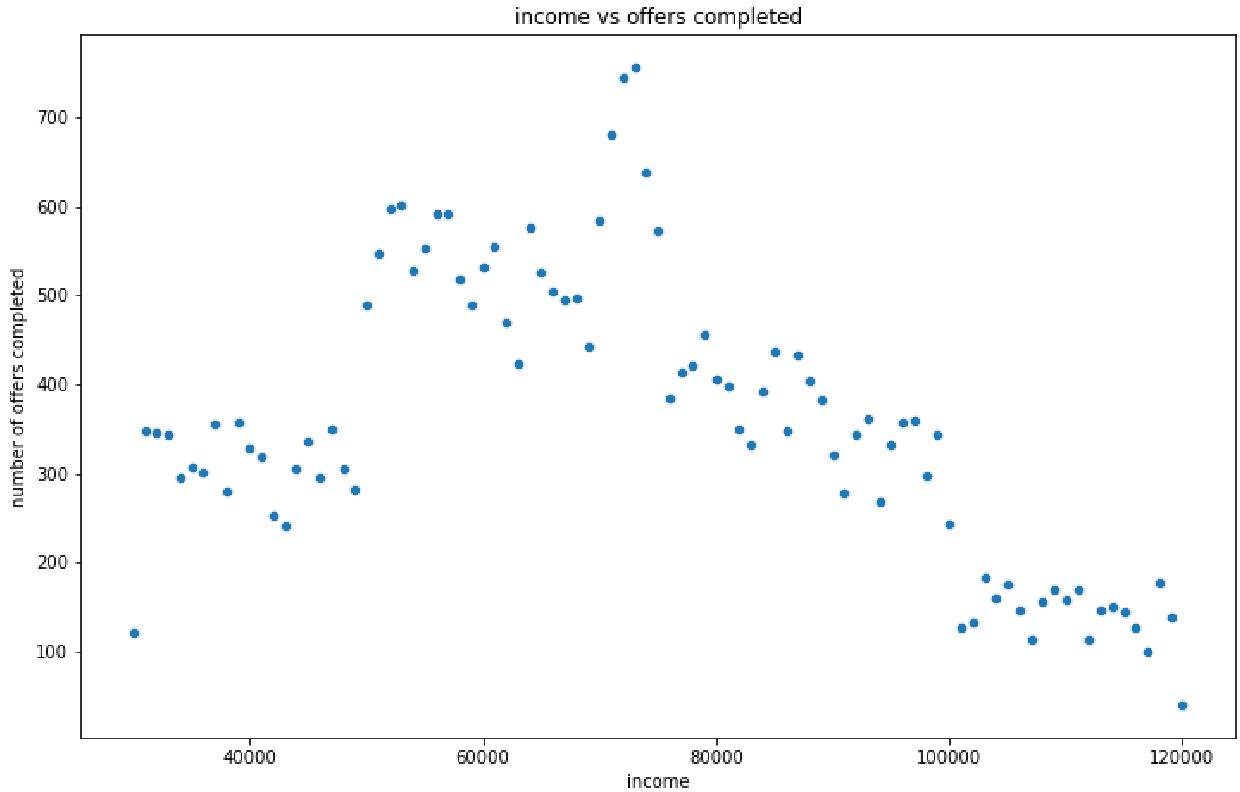
The majority of the customers are male, meaning that most of the transactions will be done by the male customers, if the offers were given to customers at random chances are that the male customers would get the majority of the offers.

The following plot shows which type of offer is likely to be completed.



The above plot shows that people are slightly more likely to complete a discount offer than a bogo type offer. Very few people view their discount offer but still complete them still. While the bogo type offers are viewed much more than discount but are not completed as much.

The below scatter plot shows the relationship of offer completion and income of the customer



It appears that for customers with high income complete few offers and the higher the income the fewers the offers.

Algorithms and Techniques

The benchmark algorithm is KNN, it is a simple to implement algorithm with simple to adjust hyperparameters. The algorithm computes the distance between data points and input points then classifies the input points as part of the majority points it is nearest to. After processing the StarBucks data, my approach to the problem started to change. I decided to make it into a binary classification problem. The algorithm determines if a customer would complete a given offer or not, to find an offer that a customer would complete you'll have to iterate through the offers and combine the data with customer data and input into the algorithm to get an output.
I used a Decision tree and a Random Forest classifier as well to see how they would do with the given data.

Benchmark

The benchmark model was inspired by the Netflix model [1] for recommendation. It uses a KNN algorithm. I have applied it on the given data and I could not get the same average precision, the model has a precision average in the 80% range. The recall I got is an average in the 60% range while netflix seems to have one in the 18% range. The benchmark model has an accuracy of ~69% and an average f1-score of 64%

	precision	recall	f1-score
0	0.63	0.40	0.49
1	0.71	0.86	0.78
accuracy			0.69
macro avg	0.67	0.63	0.64
weighted avg	0.68	0.69	0.67

Methodology

Data Preprocessing

How I processed the data for analysis and use.
I read in the data to my jupyter notebook using pandas.
I merged the data based on common unique columns.
I had to clean the column with the offer id first before it is usable, the column is below.

value
{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
{'amount': 0.35000000000000003}

After cleaning the column above , I only have the values in the dictionary. I removed the none id type of values

new_value
2906b810c7d4411798c6938adc9daaa5
2906b810c7d4411798c6938adc9daaa5
0b1e1539f2cc45b7b9fa7c272da2e1d7
0b1e1539f2cc45b7b9fa7c272da2e1d7

The column above has the cleaned values.

During analysis, I saved some of the data into files to save computation speed because they were time intensive.

Since some customers have had more than one transaction at StarBucks, their data appears more than once. I then grouped them by their unique customer id and the offer names which removes the repetitive nature of the data. The code snippet below shows how the data was processed

```
    #making dummy columns |event_offer completed|event_offer received|event_offer viewed|
    new_df_with_dummies = pd.get_dummies(df, columns=['event']).dropna()

    #The lines below are trying to get rows that show where an offer has been completed of not
    mean_df = new_df_with_dummies.groupby(['id_x', 'new_offer_names']).mean().drop(
              columns=['time','event_offer completed',
                       'event_offer received',
                       'event_offer viewed'])

    offers_df = new_df_with_dummies.groupby(['id_x','new_offer_names']).sum()[
                ['event_offer completed',
                 'event_offer received',
                 'event_offer viewed']
                ]

    formatted_df = mean_df.merge(offers_df, left_on=['id_x','new_offer_names'],
                             right_on=['id_x','new_offer_names'])\
                             .reset_index(level=['id_x','new_offer_names'])
```

Implementation

During the implementation of the algorithms, I loaded the data in the benchmark KNN model after it had been prepared and standardized. The resulting metric outputs were lower than I had hoped. Then I proceeded to implement tree type classification models, the models are Decision tree and Random forest. I made a dictionary with parameters to choose from during hyperparameter tuning to select the best model. The parameters are as follows:

```

parameters = {
    'max_depth':[1,2,3,4,5,6,8,9,20],
    'min_samples_leaf':[1,2,3,4,5,6, 8,10,15,20,50,100],
    'min_samples_split': [2,3,4,5,6,8,10,15,20,50,100],
    'criterion':['gini','entropy']
}

```

The models take in the prepared data as input and use the parameters to fit the best model. The models are evaluated on precision, recall and f1-score

Refinement

The training Algorithms I used in this project were set in such a way that the best hyperparameters were chosen from the get go of training. There is no initial model for any of the algorithms, only the best model chosen from the provided hyperparameters. The KNN model was trained for different numbers of K values and the best one was chosen. The Random forest and Decision tree classifiers had the same parameters and the best were chosen to be used for the model.

Results

Model Evaluation and Validation

During the training of the algorithms a test data set was used to check for the models' accuracy, precision, recall and f1-score. I compare these evaluation metrics from the different models and then decide on one with the highest of these metrics and f1-score being the most important of the metrics to look out for. All the models I have trained have the same f1 score.

Justification

```
best_estimator = chooseSearchGrid(d_tree, 'random', parameters, X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=6, min_samples_leaf=3,
                       min_samples_split=6, random_state=10)
```

```
prediction=best_estimator.predict(X_test)
```

```
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.63      0.43      0.51      6519
           1       0.72      0.85      0.78     11154

    accuracy                           0.69     17673
   macro avg       0.67      0.64      0.64     17673
weighted avg       0.68      0.69      0.68     17673
```

I chose this DecisionTree model as my final model. This model has a slightly better precision on non completed offers (0) this model will do ever so slightly better than other models. I chose this model based on recall of uncompleted offers represented by 0. The recall is 43% the other models I have trained have a recall just less than 43%. This decision tree model is slightly better than the benchmark model

Conclusion

The insights I have gained from the exploratory data analysis, can help in understanding the distribution of the customer demographic. I was able to find the surprising relationship between income and offers completed. I found that when a customer has high income the customer will likely complete fewer offers.I am able to tell that discount type offers are more likely to be completed than bogo type offers. The models trained have not given impressive results. I would need to go back to the data and do more analysis on it, I would need to try other techniques such as principal component analysis. I would need to do more research on similar projects to try and figure out the feature engineering techniques I could employ for this project

Reference

1. Molina, Leidy Esperanza. “Recommendation System for Netflix.” *Recommendation System for Netflix*, vol. 1, no. 1, 2018, p. 34. *www.cs.vu.nl*, https://www.cs.vu.nl/~sbhulai/papers/paper-fernandez.pdf. Accessed 14 january 2022.