Predicting Effective Starbucks Offers

A Udacity Project

Benjamin Redmond

September 2021

Contents

Abstract

This is a Udacity Machine Learning Capstone project. It uses simulated data from Starbucks to predict the likelihood of a customer to make purchases in a response to receiving an offer from Starbucks. It compares the predictive power of Scikit-Learn machine learning algorithms to that of PyTorch's deep learning neural networks in this problem space.

1 Introduction

This project aims to show how machine learning can be used to provide efficient targeted advertising. The advertising world is continuously becoming more digital and that brings about a vast amount of data that can be harnessed to provide powerful information for marketers. Advertising strategies have been researched extensively to improve the effectiveness of advertisements across various target audiences [1].

ML techniques enhance the accuracy of targeting by predicting the most relevant adverts for users based on preexisting user data [2]. It is possible to utilise this information to provide a better return on advertising spend. More and more marketers are turning to this method to better decide which adverts will have a positive impact on which demographics.

Used in this way ML aims to take past information about customers responses to advertisements and make predictions and build models about customer behaviour in the future. It examines the historical data and detects patterns autonomously. These patterns are often hidden from humans attempting to analyse data without leveraging ML. The models can then be used to continuously improve marketing by refining its ability to influence the desired audience.

2 Problem Statement

This project uses simulate Starbucks transaction data to see how customers who use the Starbucks mobile app react to adverts they receive. The objective is to use trends within the customer data to improve the efficiency of the adverts sent. If certain adverts are targeted at customers who are known to be more likely to be receptive to these adverts, then it will be possible to get a better return on advertising spend. It will also provide a better customer experience if we can stop sending adverts to customers who we think have little to no interest in them. Neither the company nor the customer want redundant adverts sent.

Once every few days, Starbucks sends out an offer to users of the mobile app. The adverts in this project can either be a simple advertisement for a drink or an actual offer such as a discount of BOGO (buy one get one free). Every offer has a validity period before expiring. Customers receive varying adverts at varying intervals. Some customers might not receive any offers during certain weeks.

Some basic demographic data is provided about the users. The time and value of the transactions they made are also provided. It is also possible to see if they viewed the offer which is of relevance because the customer could have made a purchase without seeing the offer. The goal will be to identify the most appropriate offer to send to a customer such that they view the offer and then complete a purchase before the offer expires.

The offers will have two classes; successful and unsuccessful. It is important to clearly define what constitutes a successful or unsuccessful offer. An offer can be marked as completed after a customer makes a purchase without the customer ever being aware that they received an offer.

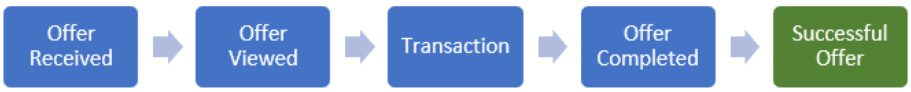

Figure 1: Successful discount and BOGO flow

Successful discount and BOGO offers are ones that follow the flow in

Figure 2: Successful informational flow

figure 1 on page 5. These events all have to occur within the duration period of the offer being received to be deemed a success.

Successful Informational offers are ones that follow flow in figure 2 on page 6. Similarly to discount and BOGO offers they have to occur in order before the offer expires. Other events can occur in the meantime whether or not another offer is involved. For example an informational offer can have a transaction, and then be viewed, and then have another transaction. If all these events occur before the offer expires then the offer is deemed a success.

I will class all other offers as unsuccessful offers. These offers could involve:

- Customers who never viewed the offer
- Customers who viewed the offer and didn't make a transaction
- Customers who didn't make enough transactions to complete the difficulty of the offer (discount or BOGO only)
- customers who completed the offer before viewing the offer

I am more focused on maximising completed offers and not discerning between unsuccessful ones. For that reason I will put all events which I don't see as successful conversions in the negative class.

My strategy for classifying the orders and then building a system that will optimise the likelihood that an offer sent will lead to a successful completion will broadly follow the steps below.

1. **Data Exploration** - I will examine all the data sets and features in isolation and in relation to each other. This will help me uncover patterns to help me decide how to build the model later on. Information

found here will form the basis for my overall strategy for the project. It will also show me which features may need cleaning.

2. **Data Preparation** - I will clean the data and join the tables together. I will then implement the logic for classifying each individual order.

3. **Benchmark Model** - I will examine various supervised machine learning algorithms from sklearn. I will try to find the optimal algorithm for this classification problem and then refine it using different techniques. The best result I can obtain will be my benchmark model.

4. **Neural Network Model** - I will build a neural network using PyTorch. I will use the same improved features as my refined benchmark model.

5. **Compare Models** - I will compare the benchmark model's results with the neural network's results and interpret them.

6. **Conclusion** - I will give my final thoughts on the project along with potential improvements and alternative ideas for approaching this project.

3 Analysis

3.1 Datasets Overview

The data sets are provided in JSON files by Udacity. It contains simulated data that mimics Starbucks data as briefly touched upon in the section above. Three files are provided

- portfolio.json – Offer ids and meta information about each offer
- profile.json – Demographic data for each customer
- transcript.json – Records for transactions and information on offers viewed, received or completed

3.1.1 portfolio.json

This data set describes the different offers Starbucks sends to users of it's app. It contains 10 rows and 6 columns.

- reward (int) - reward given for completing an offer
- channels (list of strings) – medium used (email, mobile, social, web)
- difficulty (int) - minimum required spend to complete an offer
- duration (int) – number of days before an offer expires
- offer_type (string) - type of offer (BOGO, discount, informational)
- id (string) - offer id

3.1.2 profile.json

This data set contains the distinct users and some basic demographic data. It contains 17000 rows and 5 columns.

- gender (str) - gender of the customer (Male, Female, Other)
- age (int) - age of the customer
- id (str) - customer id
- became member on (int) - date when customer created an app account
- income (float) - customer's income

3.1.3 transcript.json

This data set contains all the offer events and all the transactions that occur. It contains 306534 rows and 4 columns.

- person (str) - customer id

- event (str) - record description (transaction, offer received, offer viewed, offer completed)
- value - (dict of strings) - either an offer id or transaction amount depending on the record
- time (int) - time in hours since start of test

3.2 Data Exploration

3.2.1 Portfolio Data

I see that there are four discount, four BOGO and two informational offers. The informational offers have no difficulty or reward as expected. The non-informational offers have a difficulty range from 5 to 20 dollars. They also have a reward ranging from 2 to 10 dollars. The duration across all offers range from 3 to 10 days.

I also note that while the offers use a different combination of channels to inform the customers that they all use email as a channel. This indicates that I won't be able to extract any useful information from this channel. Therefore, I intend to remove email from all channels.

3.2.2 Profile Data

All 17000 id's are unique which important. There are no duplicate customers and it will make joining to the other data sets straightforward.

I plotted the ages in figure 3 on page 10 and saw the data looked odd. It showed a large number of ages equal to 118. This is clearly not expected. So I dug down into the data and saw these 118 values occur on the same rows where gender and income contain null values. It became apparent that the 118 value for age was how the data indicated a null age. In total 2175 rows have these null values which represents nearly 13% of the total rows. I filtered out this age and plotted the ages again in figure 4 on page 11 and the data looked much better. I think it makes sense to remove these values but

I will do more examination on these values before making a final decision on that.

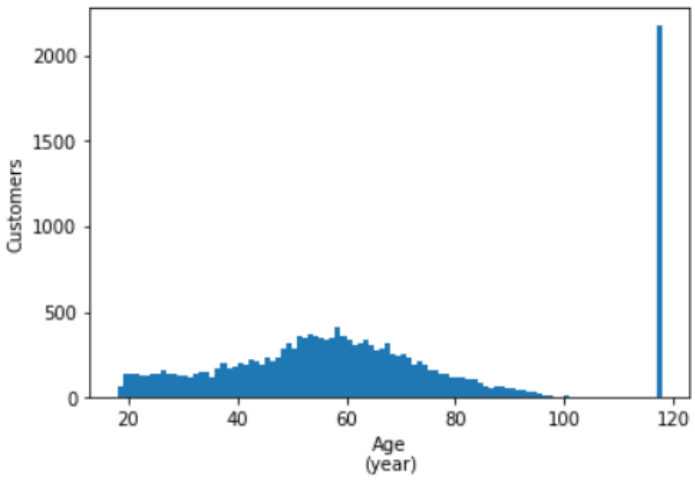

Figure 3: Customers by age

There is plenty more men than female in this study as we can see in figure 5 on page 11. But that is not an issue. There are a few non male nor female genders too.

Next I examined the income field. It all appeared suitably spread out as I expected. I examined it by gender too in figure 6 on page 11 to confirm it still appeared as expected. It does show some interesting patterns, but nothing to indicate there might be bad data. I do not need to do any filtering based on income.

I decided to examine the relationship between age and income. I would expect income to trend upwards with age. As you can see in figure 7 on page 12 the average income is positively correlated with age although the line is not as smooth as expected. This jagged line further indicates to me that the sample data generated is not an ideal real world example, although the positive correlation indicates it is probably an acceptable replica.

The field showing when a person first became a customer is heavily skewed towards more recent dates with a peak in late 2017. This can be seen in figure 8 on page 13. This does not seem an obvious indication of bad data either,

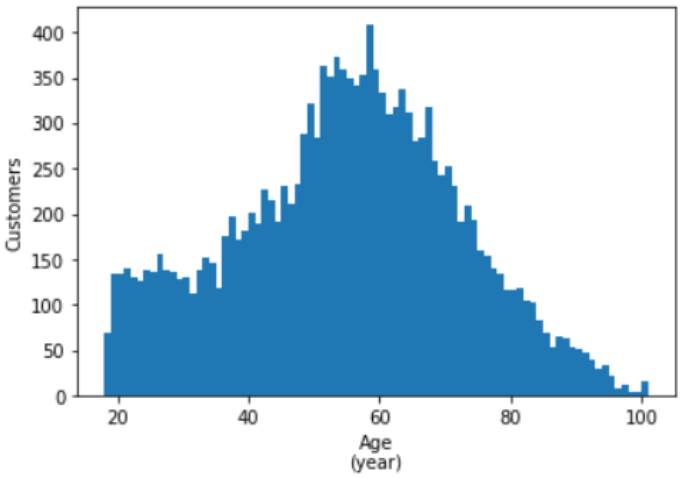

Figure 4: Customers by age after removing nulls

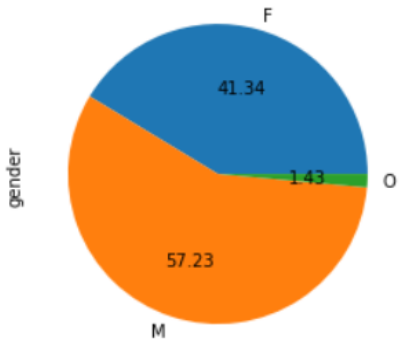

Figure 5: Gender ratio

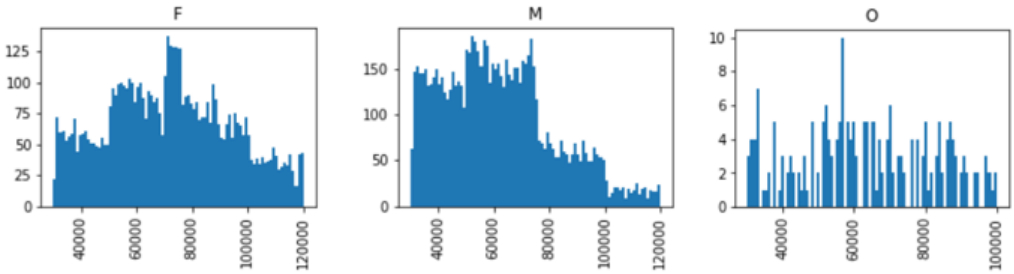

Figure 6: Income by gender

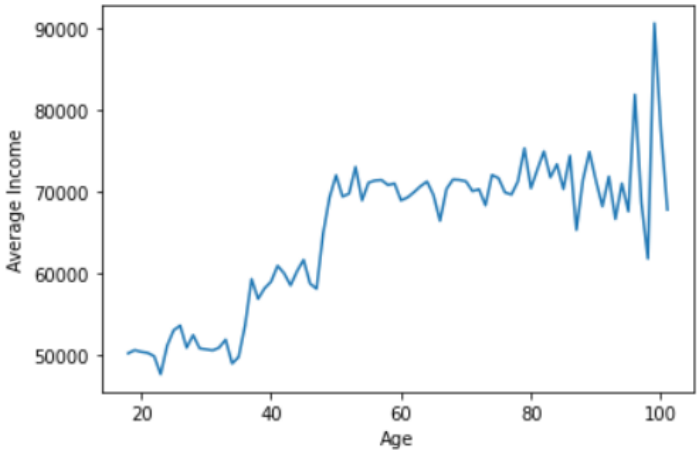

Figure 7: Average income by age

but it does suggest to me that the sample data generated is not a perfect real world example.

3.2.3 Transcript Data

I first checked if there are any null values in the transcript data set and none of the 306534 rows contained any null values in any of it's columns. I then counted that there were 17000 distinct people in the data. I confirmed that these were the same people that appeared in the profile data. I then found that the mean number of occurrences of each person in the transcript data set was over 18.

The time column showed that the time span of the data provided is 714 hours. This translates into nearly 30 days. At first it appeared the time information was highly unreliable because of huge spikes in the data. But When I broke it down by event I saw that the spikes were caused by the "offer received" value. This is illustrated in figure 9 on page 14 And I notice the "offer viewed" values peaking around here, as expected, before decaying until the next offer being received. The "offer completed" value has a similar behaviour to the "offer viewed" but to a lesser extent. The "transaction" data does appear to have a non-uniform pattern and is perhaps influenced

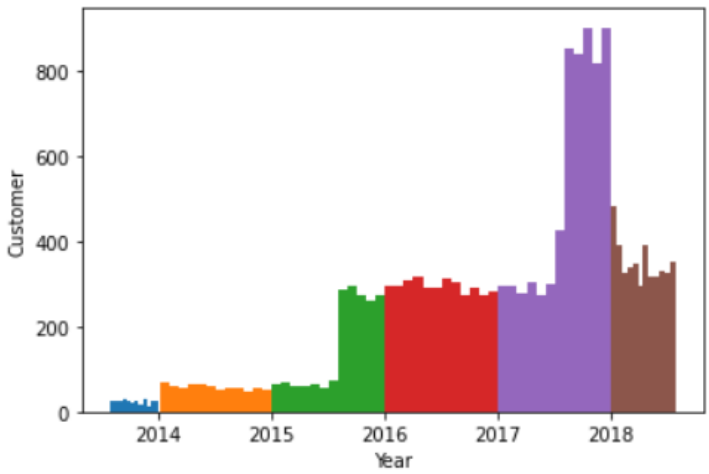

Figure 8: Date joined

by the offers but nothing is clear from this initial plot.

The event break down is shown in figure 10 on page 14. As expected the offer received value is greater than offer viewed and that in turn is greater than offer completed. The transactions make up over 45% of the events.

I had to parse out the value column to be able to be able to analyse the data there because the column contained it's information within nested dictionaries. I first decided to take a look at the dollar amount of all the transactions. I found that the vast majority of transactions were less than $50 but the data was massively skewed by a few huge transactions of a few hundred dollars.

I considered cutting off the transactions that were in the hundreds or thousands of dollars but I could not think of a solid enough reason to justify this. It is very possible that a customer has made a huge purchase like this for their family, friends, or colleagues. And while an offer of $5 might not seem like an incentive for such a big purchase, it is possible they simple notification or good gesture of an offer was enough for the customer to decide to go to Starbucks that day instead of another chain. For this reason I intend to keep these values in.

Perhaps more significantly there are a large number of small transactions.

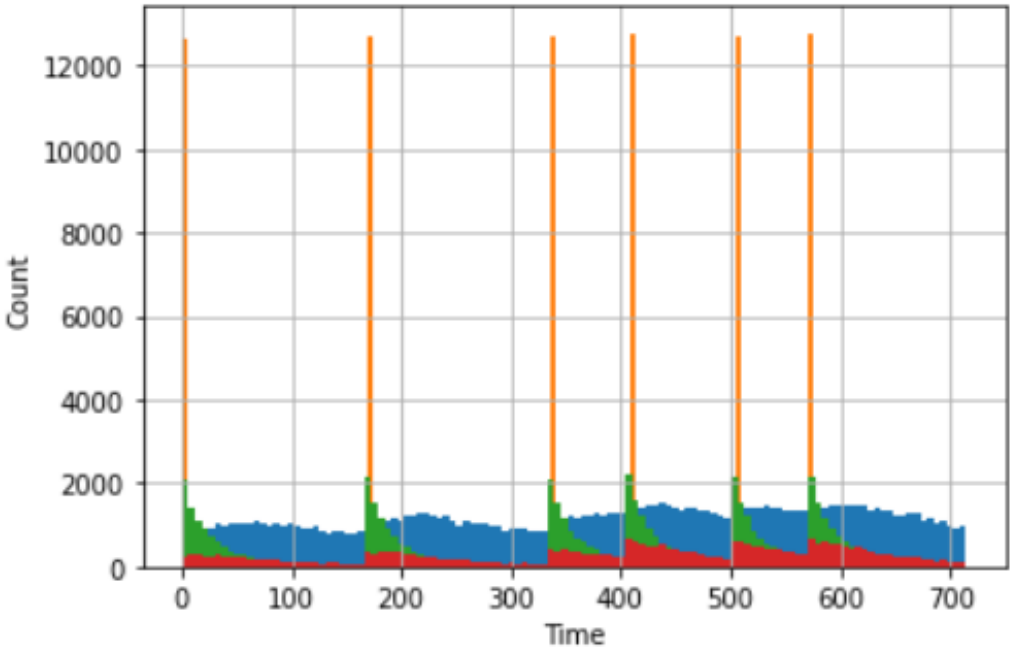

Figure 9: Events by time

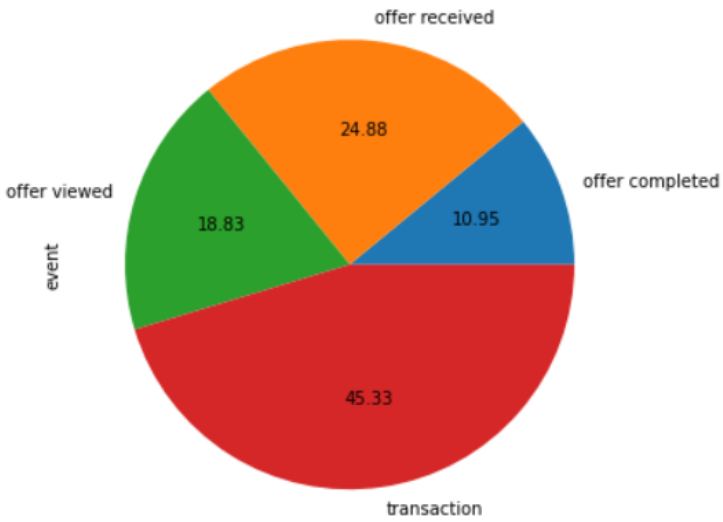

Figure 10: Event ratio

I could not tell what is causing the small transactions. They could be credit card fees that are processed a while after a valid transaction. They could be a small purchase such as a packet of sugar. They could also simply be bad data that should not be in the data set. Some Starbucks shops do have items available for less than 50cents. But I do find purchases of less than this value suspicious. When I added up the maximum total of purchases for less than 25 cents any customer made I got $1.06. This is comfortably less than the lowest difficulty of any discount or BOGO offer of $5. So I don't think they will have a significant impact of misclassifying offers if indeed these transactions are erroneous. But if these transactions were not genuine transactions then they could have the unwanted affect of misclassifying informational offers. If they are delayed credit card fees or bad data that occur after a informational offer has been viewed then that informational offer will be classed as positive. For those reasons I intend to filter out these transactions from the data preparation phase. I will probably select a threshold of around 25cents. I feel this is a safe clause to enter because 25 cents is such a low figure that Starbucks probably would not deem that a successful offer even if they were authentic transactions.

As can been seen in figure 11 on page 15 the 10 offer types first seen in the portfolio data set are sent out to customers in equal measure. Although I can see that the offers are not viewed and completed in equal measure.

Figure 11: Offer ratios by event

3.2.4 Transcript and Portfolio Data

I decided to join the transcript and portfolio data together to see if could see any correlations between certain fields of interest.

I found the number of channels used had a high correlation (0.85) with the likelihood of an offer being viewed. The p-value of 0.0017 is less than 0.05 so it seems statistical significant. The number of channels had a much weaker, but still a high correlation (0.56) with an offer being completed.

The duration very weakly correlated to the percentage the offer has been viewed. And in fact is slightly negatively correlated. I would have expected this to be a stronger positive relationship. Examining some of the values in the table does give me some ideas why this is the case. The discount offer with id=0b1e1539f2cc45b7b9fa7c272da2e1d7 has the lowest view rate even though it was available for the joint longest time. I can see the in the channel's column it was only sent via 2 channels. It was not sent via mobile or social. It is possible that mobile and social channels are of very high importance in terms of getting views. The correlation between duration and offer completed is also weak, but at least it is positive which makes sense. So while a longer duration doesn't appear to have a strong affect on the likelihood of somebody viewing the offer if certain channels are not used, it does seem to have some impact on the chances of them completing the offer. It is important to remember that an offer could be completed without the offer being viewed. The difficulty is moderately negatively correlated (-0.49) with an offer being completed as expected.

The channel column seemed to have the biggest influence on offer viewed. So I will find which channel has most impact on offer viewed and offer completed. The social channel is hugely correlated (0.96) with an offer being viewed. This also had a p-value of close to zero indicating it is statistically significant. The mobile channel had a strong correlation (0.61) with an offer viewed while the email channel actually had a low correlation (-0.27) with an offer being viewed. But neither of these correlations appear statistically significant. There is no point in trying to get the relationship between email and offer viewed as every single offer has email in it. Interestingly there is no strong correlation (0.31) with the social channel and an offer being completed despite the very high relationship between social and viewed percentage. In fact, surprisingly to me, the email channel had a higher correlation (0.43) than the social channel with an offer being completed.

I decided to check the correlation between an offer being viewed and an offer being completed because I now guessed that it would not be that high.

I obtained a modest correlation of 0.42.

As we see above they are positively correlated. But not considered a strong correlation. And it is not statistically significant. This tallies with the narrative that has been building through the previous correlations. Higher rates of views does not automatically covert to higher rates of completion. It is possible that while users are far more likely to view an offer if it pops up on their social media, they are conditioned to ignore adverts like that. While a more targeted offer straight to their mobile or email is more likely to trigger a response once they view it.

3.3 Algorithms and Techniques

A neural network will be used to process the data and maximise the efficiency of the offers. I will use a linear neural network from PyTorch. Neural networks have proved successful in areas such as handwriting recognition and speech recognition, however a neural network classifier can under perform other classifier models depending on the complexity of the issue.[3] I intend to find out if a linear neural network can outperform more classic classification models with this data. It is quite possible that the structure of this problem and it's data does not perform optimally using a linear neural network.

3.4 Benchmark Model

My benchmark model will be a machine learning algorithm from the sklearn library. I want to compare an optimal sklearn ML model to the PyTorch linear neural network. I will run tests on various sklearn algorithms to obtain the optimal one for this classification problem. I will then optimise the chosen model through a few refinement iterations. Once I have built the best version of my benchmark model I will compare it the neural network which will also be optimised in a similar way. I will then compare the ability of the two models to correctly predict the classes.

3.5 Evaluation Metrics

I will use a combination of accuracy, precision and recall to evaluate the effectiveness of the models. Depending on the balance of the classes I will decide on which metric is a best measurement for this problem. If the classes are imbalanced then a combination of precision and recall would prove to be a better way of evaluating than accuracy. This can be done by measuring the F1-score which uses a combination of precision and recall.

Often times even with a balanced class it is better to optimise recall or precision than to optimise accuracy. False positives and false negatives can have a very high importance in classification problems such as cancer screening or fraud detection. And whilst this problem does not seem as critical as those problems it is likely that Starbucks would want to optimise the true positives of the successful offers as opposed to minimising false negatives. Although I will not neglect the false negatives as that could have negative financial connotations such as wasted advertising or annoying customers such that they do not want to remain a Starbucks customer. So for those reasons I will try to optimise recall while also keeping the overall accuracy and F1-score high. I believe an F1-score of around 70% is sufficient.

4 Methodology

4.1 Data Preprocessing

4.1.1 Filtering

The first step of the data preprocessing was removing the profile data that contained mostly null data. Before I did that I wanted to make sure I wasn't removing too much of the overall data. If I was to find I was removing too much then I would have to think of an alternative strategy such as populating the empty values with dummy data. This was something I wanted to avoid because due to the fact that age, income and gender were all missing for these values any estimates would be very crude. I would only have the date they

became a member as a guideline for estimations. I already found that 12.8% of the customers contained empty values, but I wanted to find out how much of the overall transcript data was associated with these customers. I joined the transcript and profile data together in order to count this percentage. I found that 11.0% of the total events were associated with these customers and only 10.8% of transactions. So thankfully these null-customers were less prolific spenders than the rest of the customers. I would be losing slightly less data than I expected by removing these values, so I decided to delete them.

I also deleted was the aforementioned 'email' channel. All offers contain 'email' so therefore it wont be informative. Removing it makes things clearer and it is also good to cut down on features that don't contain useful information before building our model.

The only other data I wanted to delete was the transactions for less than 25 cents for reasons explained above regarding informational offers. In order to do this I needed to unpack the value column in the transcript data. This was done using the pandas pop and from_records functions. I removed values and later on I checked what this decision had on my classification of the informational offers. I found just 53 offers moved from successful to successful. This is about 0.34% of all the informational offers. I was satisfied that I was not changing the classes too much in case my reasoning for filtering out the small transactions was misguided. I believed it will make a small improvement to the algorithm. I also confirmed that the classification of BOGO and discount offers didn't change. The logic for classifying these offers do not directly rely on transactions, so I did not expect this filtering to make any change to their classifications.

4.1.2 Classification

The next step of deciding the actual classes was perhaps the most complicated of the entire project. I explained in the Problem Statement section how I would decide the classes and illustrated it in figures 1 and 2 on page 5. I first added a column showing the expiry of each offer. This was computed from the time the offer was sent and the offer duration. All offers must be completed before the expiry to be deemed a successful offer.

The reason I found this part of the project so complex was because of the possibility of be overlapping offers. I wanted to be sure to identify edge cases. It required a series of forwards fills, backward fills and shifts.

I found examples of overlapping offers so it was important to associate all transactions with the most recent viewed offer. If a user received two offers and only viewed the first one, then all transactions will be assumed to be under the influence of the first one. So I filled forward all transactions with the offer_id of the viewed offers.

I also filled forward the offer viewed events with the expiry time from the offer received events. Then I filled forward the transaction and offer completed expiry values from these offer viewed events. By doing this in two steps I avoided giving the transaction and offer completed events an expiry value in a situation where the offer was not viewed.

I decided to do a spot check on an edge case I found to see how my logic was panning out. I could see in 12 on page 20 that it was working as expected. I could verify that the transaction at time 414 is correctly associated with the offer received at 336 and not the more recent one received at 408. This is because the the more recent one was never viewed. This offer happens to be a informational offer so I will end up flagging this as positive class since it occurred before it's expiry time. This case gets more interesting still because the transaction at time 414 also triggered an offer completed at 414. This is for the more recent offer that was not viewed. So this will not be classed as an effective offer. I will eventually be checking if offer completed events occurred before their expiry times and since this event has a null expiry time I expect to class this in the negative class.

person	event	time	offer_id	amount	reward	channels	difficulty	duration	offer_type	expiry
0009655768c64bdeb2e877511632db8f	offer received	336	3f207df678b143eea3cee63160fa8bed	NaN	0.0	[web, mobile]	0.0	4.0	informational	432.0
0009655768c64bdeb2e877511632db8f	offer viewed	372	3f207df678b143eea3cee63160fa8bed	NaN	0.0	[web, mobile]	0.0	4.0	informational	432.0
0009655768c64bdeb2e877511632db8f	offer received	408	f19421c1d4aa40978ebb69ca19b0e20d	NaN	5.0	[web, mobile, social]	5.0	5.0	bogo	528.0
0009655768c64bdeb2e877511632db8f	transaction	414	3f207df678b143eea3cee63160fa8bed	8.57	0.0	[web, mobile]	0.0	4.0	informational	432.0
0009655768c64bdeb2e877511632db8f	offer completed	414	f19421c1d4aa40978ebb69ca19b0e20d	NaN	5.0	[web, mobile, social]	5.0	5.0	bogo	NaN

Figure 12: Edge case spot check

Transactions that have an offer_id means that a customer has viewed an offer prior to making a purchase. Informational offers with a transaction with a time less than expiry can now be classed as a successful event. It will have an expiry value associated with the last viewed offer, so it is safe to make this determination. Discount and BOGO offers that have been completed at a time less than it's expiry time can also now be classes as a successful event. It will have an expiry value associated with the last viewed offer. So I don't need to explicitly check again that it has been viewed prior to completion.

It made it easier for me to class the data if I split the data up between informational and non-informational offers. I found it easier to follow even though it was not strictly necessary.

4.1.3 Classifying Informational Offers

I can now set a transaction to be a successful offer if the time it occurs is less than the expiry time. I added a new column called successful_offer and populated with 1 for successful offers and 0 for unsuccessful offers.

Now I have all the transactions that correspond to a successful offer. So I want to backfill the successful_offer column to the viewed events first. And then select only the received and viewed and use the shift function to populate the offer received event's succesful_offer column. The reason I do it in two steps is to avoid certain edge cases. Where the same user gets the same offer twice. And if only the second one is successful, I dont want all previous ones classed as successful, just the latter one.

An example of the case I will be avoiding is this (same offer type and same customer all occurring within a small time frame) with a successful transaction

1. Received
2. Received
3. Viewed
4. Received

5. Transaction = True

I want the 2nd offer received to be marked as successful and all other offers received to remain unmarked. If I were to backfill all at once, then all received offers will be classed as successful. So I first backfill transactions to viewed so now the viewed offer is marked as successful. Now I must use the shift function to mark just the one previous received offer to the viewed offer as a backfill would mark the first two received offers as successful. If there are no further transactions or offers viewed then I will be filling in the unmarked successful_offer values with a 0.

1. Received = False
2. Received = True
3. Viewed = True
4. Received = False
5. Transaction = True

I then just select values from the data set where the event is an offer received. I have now successfully classed all informational offers and this will be part of my input to the machine learning algorithms.

4.1.4 Classifying Discount and BOGO Offers

I can also now set a completed offer to be a successful offer if the time it occurs is less than the expiry time. Similarly to the informational offers I added a new column called successful_offer.

It is possible that a single transaction can complete two offers at the same time. I can see in the edge case in figure 13 on page 23 that both the discount and BOGO offers that were received were viewed prior to the transaction that completed them. Neither offer had expired so they both got marked as successful. I do not think it makes sense in this instance to infer that it was the latter offer that influenced the customer more than the first. The first offer gave the customer 10 days to spent 20 dollars. Perhaps the customer

was always planning on activating that offer within the 10 day period. Or perhaps it was the combination of the two offers that persuaded the customer to go back to start Starbucks and purchase something to complete the pair of offers. It is one of many edge cases in this data. And a different data analyst might make a different call. But I will leave both as successful offers.

person	event	time	offer_id	amount	offer_type	expiry	successful_offer
0011e0d4e6b944f998e987f904e8c1e5	offer received	408	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN	discount	648.0	NaN
0011e0d4e6b944f998e987f904e8c1e5	offer viewed	432	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN	discount	648.0	NaN
0011e0d4e6b944f998e987f904e8c1e5	offer received	504	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	bogo	672.0	NaN
0011e0d4e6b944f998e987f904e8c1e5	offer viewed	516	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	bogo	672.0	NaN
0011e0d4e6b944f998e987f904e8c1e5	transaction	576	9b98b8c7a33c4b65b9aebfe6a799e6d9	22.05	bogo	672.0	NaN
0011e0d4e6b944f998e987f904e8c1e5	offer completed	576	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN	discount	648.0	1.0
0011e0d4e6b944f998e987f904e8c1e5	offer completed	576	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN	bogo	672.0	1.0

Figure 13: Edge case spot check

Now I will use similar steps to classify these offers as I did for the informational offers. Using a combination of fill and shift functions I associate the completed offer with the correct offer received.

1. Received
2. Received
3. Viewed
4. Received
5. Transaction
6. Completed = True

The flow above results in the values below with only the 2nd received offer correctly identified as successful.

1. Received = False
2. Received = True
3. Viewed = True
4. Received = False

5. Transaction
6. Completed = True

I then select just the events where an offer was received and join the data onto the informational data from the section above.

4.1.5 The Classes

After filtering just on offer received I counted the number of postive and negative classes. As can be seen in figure 14 on page 24 the classes are not perfectly balanced but not too skewed either.

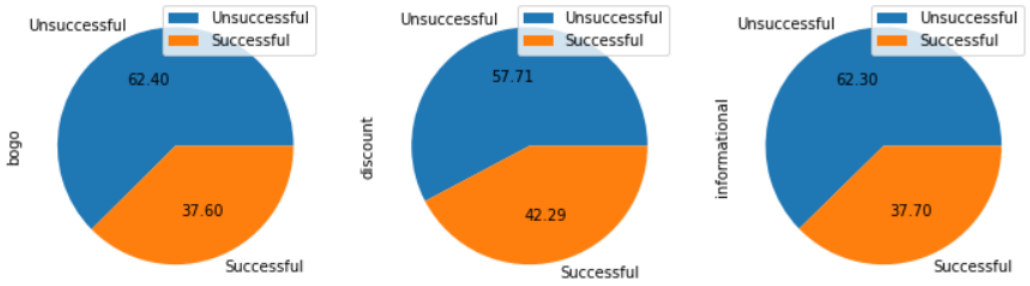

Figure 14: Classes by offer type

4.1.6 Data Preparation

All the categorical data must be converted to binary data for the machine learning algorithms. The channel column contained nested lists so I used the pandas explode function and assigned each value to be 1. I then pivoted the data and joined it back onto the portfolio table. These steps are similar to how one-hot encoding works. The other categorical fields were more straightforward and I was able to directly use pandas one_hot function to convert the data to binary columns.

I also had to convert the became_member_on field to a number. I finally renamed some field names and deleted unneeded fields. The data was now prepared for the implementation of the algorithms.

4.2 Implementation

4.2.1 Benchmark Model

First I will build and refine my benchmark model. I set my features to be reward, difficulty, duration, mobile, social, web, bogo, discount, informational, age, became_member_on, income, female, male and other. I set the label to be successful_offer.

I will use a quantile transformer from sklearn to scale all my features. It is robust at handling outliers and skewed data. I will apply this transform within a pipeline while splitting my data up between training and testing data to avoid the risk of data leakage. I decided on a split of 85:15 for the training and testing split. It is within the acceptable range for splitting of the data. I also wanted to keep the same amount of data for testing in both my benchmark model and neural network model to keep things controlled. (In the neural network model I will use a split of train= 70%, validation= 15%, test= 15%)

I chose a selection of common sklearn machine learning algorithms to run some initial tests on to see how each one performed. I tested the sklearn models DecisionTreeClassifier, AdaBoostClassifier, RandomForestClassifier, GaussianNB, SVC using my data set. The RandomForestClassifier comfortably outperformed the other models in both the accuracy (0.6976) and successful F1-score (0.6352). The confusion matrix can be see in figure 15 on page 26. I decided this will be the sklearn algorithm I will try to optimise and use as my benchmark model against the neural network. The weighted F1-score of this basic model is 0.6958. I would like to keep this figure close to that number while optimising the positive recall value.

4.2.2 Benchmark Refinement

The first approach I took to refining my model was to explore feature reduction. It is possible that too many features are causing my model to overfit the training data and is therefore weaker at predicting the classes within the testing data. First I checked to see importance of each feature in my Ran-

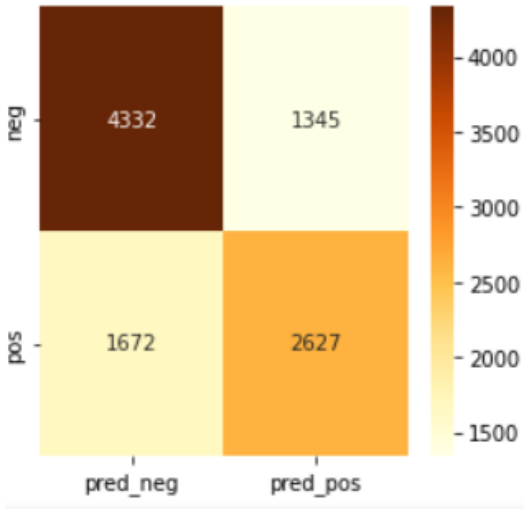

Figure 15: Confusion matrix for RandomForestClassifier with default parameters

domForestClassifier model as can be seen in figure 16 on page 27. I see that age, income, and became_member_on are the most important features. I also see social plays a relatively significant part. This tallies with what I found when doing some correlations in the Data Exploration section.

I wondered if this data set would perform better if I performed principal component analysis on it. I obtained the eigenvalues and plotted them on a scree plot as can be seen in the first image in figure 17 on page 27. A common method for deciding how many principal components to use is use all the components before an elbow in a scree plot.[4] These first 4 components represents 0.78 of the variance. This is illustrated in the second image in figure 17.

I then used the same RandomForestClassifier algorithm from earlier using just these 4 components to see if they were better at predicting the classes. I found that both accuracy (0.6779) and the f1-score (0.6155) were worse off at predicting the data than before. I decided against using PCA on this data going forward.

Next I decided to try to find the parameters for my classifier that would

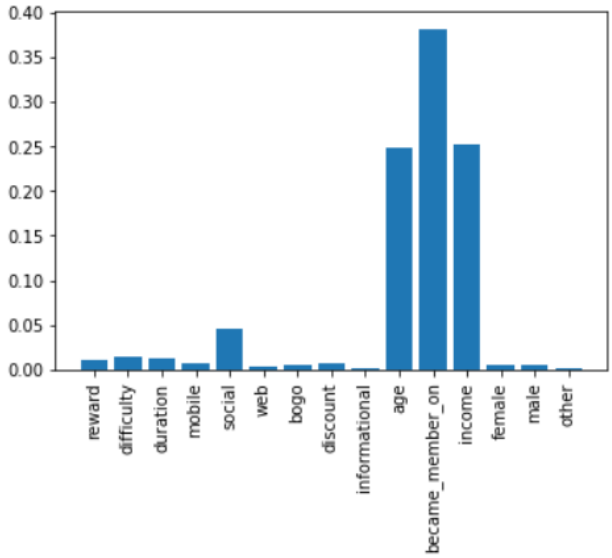

Figure 16: Feature importance

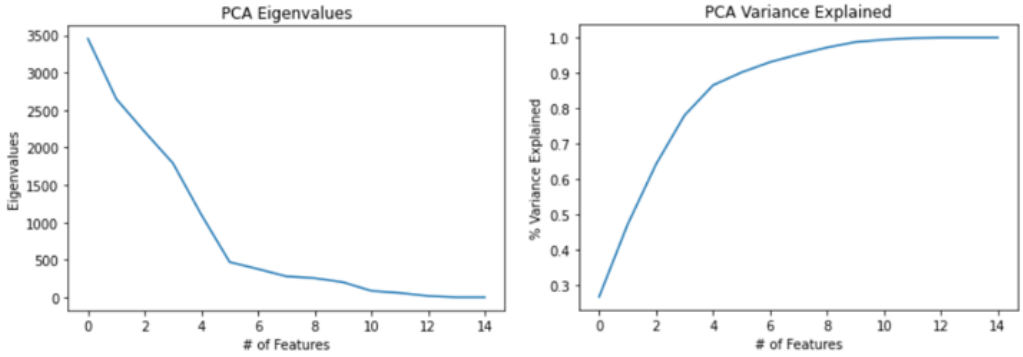

Figure 17: PCA components

optimise it's classification ability. I used sklearn's GridSearchCV function to do this. The RandomForestClassifier is an ideal candidate to perform GridSearchCV on because it has many hyper-parameters that can be tuned. It can be quite computationally expensive to check the performance of all parameters. I selected as many as I deemed reasonable and ran it using a StratifiedKFold=10. I decided to tune it for maximum recall. I discovered that the best parameters for recall were very similar to the default parameters and not much improvement was to be found.

So far my strategies for refinement were not bearing fruit so I decided to go back to the data set and try and extract additional features. I thought using information prior to an offer being sent/received would prove to be beneficial. I calculated the total spend of each customer by the time an offer was received. I did this by using the pandas cumsum function and then forward filling to each offer received. I also counted the total number of individual transactions each customer made prior to an offer being received. Then I also counted the total number of previous offers they received, viewed and completed prior to each additional offer being received. This was done by one-hot encoding the events and then doing the same cumsum and forward fill as before.

I then ran this new data set through the default RandomForestClassifier and finally I did begin to see some improvement in my model. Both the accuracy (0.7053) and the f1-score (0.6373) and increased slightly. I checked the feature importance of this new data set to see if my new features were prominent, and they did appear to play an important part as can be seen in 18 on page 29.

But I was still wary of overfitting the data and wanted to try some dimension reduction tactics again. I tried PCA on this new data set and again I saw a degradation in it's ability to classify. I decided to build a correlation matrix of new features in figure 19 on page 29. Completed, received and viewed had the highest total correlations when they were summed across the grid. I felt that I should keep received and eliminate the completed and viewed as received had the highest feature importance of the three.

I then ran it through the same algorithm as before with the default parameters and I in fact saw an improvement in both accuracy (0.7545) and

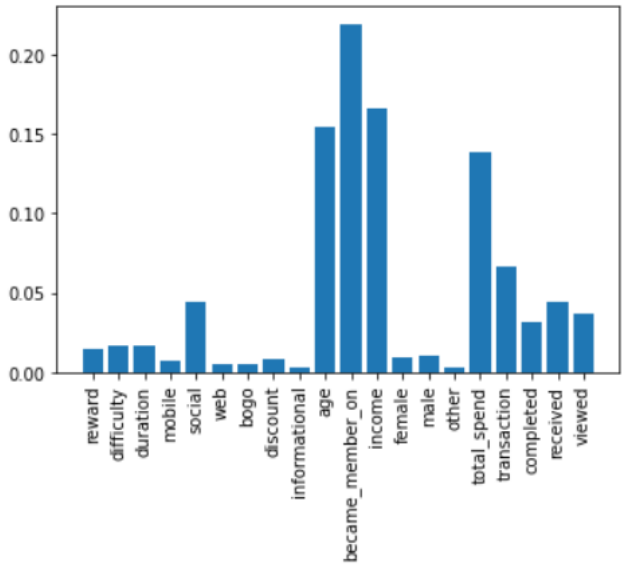

Figure 18: Feature importance

	total_spend	transaction	completed	received	viewed
total_spend	1.00	0.52	0.59	0.38	0.41
transaction	0.52	1.00	0.63	0.60	0.59
completed	0.59	0.63	1.00	0.61	0.63
received	0.38	0.60	0.61	1.00	0.86
viewed	0.41	0.59	0.63	0.86	1.00

Figure 19: Correlation matrix of new features

the f1-score (0.6415). I suspect it was a case of overfitting the data when both completed and viewed were included.

My final step of refinement was to take this final data set, with the three new features of total_spend, transactions and received, apply the GridSearchCV method as before. I found that recall was optimised when just two of the default parameters where changed. I set max_depth = 20 and n_estimators = 300. The recall score was 0.6139 and the overall accuracy was a respectable 0.7151. The confusion matrix for this refined model can be seen in 20 on page 30

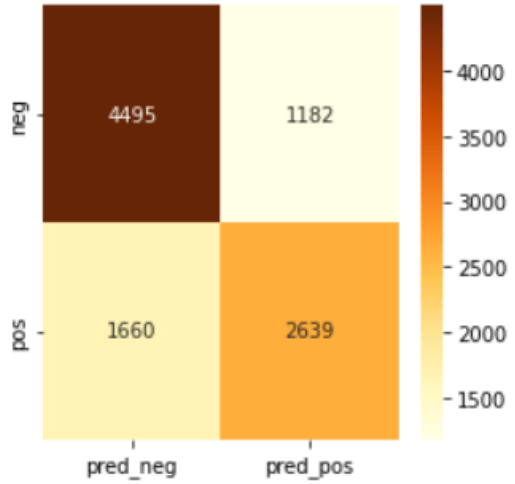

Figure 20: Confusion matrix for refined RandomForestClassifier

4.2.3 Neural Network Model

I decided to use a linear neural network for this model. The data set I would use is the same data set (ie. including the new features) I used in the optimal version of my refined benchmark model. And I scaled my data using the same QuantileScaler as in the benchmark model. Also, similarly to the benchmark model, I set apart 15% of the data as testing data. The split used was train= 70%, validation= 15%, test= 15%. I had them converted to PyTorch tensors. I confirmed that all the data tensors was shuffled correctly and had a similar positive and negative class split.

I built a linear network and used 256 hidden layers. I also used the PyTorch Adam optimiser with a learning rate and weight decay of 0.0001. I used a 0.1 dropout factor on each iteration. I also weighted the cross entropy loss by the number of positive and negative classes in the training set. I then ran the model for 200 epochs to see how it compared to the benchmark model. I saw that the validation model was not improving after about 60 iterations. So to save time I decided to just run it for 60 epochs. I avoided the risk of overfitting the training data due to too many epochs by only selecting a model after each iteration if it had a better performance on the validation data and minimised its loss. The accuracy achieved was 0.7148. This is nearly identical my benchmark model. However, the benchmark model was optimised for recall and this model had a positive recall value of just 0.5398. The confusion matrix for this refined model can be seen in 21 on page 31

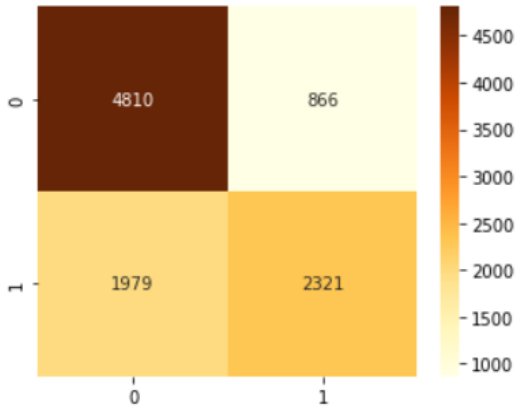

Figure 21: Confusion matrix for initial neural network model

4.2.4 Neural Network Refinement

I now wanted to refine my model by matching the positive recall score achieved in my benchmark model. I wanted to see if the linear neural network could match that score while also maintaining as good an accuracy score as the benchmark model. To do this I gave more weight to the cross entropy for the positive class. I tested a few different weights (I settled on 1.35) until I got a recall score I was satisfied with. The recall score of this model was

0.6326 and the accuracy remained high at 0.7210. The confusion matrix for this refined model can be seen in 22 on page 32

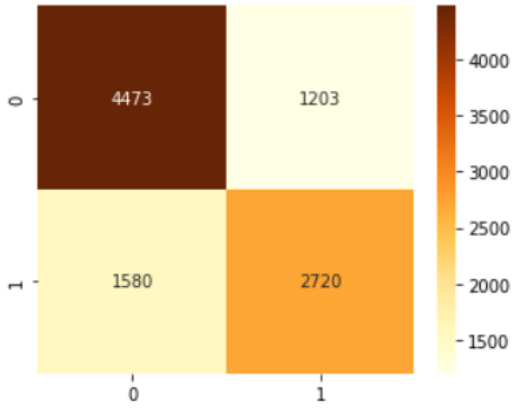

Figure 22: Confusion matrix for refined neural network model

5 Results

5.1 Model Evaluation

The models were both tested on unseen data of the same size (15% of the entire data set) The models in both cases were evaluated by sklearn's metric functions with a focus on trying to achieve a good recall score without too much detriment to the accuracy and F1-scores. I felt the recall of the positive class was the most relevant metric to this problem. Recall is a measurement of how many true positive cases where correctly identified and classes as positive. I think this is the best metric because Starbucks would be very interested in sending adverts to customers who will respond positively. I think they would be less concerned about sending adverts to customers who ignore them even though it could be a waste of time and money and possibly irritate their customers.

5.2 Justification

I found that the neural network model edged out the benchmark model slightly. I was able to achieve a higher recall (0.6326 ¿ 0.6139) score with the neural network while also maintaining a higher accuracy (0.7210 ¿ 0.7151) and F1-score (0.7191 ¿ 0.7125). Even though the scores are extremely close I feel this was a big success for the neural network as I was not confident that the neural network would be able to match the sklearn classification models.

6 Conclusion

The data sets provided many possible avenues to explore and I could easily go back and start this project with a completely new objective and angle. The openness of this project meant that I did not have settled objective until I had finished the data exploration phase in detail. I was also interested and surprised to see how well the neural network did at classifying the data. I had expected the benchmark model's scores to be equal or better than the neural network model's.

6.1 Potential Improvement

As I was doing the project other potential avenues for exploration and improvement would materialise. The new features I added based on past events were particularly successful at improving the model so I believe this could be extended even further. Another possible feature that could be added is the average time lag between a customer receiving and viewing their past offers.

I also think it would be interesting to add another class to the problem. In this project I just had successful and unsuccessful and I was primarily focused on trying to correctly identify successful classes (recall). But a new class could be a 'very unsuccessful' offer. That is one where a user completed an offer and got a discount without seeing the offer and therefore leading Starbucks to lose money by offering a discount when they did not need to and losing out on additional revenue. I believe Starbucks would be keen to

limit those occurrences.

6.2 Final Thoughts

I found this project challenging although very enjoyable. The Udacity tutorials and tutors provided great support throughout. This project was ideal for extending my learning and understanding in the data science and machine learning fields and has encouraged me to keep exploring this space.

References

[1] Kiho Lim Jin-A Choi. Identifying Machine Learning Techniques for Classification of Target Advertising. *ICT Express*, pages 175–180, 2020.

[2] Chen Y., Kapralov M., Canny J., and Pavlov D.Y. Factor Modelling for Advertisement Targeting. *Advances in Neural Information Processing System*, pages 324–332, 2009.

[3] Joko Siswantoro, Anton Satria Prabuwono, Azizi Abdullah, and Bahari Idrus. A linear model based on Kalman filter for improving neural network classification performance. *Expert Systems with Applications*, pages 112–122, 2016.

[4] Mu Zhu and Ali Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, pages 918–930, 2006.