Starbucks-Customer-Clusters

Cluster analysis of Starbucks customers

The data set contains simulated data that mimics customer behaviour on the Starbucks rewards mobile app. 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). Some users might not receive any offer during certain weeks.
Not all users receive the same offer, and that is the challenge to solve with this data set.
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 actually sells dozens of products.
Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. It is evident in the data set that informational offers have a validity period even though these ads are merely providing information about a product; for example, 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.
The transactional data is given showing user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase. This transactional data also has a record for each offer that a user receives as well as a record for when a user actually views the offer. There are also records for when a user completes an offer.

Example:
To give an example, a user could receive a discount offer buy 10 dollars get 2 off on Monday. The offer is valid for 10 days from receipt. If the customer accumulates at least 10 dollars in purchases during the validity period, the customer completes the offer.
However, there are a few things to watch out for in this data set. Customers do not opt into the offers that they receive; in other words, a user can receive an offer, never actually view the offer, and still complete the offer. For example, a user might receive the "buy 10 dollars get 2 dollars off offer", but the user never opens the offer during the 10 day validity period. The customer spends 15 dollars during those ten days. There will be an offer completion record in the data set; however, the customer was not influenced by the offer because the customer never viewed the offer.

Data Exploration:

The Starbucks dataset is split up into three different files: **profile**, which has low level information about customers, **portfolio**, which has information about different promotional offers that can be received, and **transcript**, which has all purchase history and information on when the customer received, viewed, and completed their promotional offers.

a. Offer portfolio data

According to the information provided by Udacity, the schema is as follows:

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) -
- channels (list of strings)

Moreover, some further information given about the offers is that there are 3 different offer types:

- BOGO - buy one get one free
- Discount - discount with purchase
- Informational - provides information about products

Thus, the schema is straightforward, as it contains the attributes of 3 different offer types. While the duration was not explained I assumed from context that it is in terms of number of days.

```
portfolio.head()
```

	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 `channels` column consists of nested lists. Hence, I note that I will have to expand the column later during preprocessing to become categorical variables in my dataset. I also note that the scale of each are different, for example the `difficulty` is in terms of dollars while the `duration` is in terms of days. Hence, some feature scaling will need to be done.

b. Demographic data

Demographic data for customers is provided in the `profile` dataset. The schema and variables are as follows:

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

It is also relatively straightforward, as it contains the demographic profile on the customer.

In [6]:

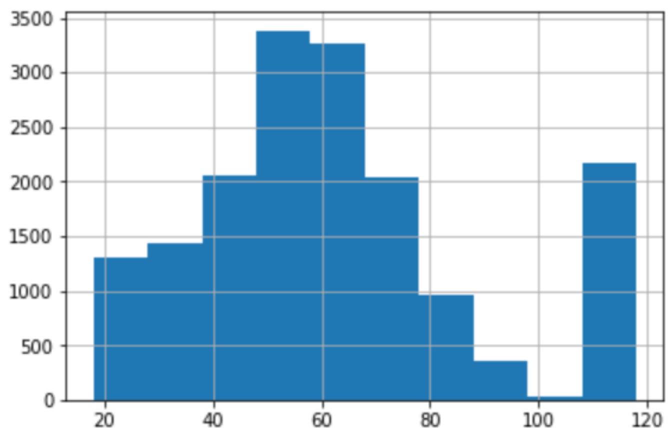
```
profile.head()
```

	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

From the first 5 lines we can already see some null values in `gender` and `income`, while the `age` column contains some values that don't make sense (e.g. 118).

```
#check distribution of age column
profile.age.hist()
```

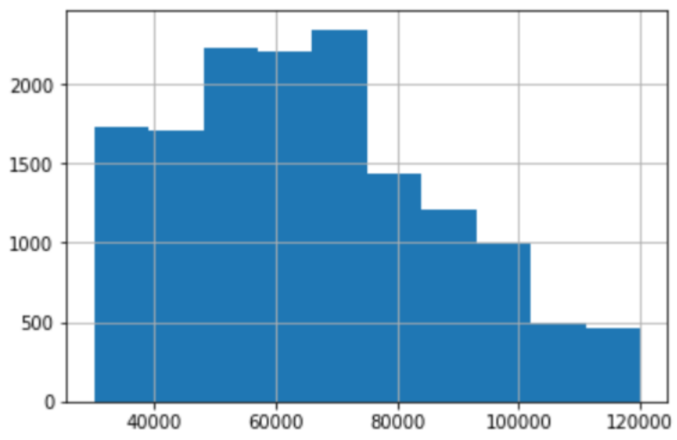
```
<matplotlib.axes._subplots.AxesSubplot at 0x26e2b3700f0>
```



We can see above that the `age = 118` value does not make sense there as it is clearly out of the normal distribution.

```
#check distributions of income
profile.income.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x26e2b3af710>
```



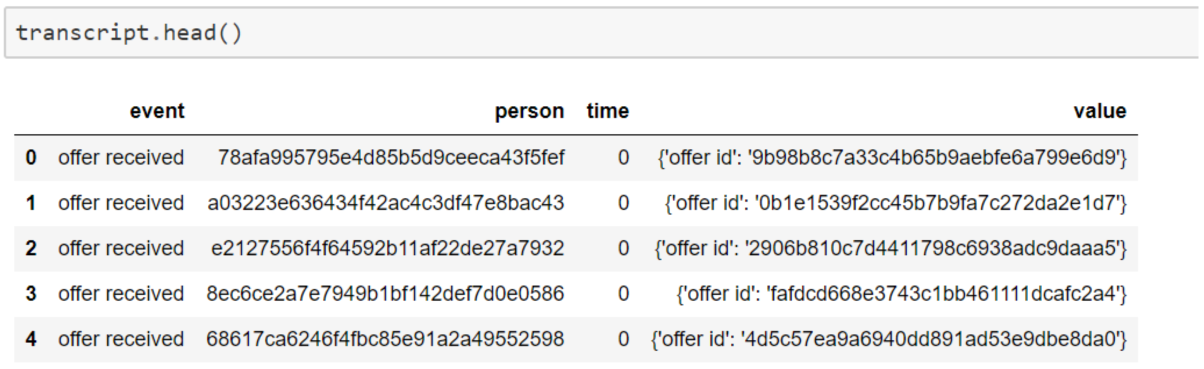
Last but not least, the `became_member_on` column has some potential to be feature engineered to get the tenure of membe
have some influence on whether an offer is effective or not.

c. Transactional records

The schema for the transactional data is as follows:

transcript.json

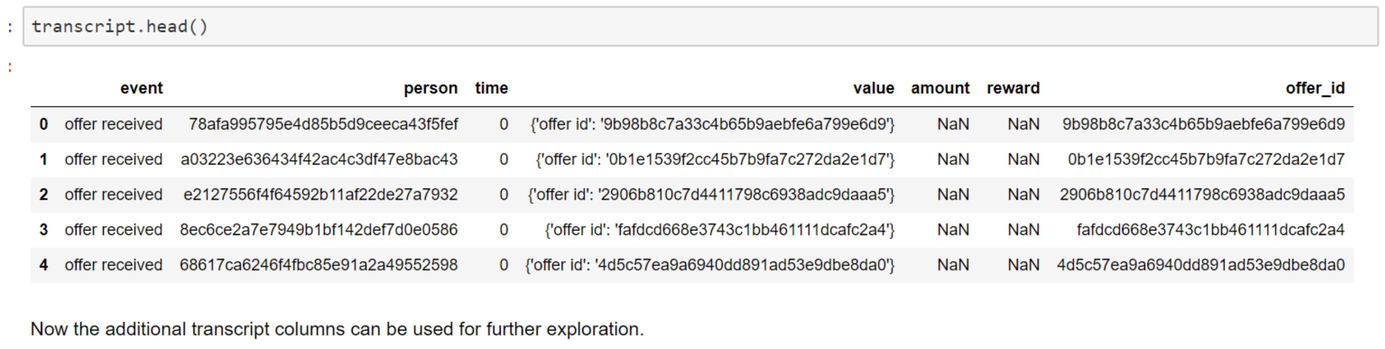
- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- person (str) - customer id
- time (int) - time in hours. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record



```
transcript.head()
```

	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'}

In order to extract insights from the value column, I will have to expand the values into individual columns depending on the event. It appears as though the offer id column ended up being duplicates so we have to clean it up further to ensure there is only one offer id column.



```
transcript.head()
```

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

Now the additional transcript columns can be used for further exploration.

Data Pre-Processing

a. Assigning offer ids to transactions

After defining the approach above, we now have to explore methods to assign offer_ids to specific transactions. Among the considerations is to define the following main groups of customers:

1. People who are influenced and successfully convert - effective offers:

- `offer received` -> `offer viewed` -> `transaction` -> `offer completed` (BOGO/discount offers)
- `offer received` -> `offer viewed` -> `transaction` (informational offer s - must be within validity period of offer)

2. People who received and viewed an offer but did not successfully convert - ineffective offers:

- `offer received` -> `offer viewed`

3. People who purchase/complete offers regardless of awareness of any offers:

- `transaction`
- `offer received` -> `transaction` -> `offer completed` -> `offer viewed`
- `transaction` -> `offer received` -> `offer completed` -> `offer viewed`
- `offer received` -> `transaction` -> `offer viewed` -> `offer completed`
- `offer received` -> `transaction` (informational offers)
- `offer received` -> `transaction` -> `offer viewed` (informational offer s)

4. People who received offers but no action taken:

- `offer received`

For people in group 2, I would need to check if there are events where there is an offer received and offer viewed event, but no conversion event, i.e. offer completed or transaction - these are cases of ineffective offers.

I would have to separate out the people in group 2 from people in group 4, as people in group 2 may have viewed an offer but did not take any action, whereas people in group 4 did not even have an offer viewed event.

Separating the conversions for effective offers (group 1) and people who purchase/complete offers regardless of awareness of any offers (group 3) is particularly tricky. For people in group 3, a conversion is invalid (i.e., not a successful conversion from an offer) if an offer completed or transaction occurs before an offer viewed. There also may be scenarios where an offer completed occurs after the offer is viewed, but a transaction was done prior to the offer being viewed. In this instance, the offer may have been completed, but it is also not a valid conversion.

Defining the target variable effective offer:

After defining these conditions, we have to decide what the target variable will be.

We know that group 1 customers will be our target variable effective_offer=1, but there are many ineffective offer definitions for groups 2-4.

So what would we define as an ineffective offer? As already stated above, group 2 would be within our definition of an ineffective offer; where a user is aware of an offer, but the offer is ineffective as it does not convert the user into a customer. So group 2 can be defined as our target variable effective_offer=0.

What about group 3 and group 4? Group 3 consists of users who may have received offers but would have purchased regardless. From the business point of view, we would not want to be sending them any offers.

Meanwhile, group 4 users would be considered low priority customers, as they do not do any action, regardless of whether they receive offers or not.

So, we can deprioritise group 3 and group 4 users from our model. It would still be worth doing some exploratory analysis onto group 3 and 4, just to explore on their demographics.

The conditions above are the basis of which I can assign the offer id that 'influences' a transaction by ensuring that the transaction occurs after an `offer viewed` event.

After sorting the transcript dataset by person and time to ensure that each event for each person occurs in sequence, I can filter the dataset by events `offer viewed` and `transaction` to ensure that it only contains those events in order.

Then, I can use pandas' `ffill()` method to fill every transaction with the offer_id of the viewed offer, only if it occurs before the transaction.

b. Flagging transactions and offers completed after offers viewed

The next important step for preparing our data for modeling and analysis is to identify a completed offer and transactions occurring after an offer is viewed.

Once we have assigned a transaction occurring after an offer is viewed, I can use that information to subset my data according to the groups defined above and analyse within each group.

Using our dataset with the offer_ids populated for `transaction` events, we can flag the converted transactions and completed offers. We have to first ensure that the offer id of the previous event is the same one. Since we have tagged the offer id for all viewed, transactions and completed offers, we can use the `offer_id` field to ensure that the previous offer consists of those events.

This means that as long as the events `offer viewed`,`transaction`, and `offer completed` occur in the same event space and are in the correct sequence of time, we can be assured that it is a transaction and/or completed offer occurring only after an offer is viewed.

To do this, I created a new column to flag the previous offer id using pandas' `shift` function.

c. Considering duration/validity of offers in converted transactions from informational offers

There is an additional rule to consider when considering an effective/converted transaction and offer. This applies for offers that are of type 'informational'. As already elaborated above, the reason why informational offers get a different treatment is because the conversion event is not an offer completed event, but a transaction.

For informational offers, the duration of the offer can be considered to be the duration of the influence. Hence, we can make the assumption that an offer should only be considered effective if it is within the duration of the offer.

Meanwhile, for BOGO and discount offers, we can assume that if there is a conversion/ offer completed event, it should be within duration as it would not make sense for an offer to be completed if an offer is past its validity period. With the valid_completed and valid_completed_duration flag columns, we have 4 possible scenarios for an informational offer within the transcript_info dataset:

No.	valid_completed	valid_completed_duration	Scenario
1	1	0	completed transaction after offer viewed event, but not within duration
2	0/null	1	completed transaction within duration, but with no offer viewed event prior
3	1	1	completed transaction within duration, with offer viewed event - **an effective offer**
4	0//null	0	did not complete transaction within duration, no offer viewed event prior

Following the above scenarios, only Scenario 3 would be considered our label effective_offers = 1 for informational offers (group 1 of customers).

Meanwhile, Scenarios 1 and 2 can be considered to be actions that would put the customer into our Group 3 of customers - People who purchase/complete offers regardless of awareness of any offers.

For customers in Scenario 1, even though according to our valid_completed flag, they had viewed an offer prior to the transaction, but it is not within the duration, thus they are not 'influenced' by the offer.

Meanwhile for customers in Scenario 2, they are in Group 3 as they completed transactions without viewing an offer.

Scenario 4 can be considered in group 4, as they only consist of transactions.

We will need to separate those users in group 2 - those who may have received and viewed an offer, but no transactions after. We need to subset those where effective_offer!=1 into groups 2,3 and 4.

d. Feature engineering

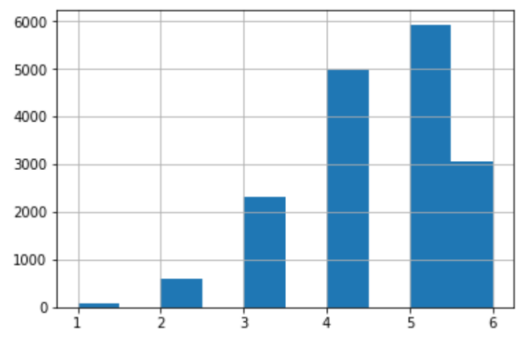
Now we must look back into the features and see how to be creative in creating new features.

d.i. column to be engineered

Recalling my preliminary data exploration steps, the became_member_on column were in date format. Hence in order to extract meaningful insights from that feature, we can convert it as a feature indicating tenure of membership. There could be some influence in how long someone has been a member, with whether he takes up an offer.

d.ii. Count of offers received

As part of some further data exploration, I discovered that there could be multiple offers received per person.



We can see above that the offer received per person in the transactional data could range from 1 to 6 offers received. I had the hypothesis that the frequency of offers received per person might result in more effective offers, so decided to engineer a feature `offer_received_cnt` to account for this frequency.

d.ii. Separating user behaviours by transactions

I also wondered how many transactions were considered 'invalid' by my definition. Ordinarily, these would be the sum of transactions done by people not in group 1. The objective of offers are to drive purchases, so it would already be the case that users with high spend in their transactions would be flagged as `effective_offers`.

We've already defined that there are people in groups 3 and 4, where they are separate pools of users who are loyal spenders, and already tend to purchase more, isolated from the the effect of offers.

But for users in group 1 have a high amount of 'invalid spend' outside of the effect of offers, there might be some predictive power onto the effectiveness of offers; since a loyal user might have a higher tendency of taking up an offer.

In my datasets, I had already separated the transactions who are conversions versus transactions who are just the users' normal purchasing behaviour. This is through the `valid_completed` column, where I checked if a transaction had an `offer viewed` event prior.

In the cases where `valid_completed`=1, I had already included them in my effective offers flag for BOGO and Discount offers. However, for those transactions where `valid_completed`=0, I have not considered them, and this could be a potential feature to include, as a proxy for the 'baseline' level of spending for a user.

The logic is to wonder if there is some baseline level of spending for users who are highly influenced by certain offers (in group 1), and group 2, and if there is some predictive power in this baseline level of 'invalid transactions' that can predict the propensity of a user to take up an offer.

d. iii. Time elapsed between offers received

I also wanted to include time as a potential feature into my dataset, but since the transactional data starts from time=0, I suspected it would not have been of much predictive power without some feature engineering. I had the hypothesis that if there were multiple offers received per person within a certain time period, there might be some predictive power in the time elapsed between offers received.

Preparing data for implementation

Now we can finally begin with preparing the data for modelling.

To do this, there are some additional preparation steps for each dataset. Recalling our initial preliminary data exploration, there are some steps to prepare the data:

a. Merge with temporary datasets created above to include engineered features

b. Drop missing values in `gender` column for demographic data; convert gender into dummy variables

c. Separate the `channel` column into categorical variables

d. Treatment of duplicate records

Implementation

Now that the datasets are ready, we can proceed to implementing the model. Revisiting our objective, we wanted to analyse the drivers of an effective offer, with the target variable being `effective_offer`.

Since we have 3 offer types, there are thus 3 different models to be built. Since we are predicting whether an offer would be effective or not, this is effectively a binary classification supervised learning model.

Metrics Evaluation:

BOGO offers model:
First we try to build the BOGO offers model. I initialize the models with some randomly chosen parameters to check the initial performance. If performance needs to be improved further, I will attempt Grid Search to find the optimal parameters.

```
DecisionTreeClassifier trained on 9829 samples.
MSE_train: 0.1770
MSE_test: 0.1823
Training accuracy:0.8230
Test accuracy:0.8177
             precision    recall  f1-score   support

          0     0.4797    0.2694    0.3450       438
          1     0.8553    0.9366    0.8941      2020

avg / total     0.7884    0.8177    0.7963      2458

RandomForestClassifier trained on 9829 samples.
MSE_train: 0.1670
MSE_test: 0.1786
Training accuracy:0.8330
Test accuracy:0.8214
             precision    recall  f1-score   support

          0     0.4906    0.0594    0.1059       438
          1     0.8287    0.9866    0.9008      2020

avg / total     0.7684    0.8214    0.7591      2458
```

The accuracy for Random Forest Classifier (RF) model actually ends up outperforming the Decision Tree Classifier (DT) model slightly, but overall the performance for both models is about the same (82.14% vs 81.77% respectively in terms of accuracy). Accuracy for a first attempt is quite good, more than 80%. I will try to tune the model further to get a better accuracy.
However, in terms of the F1 score, both models are below 80%, with the Random Forest model performing worse compared to the

Decision Tree Classifier, with 75.91% vs. 79.63%. To analyse this, we have to refer to the formula for Precision, Recall and F1 score:

Recall or Sensitivity or TPR (True Positive Rate):
According to sklearn documentation, the recall is intuitively the ability of the classifier to find all the positive samples.
Number of items correctly identified as positive out of total true positives: True Positives /(True Positives +False Negatives)

Precision:
According to the sklearn documentation, it is intuitively the ability of the classifier not to label as positive a sample that is negative.
Number of items correctly identified as positive out of total items identified as positive: True Positives /(True Positives + False Positives)

F1 Score:
Since my F-beta score is F1 with beta=1, I am weighting recall and precision as equally important.
The formula is given by the harmonic mean of precision and recall:
F1 = 2*Precision*Recall/(Precision + Recall)
We can see that the F1 scores for DT outperformed RF slightly, but both are lower than the accuracy. This would indicate that DT model is doing slightly better compared to RF at not misclassifying negative events as positive (meaning, misclassifying people on which offers are ineffective, as people on which offers would be effective).
The difference in F1 score vs accuracy indicate that there could are instances where both models are falsely classifying negatives as positives, likely due to the imbalance of classes. But the overall higher recall/accuracy compared to F1 score indicates that the model is predicting the positive case (i.e. where an offer is effective) more accurately compared to predicting the negative cases (i.e. where an offer is ineffective), which is expected given the uneven classes.
However, revisiting our use case, we are perhaps not as concerned with these misclassifications since we don't mind sending people

more offers than they would have liked; we would rather not miss anyone on which an offer would have been effective.
Given this case, I will still go with the RF model.
Since I aim to analyse the drivers of an effective offer, I will check the feature importance for the models after I have selected the best model from refinement.

Discount offers model:
I repeat the same steps above but with my offer_discounts dataset.

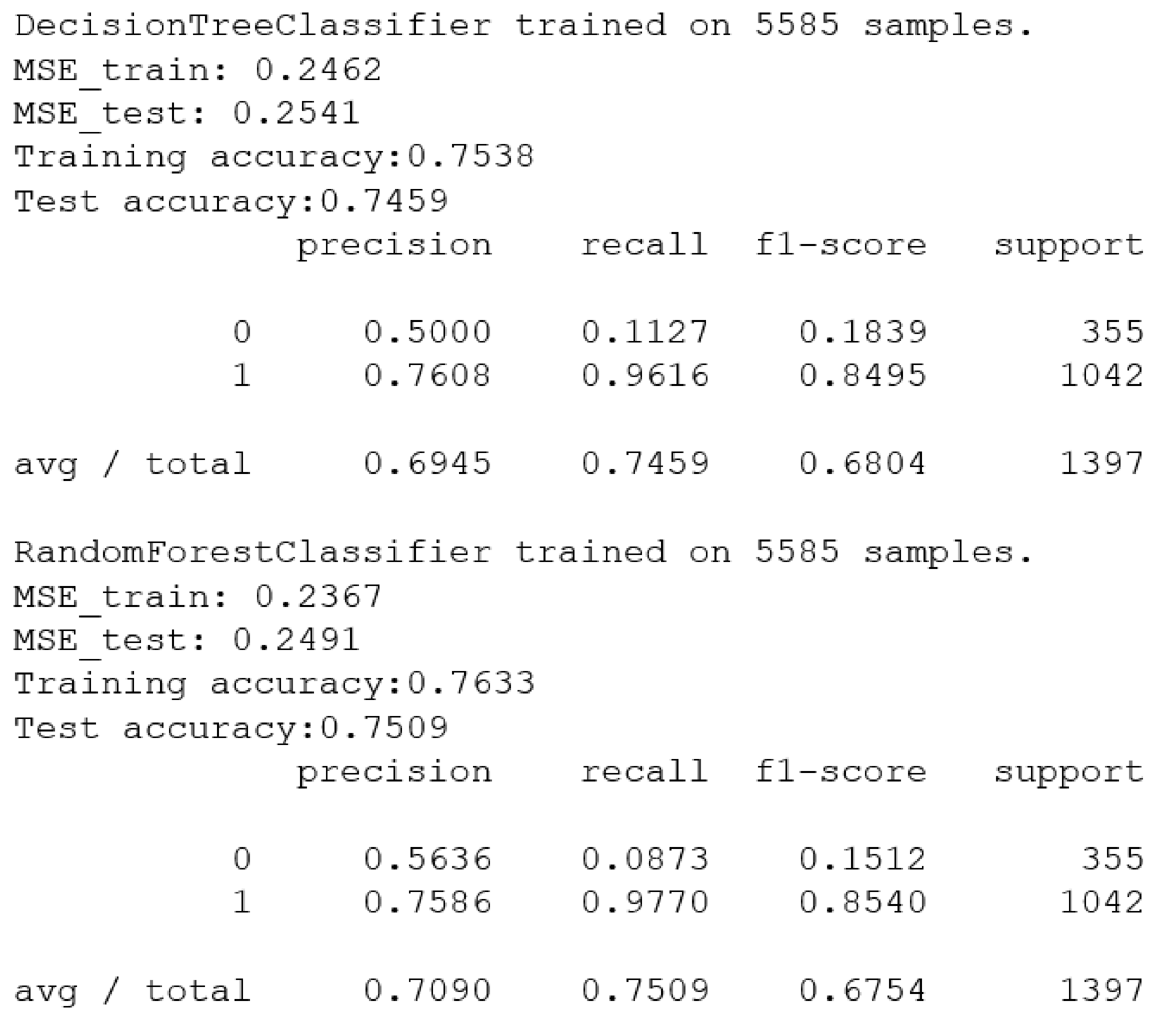
```
DecisionTreeClassifier trained on 10179 samples.
MSE_train: 0.1371
MSE_test: 0.1277
Training accuracy:0.8629
Test accuracy:0.8723
             precision    recall  f1-score   support

          0     0.0000    0.0000    0.0000       325
          1     0.8723    1.0000    0.9318      2220

avg / total     0.7609    0.8723    0.8128      2545
```

```
RandomForestClassifier trained on 10179 samples.
MSE_train: 0.1313
MSE_test: 0.1277
Training accuracy:0.8687
Test accuracy:0.8723
             precision    recall  f1-score   support

          0     0.5000    0.0062    0.0122       325
          1     0.8729    0.9991    0.9317      2220

avg / total     0.8253    0.8723    0.8143      2545
```

This time, the Random Forest Classifier model also has a better performance compared to the Decision Tree Classifier in terms of accuracy (87.23% vs 86.72%), and the F1 score is also lower (81.43% vs 82.87%).

The F1 score for these models are lower overall compared to the Accuracy score. This could be an indication that there are some instances where both models are classifying the negative cases (effective_offer = 0) falsely. Again, I am not too bothered by this as I am more concerned with the model predicting positive cases accurately, so would rather go with a higher accuracy model where F1 score for cases effective_offer=1 is higher, for which our RF classifier has better performance (0.9317 vs 0.9280).

Informational offers model:



```
DecisionTreeClassifier trained on 5585 samples.
MSE_train: 0.2462
MSE_test: 0.2541
Training accuracy:0.7538
Test accuracy:0.7459
             precision    recall  f1-score   support

          0     0.5000    0.1127    0.1839       355
          1     0.7608    0.9616    0.8495      1042

avg / total     0.6945    0.7459    0.6804      1397

RandomForestClassifier trained on 5585 samples.
MSE_train: 0.2367
MSE_test: 0.2491
Training accuracy:0.7633
Test accuracy:0.7509
             precision    recall  f1-score   support

          0     0.5636    0.0873    0.1512       355
          1     0.7586    0.9770    0.8540      1042

avg / total     0.7090    0.7509    0.6754      1397
```

The performance for these models are worse compared to the other 2 datasets, with accuracy below 80% for both models, but RF model still performing better. The F1 score is also worse, at 67.54% RF Classifier, worse than the DT model at 68.66%.
One potential reason for the worse performance is perhaps due to the fact that I had the key assumption to assign the conversion events to be

transactions that only occur after an offer is viewed and within the specified duration; I might have missed out on some valuable information by removing those transactions that occur regardless. We can see this from how the overall sample dataset is smaller (about half) the datasets for the other 2 offers, with only about 5K samples compared to about 10K for both BOGO and discount respectively.

Out of curiosity, I wondered if we could predict the effectiveness of an offer if the offer type was included as a categorical feature. Would the type of offer affect the user's responsiveness?
To do this, I would need to do some minor data preparation to prepare the data for a multiclass model.

```
:  drop_cols_prep=['person','offer_id','effective_offer','amount_invalid']
   features,target=data_prep(offers,drop_cols_prep)
   X_train, X_test, y_train, y_test=model_pipeline(features,target)

   #Initialize the model
   all_in_one = RandomForestClassifier(random_state=5,criterion='gini',max_depth= 20, max_features= 'auto',min_samples_sp
   lit= 2,n_estimators=50,min_samples_leaf=15)

   results=pd.concat([results[:],run_model(baseline,all_in_one,'all_in_one')],axis=1)
```

```
DecisionTreeClassifier trained on 25594 samples.
MSE_train: 0.1761
MSE_test: 0.1753
Training accuracy:0.8239
Test accuracy:0.8247
             precision    recall  f1-score   support

          0     0.5269    0.0778    0.1356      1131
          1     0.8326    0.9850    0.9024      5268

avg / total     0.7786    0.8247    0.7669      6399

RandomForestClassifier trained on 25594 samples.
MSE_train: 0.1638
MSE_test: 0.1749
Training accuracy:0.8362
Test accuracy:0.8251
             precision    recall  f1-score   support

          0     0.5316    0.0893    0.1529      1131
          1     0.8341    0.9831    0.9025      5268

avg / total     0.7806    0.8251    0.7700      6399
```

After Refinement:

	RandomForestClassifier_bogo_3	RandomForestClassifier_discount_3	RandomForestClassifier_info_2
pred_time	0.062512	0.058087	0.018027
testing_score	0.828316	0.873477	0.753042
train_time	0.181771	0.191597	0.060833
training_score	0.846882	0.869044	0.759534

Overall, we can see that the top performing models are the 3rd model (with GridSearch to find optimal model parameters and removing amount_invalid column) for predicting effectiveness of BOGO and discount offers, whereas the best performing model for informational offers was just after performing GridSearch to find the optimal parameters.

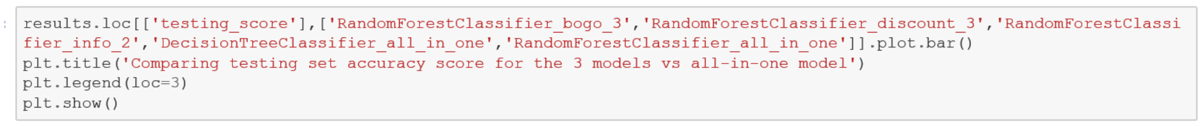
Comparative Analysis of different Classifiers

Random Forest Classifier:

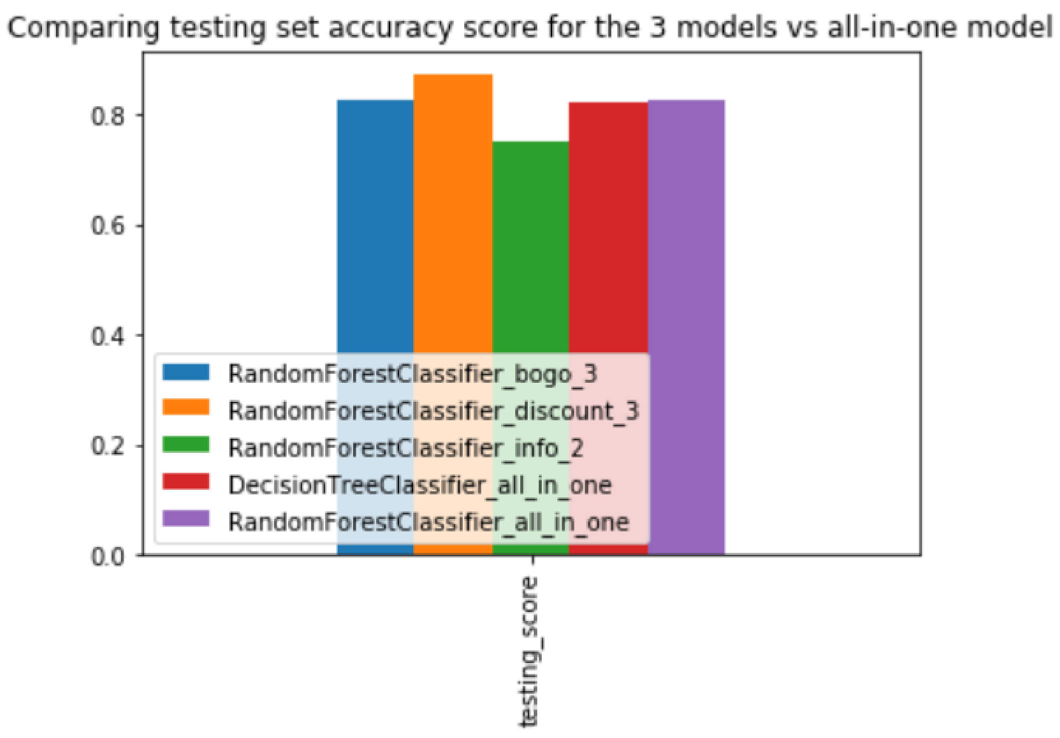
	RandomForestClassifier_bogo_3	**RandomForestClassifier_discount_3**	**RandomForestClassifier_info_2**
pred_time	0.062512	0.058087	0.018027
testing_score	0.828316	0.873477	0.753042
train_time	0.181771	0.191597	0.060833
training_score	0.846882	0.869044	0.759534

Decision Tree Classifier

	DecisionTreeClassifier_all_in_one	**RandomForestClassifier_all_in_one**
pred_time	0.000000	0.262158
testing_score	0.824660	0.825129
train_time	0.034128	0.838811
training_score	0.823904	0.836212



```
results.loc[['testing_score'],['RandomForestClassifier_bogo_3','RandomForestClassifier_discount_3','RandomForestClassi
fier_info_2','DecisionTreeClassifier_all_in_one','RandomForestClassifier_all_in_one']].plot.bar()
plt.title('Comparing testing set accuracy score for the 3 models vs all-in-one model')
plt.legend(loc=3)
plt.show()
```



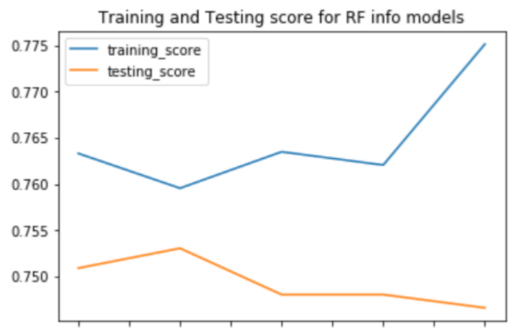
Comparing the performance of the 3 best models for each offer type with the all_in_one model, we can se that having the all-in-one model is not as good as the RF bogo and discount models, and is about slightly better than the info model. This is probably due to the info model pulling down the performance, resulting in lower accuracy for the all in one model. I suspect that if we were to break down the all-in-one model performance to just looking at its ability to predict the effectiveness of informational offer types, it would also be worse than its performance predicting the other 2 types.

If we take a step back and look at the big picture, it is more useful to have a higher accuracy for 3 separate models, as opposed to one all-in-one model. This is because the BOGO and discount offers are actually aimed at driving sales with some promotional cost, whereas the informational offer is essentially 'free' with no cost, and if they can drive sales that would be a bonus.

Hence, I would actually suggest that the 3 separate models are more useful.

I decided to compare the performance of a simple decision tree classifier model as a baseline model, with an ensemble random forest classifier model. Reason why I selected a decision tree as the baseline model is because I wanted to prioritise the interpretability of the model. Going back to the objective, since we intend to analyse the feature importance to determine the drivers of an effective offer, a decision tree would provide good interpretability for us to analyse.

Meanwhile, I also selected random forest as an alternate model to compare the baseline model is as an improvement over simple ensemble bagging of decision trees, in order to drive towards a high accuracy in training the model.



A note however, we can above actually see the model is performing better in the training accuracy as we add more variables for each model via polynomial features and removing the amount_invalid feature. It is just that the testing accuracy was reducing, and we can see this is due to overfitting.

I can improve the accuracy and performance of the info model further by using RF info model 5, but adding more data, as we already noted the dataset for the `offers_info` dataset is half the size of the BOGO and discount datasets. Hence, ultimately with more data and with performance tuning, removing unnecessary variables and feature transformation, with more data I could have ultimately got the performance of the model perhaps above 80%.

Now that I am done with refining the 3 models, we can check the results for our best models for all 3 and check the feature importances to see the top drivers of effectiveness of offers.

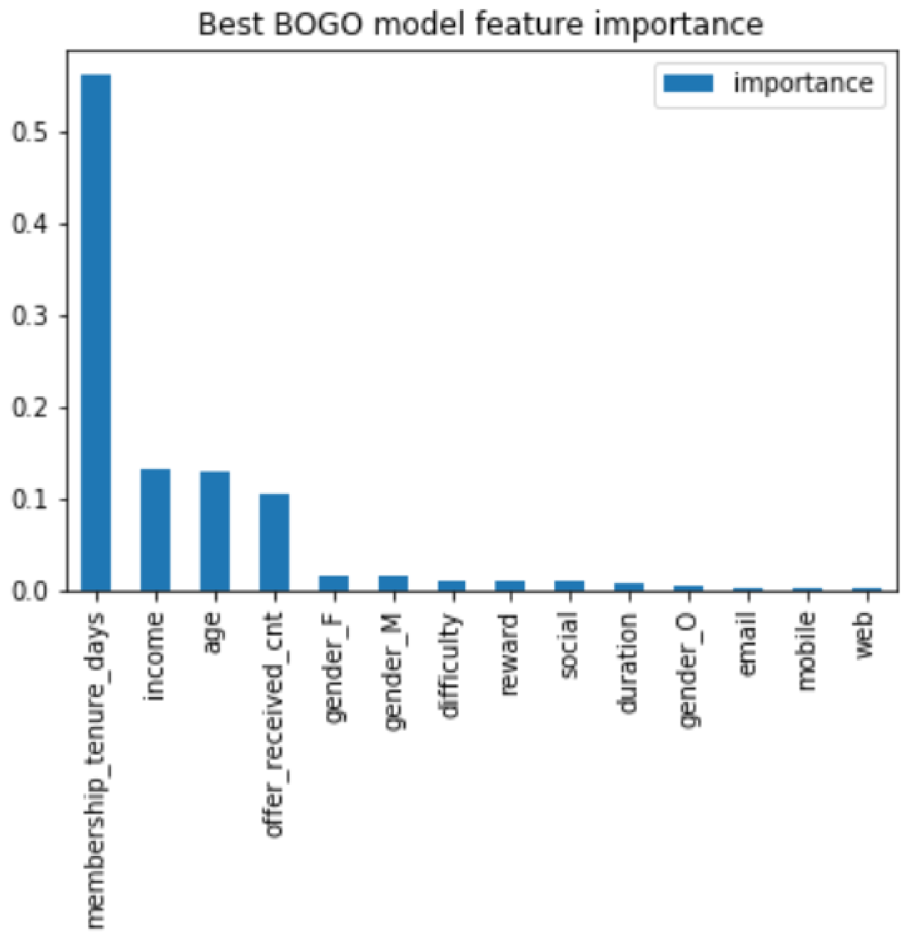
```
#get best model overall for bogo,discount and info offers
best_model('bogo').append([best_model('discount'),best_model('info')]).transpose()
```

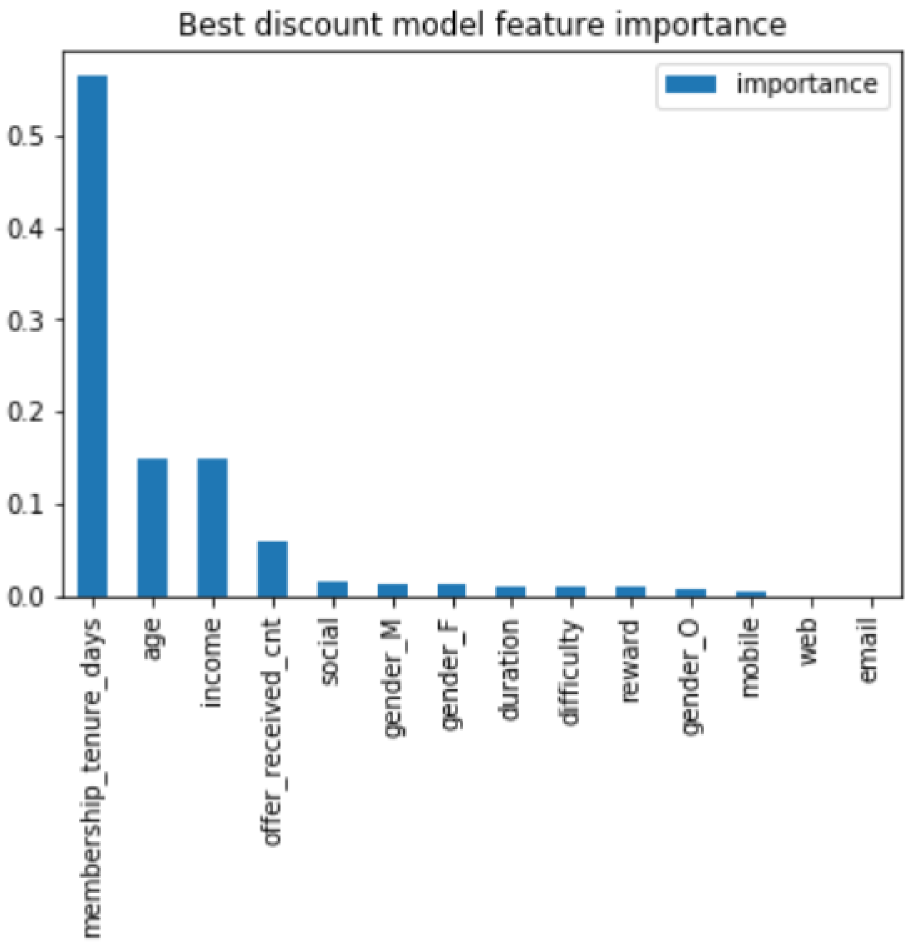
```
For bogo RF model:
For discount RF model:
For info RF model:
```

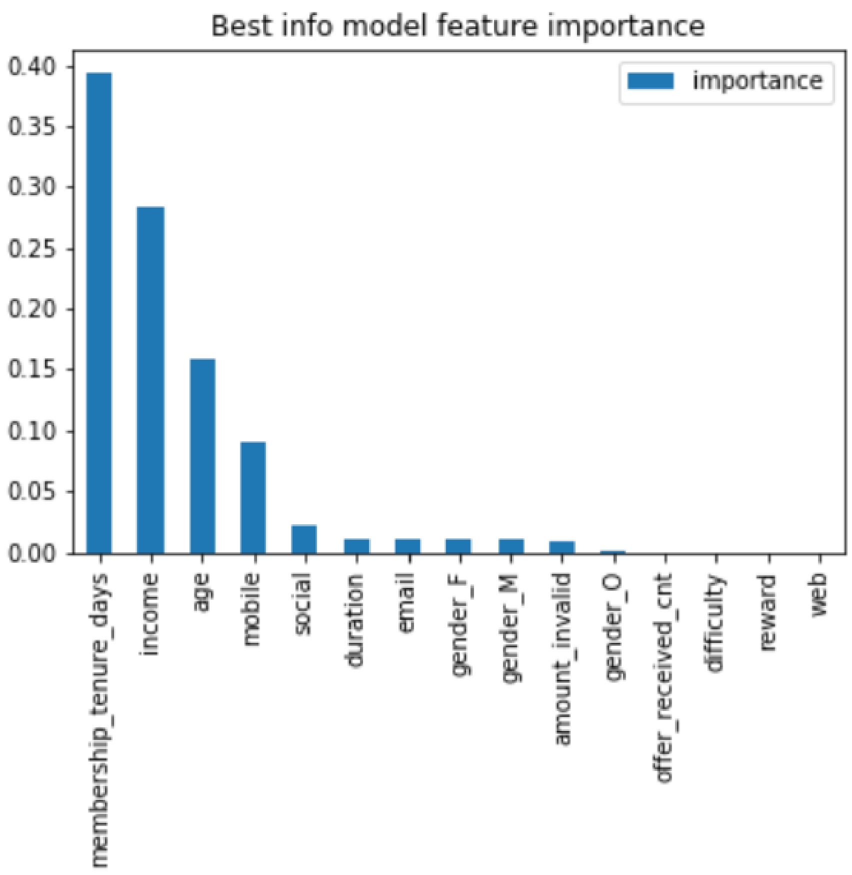
	RandomForestClassifier_bogo_3	RandomForestClassifier_discount_3	RandomForestClassifier_info_2
pred_time	0.062512	0.058087	0.018027
testing_score	0.828316	0.873477	0.753042
train_time	0.181771	0.191597	0.060833
training_score	0.846882	0.869044	0.759534

Overall, we can see that the top performing models are the 3rd model (with GridSearch to find optimal model parameters and removing amount_invalid column) for predicting effectiveness of BOGO and discount offers, whereas the best performing model for informational offers was just after performing GridSearch to find the optimal parameters.

In order to find the most influential drivers of an effective offer, we can check the feature importances of our best models above.


Best BOGO model feature importance
importance
0.0
0.1
0.2
0.3
0.4
0.5
membership_tenure_days
income
age
offer_received_cnt
gender_F
gender_M
difficulty
reward
social
duration
gender_O
email
mobile
web


Best discount model feature importance
importance
0.5
0.4
0.3
0.2
0.1
0.0
membership_tenure_days
age
income
offer_received_cnt
social
gender_M
gender_F
duration
difficulty
reward
gender_O
mobile
web
email

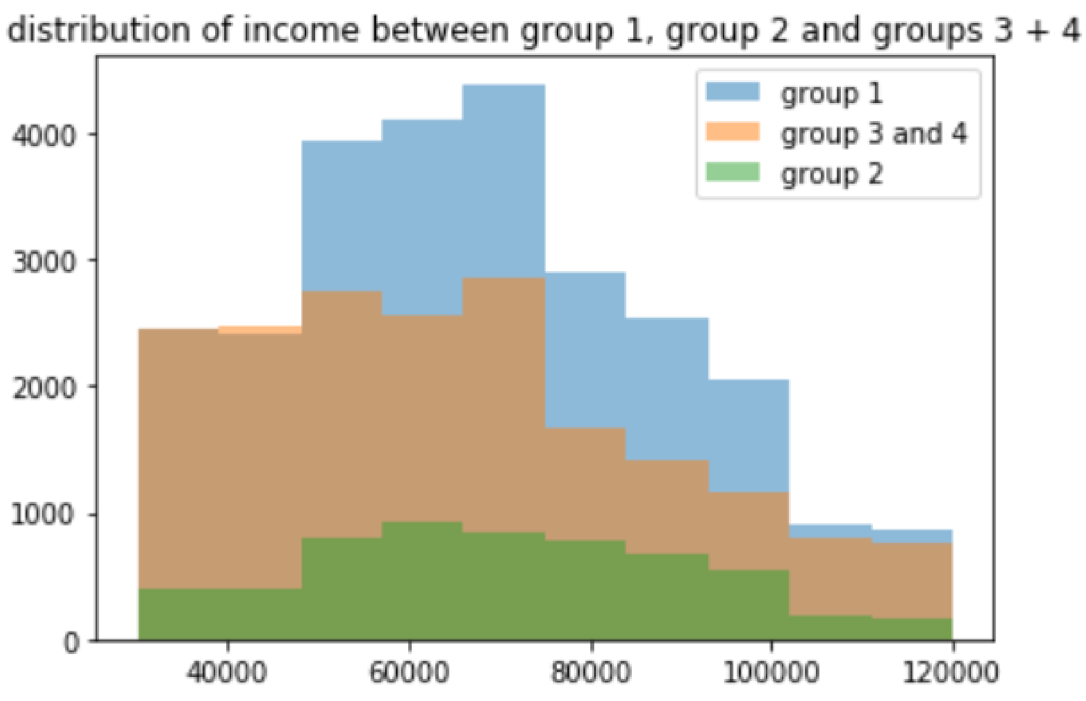


Checking on the feature importance to analyse the main drivers of an effective offer, we can see that the most important driver of effective offers across all three are the tenure of membership. However, the 2nd most important feature is different for each of the three models.

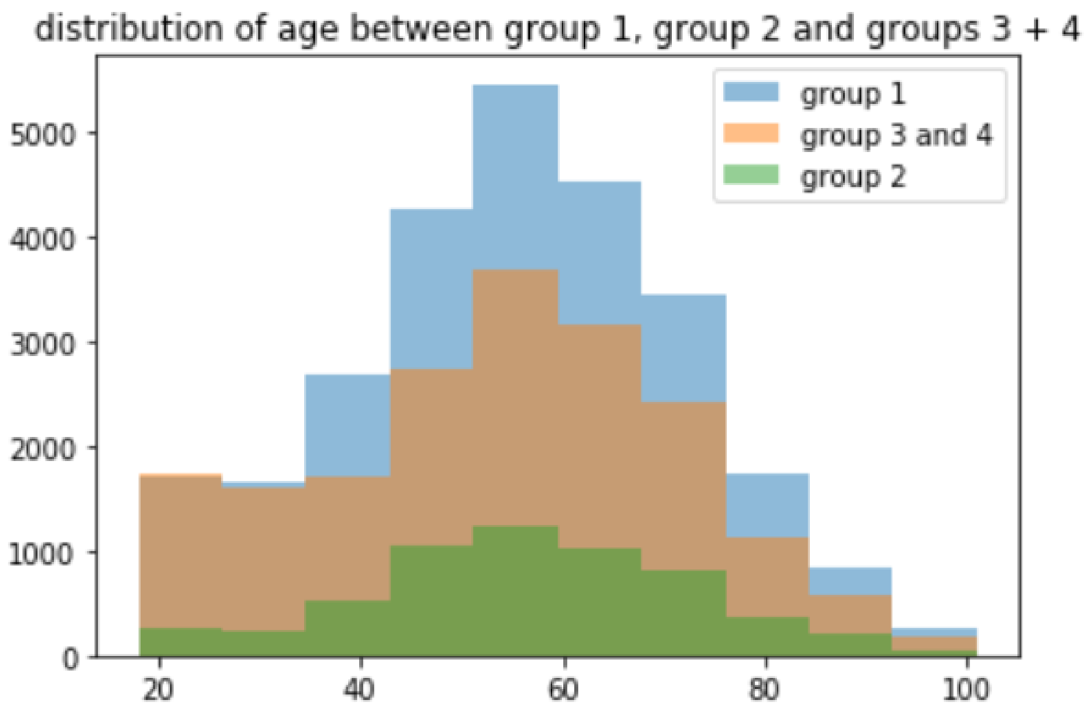
For a BOGO offer, the membership tenure is the most important feature, and the other variables are a lot smaller in proportions. Income, age and offer_received_cnt are the 2nd, 3rd and 4th most important features, but their proportions are very small.

For a discount offer, after the membership tenure, age and income are the next most important variables. But it is still very small in proportions.

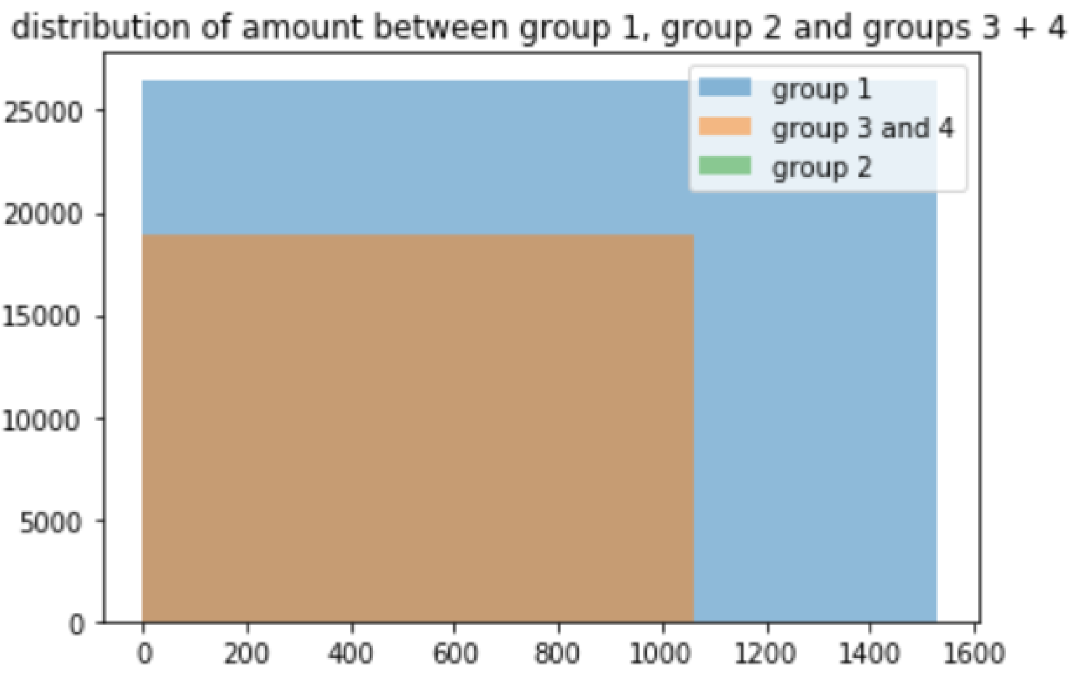
The feature importance's for the informational offer models are more distributed compared to the BOGO and discount models, with income being the 2nd most important feature. Age is the third and mobile channel interestingly being the 4th.



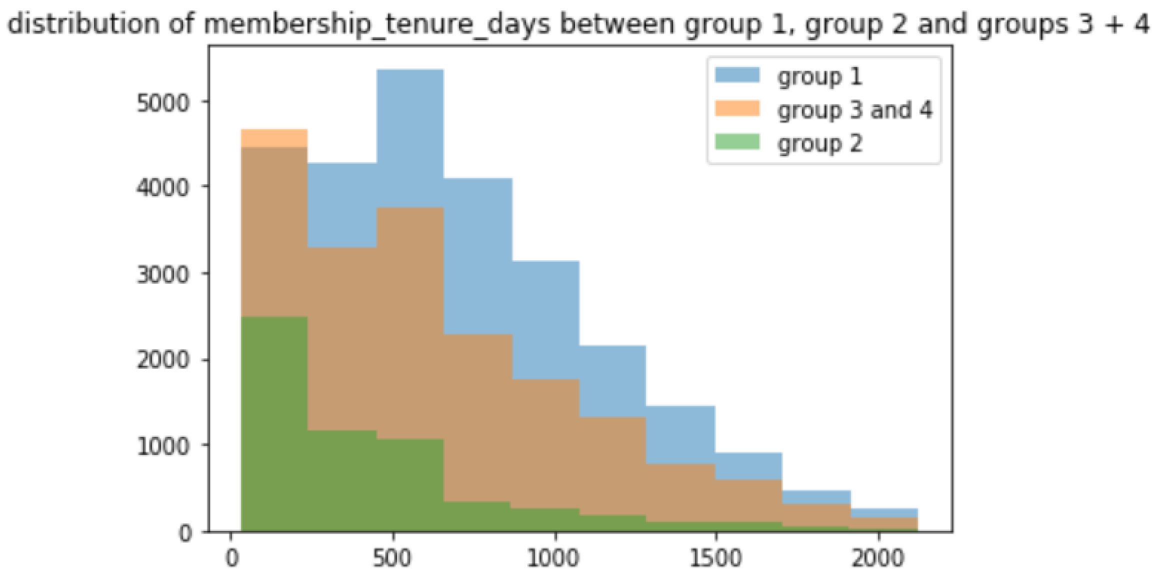
Across the 3 segments, most people fall within the middle range of income (50K - 100K). The income distribution between the 3 segments are relatively similar.



Age distribution looks relatively similar between the 3 groups as well, with most people between the age 40-80 years old.



Group 2 are people who did not spend at all as the offers were ineffective on them, hence they are not in the graph. But for groups 1 and 3+4, we can see that the amount spent is relatively similar, except that people in group 1 spent slightly more. This is to be expected as we might expect that the offers managed to incentivise them to purchase more, hence their overall spend increased.



The distribution of membership tenure also looks similar between the 3 segments, with most people between 0-700 days of tenure. It appears as though there are not much demographic characteristic differences between the 3 groups, at least in the current data provided.

Conclusion

Overall, I found this project challenging, mainly due to the structure of the data in the transcript dataset. I had started out with 2 business questions:

1. What are the main drivers of an effective offer on the Starbucks app?
2. Could the data provided, namely offer characteristics and user demographics, predict whether a user would take up an offer?

a. Reflection:

a.i. Question 1 findings:

For Question 1, the feature importance given by all 3 models were that the tenure of a member is the biggest predictor of the effectiveness of an offer. Further study would be able to indicate what average tenure days would result in an effective BOGO offer.

For all three models, the top 3 variables were the same - membership tenure, income and age. However, income and age switched orders depending on offer type.

For BOGO and discount offers, the distribution of feature importances were relatively equal. However, for informational offers, the distribution is slightly more balanced, with income the second most important variable.

a.ii. Question 2 findings:

My decision to use 3 separate models to predict the effectiveness of each offer type ended up with good accuracy for the BOGO and discount models (82.83% for BOGO and 87.35% for discount), while slightly less accurate performance for informational offers (75.3%). However, I would regard 75% as acceptable in a business setting, as for informational offers, there is no cost involved to inform users of a product.

Meanwhile, for BOGO and discount models, I am quite happy with the 80% and above accuracy, as in a business setting that would be acceptable to show offers to people, even if the model misclassifies a few, the overall revenue increase might justify the few mistakes.

b. Main challenges and potential improvement:

When analysing and building the machine learning models to answer the above questions, reflections on my main challenges and findings are as follows:

b.i. Attribution framework for assigning offer_ids for transactions:

In order to answer Question 1, I had to first define what an 'effective offer' means using the transactional records. This proved to be the trickiest portion of the project. I had to define a funnel for what what an effective conversion would look like, as we had data on both effective and noneffective conversions. Thus, I was desigining an attribution model for the conversion events (offer completed and transaction events) based on the events that occurred prior for each person.

I ended up having to separate the users into 4 different pools, based on their actions in the transcript data:

- Group 1: People who are influenced by offers and thus purchase/complete the offer(successful/effective conversion of offer)
- Group 2: People who receive and an offer but is not influenced and thus no conversion event (ineffective conversion of offer)
- Group 3: People who have conversion events but was not actually influenced by an offer
- Group 4: People who receive offers but no views or action taken

Even after separating the groups, it was challenging to assign the people in group 3 based on the transactional data. I had to define the event space where the right sequence of events would occur before I could assign an offer id to transactions (which did not have an offer_id), essentally designing a event/sequence-based attribution window.

After attributing the conversions to specific offers, the rest of the data preparation and cleaning was relatively straightforward. I was grateful that there were not many missing values, and the preparation of categorical variables was also relatively straightforward.

b.ii. Feature engineering:

I decided to do some basic feature engineering as I found the model had slightly underfit on my first attempt in this project, so I had added the feature engineering section later. It improved the performance of the model slightly, and the membership_tenure feature I had engineered out of the became_member_on column ended up being the most important predictor variable.

However, overall I found that I could not think of additional features using the time data, even though I had the hunch that the time of receiving the offer might be quite influential in determining whether it is effective or not.

b.iii. Model implementation decisions:

I had made the decision to build 3 separate models depending on offer types based on my definition of the problem statement - as I wanted to discover what would drive an effective offer, I thought it made more sense to remove noise from the data by separating the data into the offer types. My decision ended up to be quite a good one as the single BOGO and discount models got good performance in testing scores, compared to the all-in-one model overall score.

For the info model, the accuracy was slightly worse as we had less records overall (half of the BOGO and discount models). As elaborated above, I believe that if we had more data, I could have gotten the accuracy higher, as there was a clear diverging pattern occurring between the training and testing score as I made decisions to improve the model fit like adding polynomial features and removing 'noisy' features like the amount_invalid feature. Due to the limited data, my decisions ended up with the model overfitting, hence I believe the model accuracy would have benefitted from more data.

An additional note on model selection - I selected tree-based models as I wanted to assess feature importance, but I could have extended this study further by testing a parametric/ regression model (e.g. logistic regression for classification tasks). The weights of the coefficients from a regression model might have been interesting to contrast with the feature importance of a tree-based model, given that both models have different ways of analysing the data. The feature membership_tenure_days might not have been the highest weighted feature, in contrast to how it was in this study.

b.iv. Exploring demographics of different customer groups:

I was curious to know what the characteristics were of groups 3 and 4, which are customers who are not influenced by an offer at all. However, after comparing their characteristics with groups 1 and 2, I could not see any significant differences in their demographics.

I would have liked to have more data to perhaps understand why this group of customers tend to not be influenced by offer, in order to make useful suggestions on how to give a good customer experience to these customers, even if we do not serve them any offers.

b.v. Model accuracy in predicting amount spent given an effective offer:

The regression model I built out of curiosity to see if we could predict the amount a user would spend, given that they are effectively influenced by an offer. The motivation was that if we can predict how much someone would spend given an offer, perhaps we can assess which offers bring in the most revenue.

However, my model found virtually no correlation between the features provided (namely, offer characteristics and demographics of app users) with the amount spent per user. These features aren't strong enough to predict the amount spent per user. Perhaps if we also have a value of the offer, for example, for a discount offer, the value of the discount in dollar terms, perhaps we might be able to predict better.

Perhaps I could have broken them up into 3 different models for the 3 offer types, the way I did with the binary classification models, in order to get a better result. However, given that this was just a curiosity and I wanted to explore if the offer type would be a statistically significant predictor feature, I built an all-in-one model for this instance. This would be worth exploring further, given more time and data.