Capstone Project

Mohd Zaki Aftab

Machine Learning Engineer Nanodegree

31st October 2020

Starbucks Project

Project Overview:

For this project, we were working to understand about how real-world data rarely comes clean. How can that data be cleaned and have meaningful representation of data. Which can be further utilised to send offers to various customers, regarding the current offers that Starbucks has exclusively for them, this project was quite challenging and the results were quite fine, We reported an accuracy of around 67% on the test data which is enough for sending offers.

To sum up the task:

The task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks sells dozens of products.

Problem Statement:

The prime thing here is to gather meaningful representation of data and conclude correct outputs from it. To add on the objective of the venture is to manufacture an AI model that can be utilized inside web application to handle genuine world, clients. I'd be focusing on creating an optimal model that would yield correct predictions on the dataset, that is to check if a particular offer is suitable for a customer or not. The data needs to be specifically be cleaned and meaningful correlation has been

found out between the three datasets. I have given my various conclusions as well, in the conclusions part. Creating a model that is suitable enough for doing the needful.

Metrics:

The information is part into train, test, and substantial dataset. The model is prepared utilizing the train dataset. We utilize the testing information to foresee the exhibition of the model on concealed information. We will utilize exactness as a measurement to assess our model on test information.

Accuracy=Number of things effectively characterized/All ordered things

Likewise, during model preparing, we contrast the test information expectation and approval dataset and ascertain Multi class log loss to locate the best performing model. Log loss assesses vulnerability of forecast dependent on the amount it fluctuates from genuine result and this will help in assessing the model.

Benchmark:

It is worth noting that Starbucks only had a 63.252% success rate during their trial with the offers they sent[6]. So my models have to beat 63.252% to be considered successful. My baseline was achieving an accuracy of above on test set and all my models seem to have achieved that. The base model which I kept was KNeighborsClassifier which offered 55% accuracy on the test set, I had to outperform that model, all my models did outperform it.

Data Exploration, Cleaning & Analysis:

The data has many biases, like the age which is 118 years, causing fluctuations, it has missing values as well, the characteristics of dataset is explained below:

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

The profile data had missing values in the gender and income tab which accounted to 2175 which is can cause errors in our model, so I decided to drop those values because they were the same. Here's the list of the empty dataset:

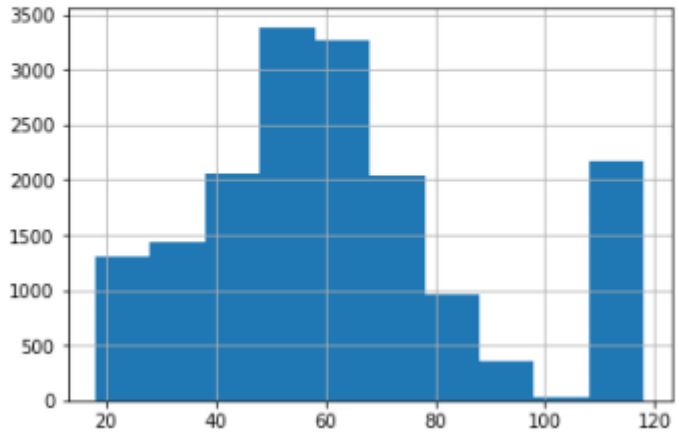
```
In [4]: #check for null values
        profile.isnull().sum()

Out[4]: gender              2175
        age                    0
        id                     0
        became_member_on       0
        income              2175
        dtype: int64
```

One more observation that I came across was the age of the customers, the customers were having ages as 118 and the histogram showed the fact that the number of people having age as 118 was too high, so I decided to drop the people having age as 118 or higher. Here's the histogram showing the number of people:

```
In [5]: #check distribution of age column
        profile.age.hist()

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x250b7673cc8>
```



The age was dropped because the data seemed too skewed due to that.

I decided to clean the data by dropping those values, those values were dropped by creating a missing value table on the dataset, which had 1 as a value if the data was missing and 0 if the data wasn't. I implemented the code and here's what I got as a cleaned profile.

```
In [8]: #storing 0s and 1s in new columns
        profile['missing'] = profile['age'].apply(lambda val: 1 if pd.isna(val) else 0)
        dropped = profile[profile['missing'] == 1].index
        # cleaned Dataset
        clean_profile = profile.drop(dropped)
```

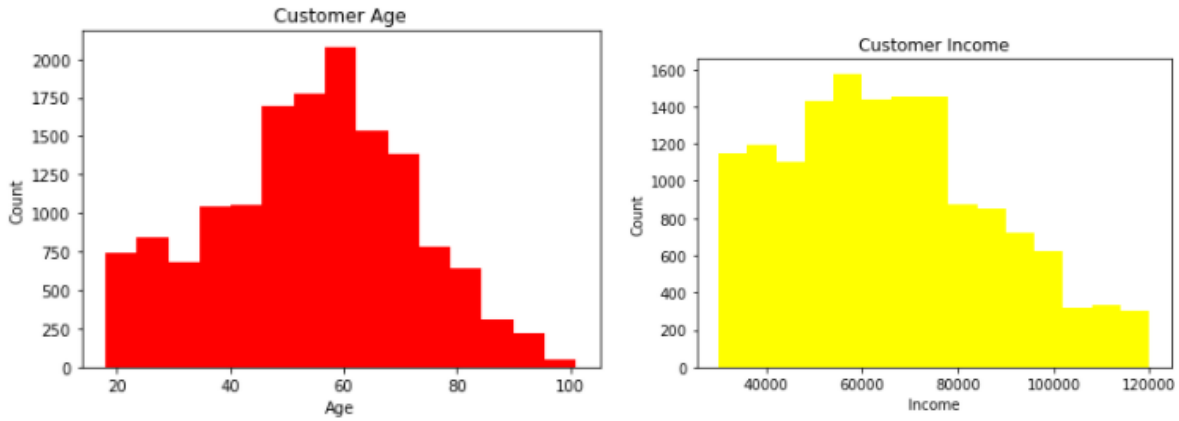
```
In [9]: clean_profile.head()
```

Out[9]:

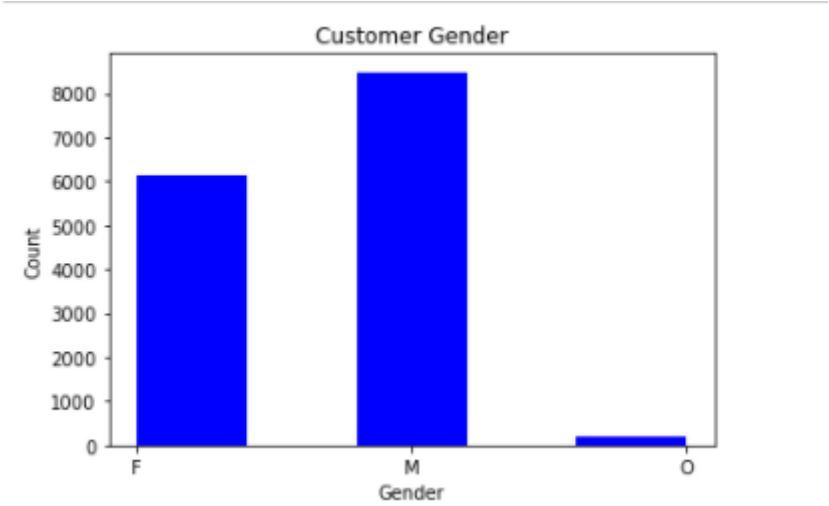
	gender	age	id	became_member_on	income	missing
0	F	55	0610b486422d4921ae7d2bf64640c50b	2017-07-15	112000.0	0
1	F	75	78afa995795e4d85b5d9ceeca43f5fef	2017-05-09	100000.0	0
2	M	68	e2127556f4f64592b11af22de27a7932	2018-04-26	70000.0	0
3	M	65	389bc3fa690240e798340f5a15918d5c	2018-02-09	53000.0	0
4	M	58	2eeac8d8feae4a8cad5a6af0499a211d	2017-11-11	51000.0	0

The notable observations that I came across was there were more male customers in the dataset. The mean income was around 65000 which seemed to be correct for a huge population. Next thing that can be noted was the customer age which had most customers from the age range of 50 to 60 years. One can also notice that the dataset had most memberships in the year 2017.

The data was cleaned and organised, it was sorted according to discounts and BOGO offer.



The other observation was Starbucks was having high number of males as their customers, which can lead to having more offers which can target specifically male customers.



One can notice that the above observations are quite useful in targeting the offers to the customer, the target population is the age of around 50-60 years where some meaningful offers can be sent. Like coffees which are comparatively strong and less sweet, these are important observations that Starbucks can notice.

Data Preparation & Preprocessing:

I started to prepare the data from the transcript.json file, the file had values for the event column as offer received/completed/viewed and transaction. I decided to create specific columns for every offer completed, received, viewed and transaction, for ease of access, this can help one in preparing the data easier. Here's the code and the transcript data frame:

In [13]:
```
x1 = transcript[transcript.event == 'transaction']
x2 = transcript[transcript.event == 'offer received']
x3 = transcript[transcript.event == 'offer viewed']
x4 = transcript[transcript.event == 'offer completed']
x1['amount'] = x1.value.apply(lambda x:list(x.values())[0])
x2['offer_id'] = x2.value.apply(lambda x:list(x.values())[0])
x3['offer_id'] = x3.value.apply(lambda x:list(x.values())[0])
x4['offer_id'] = x4.value.apply(lambda x:list(x.values())[0])
x4['reward'] = x4.value.apply(lambda x:list(x.values())[1])
transcript = pd.concat([x1,x2,x3,x4])
transcript = transcript[['event' , 'person' , 'offer_id' ,'time', 'amount' , 'reward' , 'value']]
transcript.drop(columns='value' , inplace=True)
```

For the time been I have kept the values for all new columns as zero except transaction, this would be corrected later. I dropped the duplicate values as well and here's how the head of the data frame looks like:

```
In [15]: transcript.head()
```

Out[15]:

	person	offer_id	time	amount	reward	offer completed	offer received	offer viewed	transaction
12654	02c083884c7d45b39cc68e1314fec56c	NaN	0	0.83	NaN	0	0	0	1
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	NaN	0	34.56	NaN	0	0	0	1
12659	54890f68699049c2a04d415abc25e717	NaN	0	13.23	NaN	0	0	0	1
12670	b2f1cd155b864803ad8334cdf13c4bd2	NaN	0	19.51	NaN	0	0	0	1
12671	fe97aa22dd3e48c8b143116a8403dd52	NaN	0	18.97	NaN	0	0	0	1

In the next few code cells the values of NaN has been converted to zero and the offer completed/received/viewed values has been assigned as either 0 or 1 depending on the respective values. I have also combined the transcript, portfolio and profile datasets to create a merged dataset which has categorical values for offers and transactions (either 0s or 1s). The merged file has all the values which shall be cleaned and sorted in the next step, due to the complexity of the code and the width of the data it is hard to quote it here.

In the last steps of data analysis, I have created a trans_data dataset which has transformed values of offers which are grouped by profile. I have sorted the offers by BoGo offers and discount offers as well. This has led to creating a fairly good dataset which has all the values we need! Again due to the width of the dataset I am unable to quote it here but I have quoted a few columns and some code for better understanding.

trans_data

person	age	became_member_on	gender	income	bogo_received	bogo_completed	discount_received	discount_completed	bogo_r
0009655768c64bdeb2e877511632db8f	33	2017-04-21	M	72000.0	1	1	2	2	1
0011e0d4e6b944f998e987f904e8c1e5	40	2018-01-09	O	57000.0	1	1	2	2	1
0020c2b971eb4e9188eac86d93036a77	59	2016-03-04	F	90000.0	2	1	2	2	
0020ccbbb6d84e358d3414a3ff76cffd	24	2016-11-11	F	60000.0	2	2	1	1	1
003d66b6608740288d6cc97a6903f4f0	26	2017-06-21	F	73000.0	0	0	3	3	
...	...	...	...	...	...	...	...	...	...
fff3ba4757bd42088c044ca26d73817a	69	2015-09-20	F	83000.0	1	1	3	2	1
fff7576017104bcc8677a8d63322b5e1	71	2017-10-31	M	73000.0	3	1	2	2	
fff8957ea8b240a6b5e634b6ee8eafcf	71	2018-02-18	M	56000.0	1	0	1	0	
fffad4f4828548d1b5583907f2e9906b	34	2017-01-23	M	34000.0	3	3	0	0	1
ffff82501cea40309d5fdd7edcca4a07	45	2016-11-25	F	62000.0	1	1	5	5	1

ows × 13 columns

Trans_data

```
In [26]: def offers_transformation(df):
             '''
             Function: to transform the offers

             Returns: The transformed offers
             '''

             df['bogo_received'] = 0
             df['bogo_completed'] = 0

             df['discount_received'] = 0
             df['discount_completed'] = 0

             for index , row in df.iterrows():
                 if(row['offer_type'] == 'bogo'):
                     df.loc[index , 'bogo_completed'] = row['offer completed']
                     df.loc[index , 'bogo_received'] = row['offer received']

                 elif(row['offer_type'] == 'discount'):
                     df.loc[index , 'discount_completed'] = row['offer completed']
                     df.loc[index , 'discount_received'] = row['offer received']



             return df.groupby(['person']).agg(
             {
                 'age':'last',
                 'became_member_on':'last',
                 'gender':'last',
                 'income':'last',
                 'bogo_received':sum,
                 'bogo_completed':sum,
                 'discount_received':sum,
                 'discount_completed':sum,
             }
```

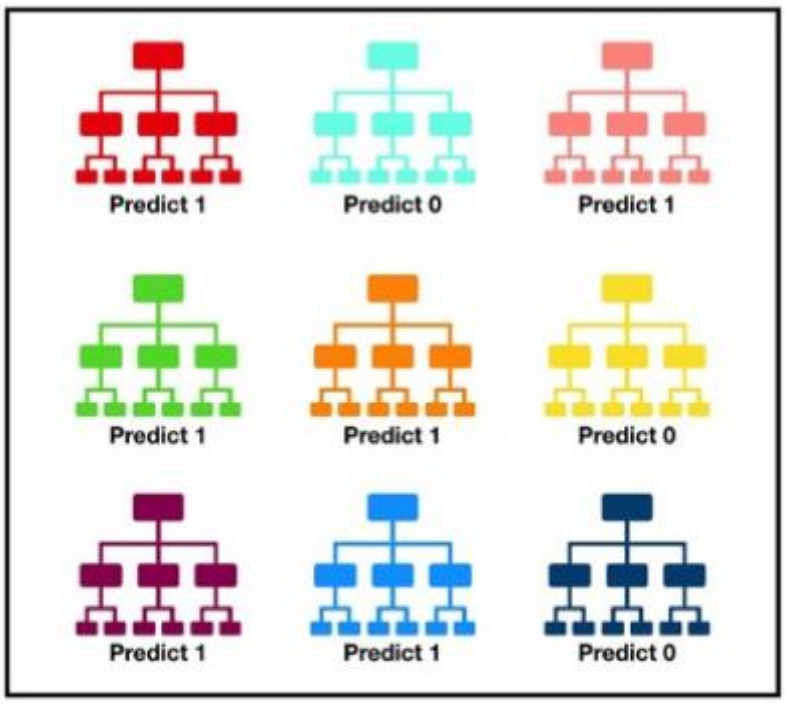
Here's the offers_transformation function which gives a data frame having discounts and bogo offers mentioned, which are grouped by their ages etc.

Algorithms:

I have tried various classifier algorithms on the cleaned data, which did fit quite well on the training data, the best being Extra Tree Classifier and Decision Tree Classifier, I have tried various other algorithms such as AdaBoostClassifier, GradientBoostingClassifier, RandomForestClassifier, KNeighborClassifier. The best accuracy on test data was found to be 67% by Gradient Boost Classifier, the less accuracy can be attributed to the complexity of the dataset. One can further refine data and try more complex algorithms to fit the data.

Extra Trees Classifier is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. [3]

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. [4]



Visualization of a Random Forest Model Making a Prediction

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Data Modelling:

The data was modelled in such a way that it would be able to predict discount offer or bogo offer when other variables are fed into it, the data was split using train_test_split function, then was fed onto various models.

Model Evaluation and Validation:

The highest accuracy on the training test was found to be 99.2%by Extra Tree classifier and Decision Tree Classifier, moreover Extra Tree Classifier is preferred due to the fact that it has more accuracy on test data as well, the best accuracy on test data was found to be 67% by Gradient Boost Classifier, the less accuracy can be attributed to the complexity of the dataset

```
Bogo offer F1_score 0.6673864147548358
Bogo offer Test F1_score 0.6035087719298246
Discount offer Training F1_score 0.9913630229419703
Discount offer Test F1_score 0.6159244264507422
```

Extra Tree Classifier

The model seems to fit the training data exceptionally but performs with an accuracy of 62% this seems to be quite problematic, but the data is too complex to have a classifier work this well, still the accuracy seems fine to me because we are only sending offers to users which doesn't needs to be too perfect having a model accuracy can of 62% is fine, I have also used various other models one of those being Gradient Boosting Classifier, which seems to perform well and uniformly on the data.

```
Bogo offer F1_score 0.6537112010796221
Bogo offer Test F1_score 0.6588394062078272
Discount offer Training F1_score 0.6764732343679712
Discount offer Test F1_score 0.6720647773279352
```

Gradient Boosting Classifier

Refinement:

The first model which I tried was KNeighbors, it yielded a satisfactory result of 55% I had to outperform the score, despite hyperparameter tuning, the model hardly witnessed a difference of 1% on the data. The regression models performed too poorly that they were not worth mentioning, they scored an accuracy of 10% which shows since this is a classification problem, we should try better classifiers. I

moved ahead to trying better classifiers, the hyperparameters were kept same in all the remaining models to ensure uniformity, which then resulted in a better performance than KNeighbors. The Gradient Boosting Classifier was better and provided accurate results. Hyperparameter tuning hardly made any difference so the hyperparameters were set to default and the results were displayed. The final product is trustworthy as it outperformed what Starbucks had achieved.

Justification & Final Discussion:

I think the model exhibition is same as anticipated. The model made utilizing Extra Trees Classifier has an accuracy of 99.2% on training data. The complexity of transcript dataset can be attributed to this. Therefore, I suggest that accuracy of 67% is ok, though we can improvise but that is by utilising more complex algorithm. Owing to the complexity of the dataset one can expect the test accuracy of the model to be around 60%. This model is basically about sending offers to people, so this amount of accuracy is fair enough. There is scope of improvement in Gradient Boosting Classifier, which can be a scope of further research, I personally had one objective that was evaluate the models, which I feel I have successfully done, also cleaning and other aspects have been totally done. The other notable observations have also been mentioned in the report as and when I came across it, I hope that this project have concluded critical observations. The model also did outperform what Starbucks had achieved which is a huge achievement of the project itself. Moreover, the model can be trusted at the first place because it has not overfit the data, the model performs better than the threshold limit and is suitable for deployment due to the fact that this had provided consistent results on trained as well on test data which approves of the robustness of the model.

References:

1. Udacity Starbucks dataset

2. Gradient Boosting Classifier: https://machinelearningmastery.com/gradient-boosting-machine-ensemble-in-python

3. Extra Trees Classifier: https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/#:~:text=Extremely%20Randomized%20Trees%20Classifier(Extra,to%20output%20it%27s%20classification%20result.

4. Random Forest Classifier: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

5. Benchmarking Machine learning prediction models: https://www.stratio.com/blog/benchmarking-machine-learning-prediction-models/

6. Benchmark: https://medium.com/@stephenblystone/starbucks-capstone-project-ccc477b6802c

7.