Udacity Machine Learning Nanodegree

Capstone Project Report

Sanjeev Kumar

13th April 2021

Contents

Definition

Project overview

This project derives from the direct marketing system which Starbucks uses to keep in touch with its customers.

Aiming to incentivize and reward the customers registered in its platform, Starbucks periodically sends individual messages containing offers related to its products.

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels: e-mail, social media, on the web, or via the Starbucks's app.

Nonetheless, marketing campaigns have associated costs. Hence, to be considered a successful campaign, it must generate profit higher than that initial cost. That means, companies expect to have a return on investment (ROI) as high as possible.

Nonetheless, marketing campaigns have associated costs. Hence, to be considered a successful campaign, it must generate profit higher than that initial cost. That means, companies expect to have a return on investment (ROI) as high as possible.

Sometimes, recurrent consumers deserve some reward so they can feel appreciated and not forgotten. However, some customers keep coming back even if they do not receive offers. To give an example, from a business perspective, if a customer is going to make a 10-dollar purchase without an offer anyway, you would not want to send a "buy 10 dollars, get 2 dollars off" offer, unless this relative short-term loss means a more satisfied customer who will consume more in the future.

Another case is those customers who only buy products when receiving some reward, while other ones are opposed to marketing campaigns and do not want to be contacted at all.

Considering the recent advances of artificial intelligence and the massive amount of data gathered over the years, this is a topic that could be widely improved by intelligent systems owing to the fact that they can analyze a large amount of data and understand patterns sometimes hidden for the human perception.

This is a very interesting subject since proposing better marketing campaigns may benefit not only the companies by raising their profit, but also the customers who receive more relevant offers in accordance with their consumer behavior. Perhaps, this win-win situation is the key to maintain the economy growing while people can afford better services and products.

The data set used in this project is provided by Udacity and Starbucks as part of the Machine Learning Engineer Nanodegree program. It contains simulated data that mimics customer behavior on the Starbucks rewards mobile app.

The program used to create the data simulates how people make purchasing decisions and how those decisions are influenced by promotional offers.

Each person in the simulation has some hidden traits that influence their purchasing patterns and are associated with their observable traits. People produce various events, including receiving offers, opening offers, and making purchases.

Problem statement

Starbucks, as well as any other company, invests money in marketing campaigns expecting to have a profit higher than the assumed before. So, identifying the most relevant offer to the correct customers is crucial for a successful campaign.

However, some targeted customers do not even see the offer sent to them, which may be a problem with the channel chosen. Other ones do not buy anything, despite seeing the offer, what might be a problem with the offer type sent, or maybe that is not a customer to be considered as a target.

There are other cases where the customers identify themselves with the offer, which leads them to try new products or spend more money than usual. Those are the situations to be identified and pursued.

The problem this project proposes to solve is finding the most appropriate offer for each one of the customers, which means finding the offer that is more likely to lead the customer to buy Starbucks products.

If a customer does not see an offer, it is not an appropriate one. If he or she sees the offer but does not complete it, it is not appropriate as well, since it

did not lead the consumer to buy products. Similarly, if the customer buys some products, completes an offer, and receives a reward before visualizing that offer, it is not considered effective because the customer was not under the influence of that offer when decided to make a purchase.

As there are many permutation and combination of the problem, I will be focusing on the assumption that: the company may not be too concerned about accuracy and perhaps expects to experience a fairly large number of incorrect predictions. The initial intuition is that Starbucks would want to minimize lost sales – it is worse for the company to lose out on a potential sale by not providing an offer than to hand out an offer that is not used. Perhaps the revenues from a few sales more than make up for the costs of many offers given to customers. With this primary motivation, Starbucks may place more weight on recall, to reduce the number of false negatives, meaning if the model predicts the customer will not make a purchase after being shown an offer, the customer does not make a purchase. Another way to frame it is that the company would like to correctly identify as many of the people who would buy as possible and is willing to overshoot on identifying customers as potential buyers. So Starbucks may be willing to tolerate the model scoring lower on precision, meaning it would allow for the model to produce more false positives.

Main model to be used will be a deep feedforward neural network having three or more hidden layers and a convolutional neural network to avoid overfitting through regularization. A benchmark model I could use will likely be a model outside the category of neural networks, relatively simple, and one that has been widely used in academia and industry in the past. Another good candidate may be a linear model such as a support vector machine with a linear kernel.

The final result of this project is a direct marketing system that, given a customer, is able to predict the likelihood of each offer be completed. The theoretical workflow for approaching the solution stated includes several machine learning techniques, following the guideline sections below.

<u>Data loading and exploration</u>
Load files and present some data visualization in order to understand the distribution and characteristics of the data, and possibly identify inconsistencies.

<u>Data cleaning and pre-processing</u>
Having analyzed the data, handle data to fix possible issues found.

<u>Feature engineering and data transformation</u>
Prepare the data to correspond to the problem stated and feed the neural

networks. The transcription records must be structured and labeled as appropriate offer or not.

Splitting the data into training, validation, and testing sets
Prepare three datasets containing distinct registers within each one.
The largest dataset is employed to train the networks, while the validation set, to evaluate the models during the training phase.
The testing set contains data never seen before by the networks, so it will be possible to consider this dataset as being new interactions between Starbucks and customers. By using this dataset, it will be possible to measure the final performance and compare the results of the trained models.

Defining and training a LinearSVC from sklearn library
Training of the benchmark model.

Defining and training a Feed-Forward Neural Network (Binary Classifier)
Training of the proposed model.

Evaluating and comparing model performances
Comparison between the accuracy of both network models to verify each one is more suitable to solve the problem stated.

Presenting predictions for offer sending
Present the resulting predictions, along with discussions on how this system should be employed.

Metrics

To evaluate the performance of the model I may look at accuracy, precision and recall.

The company may not be too concerned about accuracy and perhaps expects to experience a fairly large number of incorrect predictions. The initial intuition is that Starbucks would want to minimize lost sales – it is worse for the company to lose out on a potential sale by not providing an offer than to hand out an offer that is not used. Perhaps the revenues from a few sales more than make up for the costs of many offers given to customers. With this primary motivation, Starbucks may place more weight on recall, to reduce the number of false negatives, meaning if the model predicts the customer will not make a purchase after being shown an offer, the customer does not make a purchase. Another way to frame it is that the company would like to correctly identify as many of the people who would buy as possible and is willing to overshoot on identifying customers as potential buyers. So

Starbucks may be willing to tolerate the model scoring lower on precision, meaning it

would allow for the model to produce more false positives.

I am aiming for:

1. Accuracy: 70 %
2. Recall: 90 %
3. Precision: 40 %

balanced accuracy: it is the percentage of correctly classified records, corrected for unbalanced targets. The formula is

$$bal\_acc = \left(\frac{TruePositive}{Positive} + \frac{TrueNegative}{Negative} \right) / 2$$

Precision/Recall: these two measures focuses on the Positive labels:

$$precision = \frac{TruePositive}{PreictedPositive}, \qquad recall = \frac{TruePositive}{Positive}$$

Analysis

Data Exploration and Exploratory Visualization

This section presents some data visualization in order to understand the distribution and characteristics of the data, and possibly identify inconsistencies.

Data exploration is one of the most important parts of the machine learning workflow because it allows you to notice any initial patterns in data distribution and features that may inform how to proceed with modeling and clustering the data.

Data exploration uses visual exploration to understand what is in a dataset and the characteristics of the data. These characteristics can include size or amount of data, completeness and correctness of the data, and possible relationships amongst data elements.

portfolio

Offer ids and meta data about each offer sent during 30-day test period.
Size: 10 offers x 6 fields

- id (string) - offer id
- offer_type (string) - type of offer ie BOGO, discount, informational

- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings)

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

There are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational:

- In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount.
- In a discount, a user gains a reward equal to a fraction of the amount spent.
- In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend.

Offers can be delivered via multiple channels: email, social media, on the web, via the Starbucks's app.

Every offer has a validity period (duration) before the offer expires. We see that informational offers have a validity period even though these ads are merely providing information about a product. Here, the duration is the assumed period in which the customer is feeling the influence of the offer after receiving the advertisement.

As we can see, offer_type and channels are presented as a categorical values, which must be converted to columns (one hot encoding) before passed into a neural network.

In this portfolio, any offer is sent by email, so this is an informative feature and might be filtered out when feeding the neural networks.

Features reward, difficulty, and duration present values in different ranges. It is a good practice to scale then to the same range than other features.
Apart from that, no issue is noticeable in this dataset.

profile

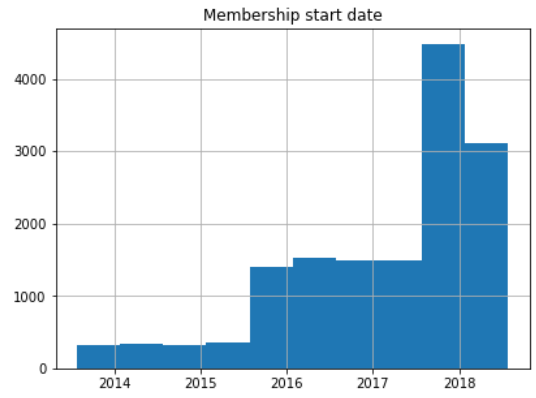
Demographic data for each rewards program users.

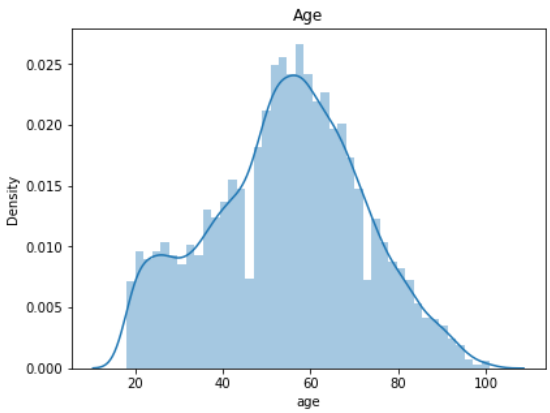
Size: 17000 users x 5 fields

- gender: (categorical) M, F, O, or null
- age: (numeric) missing value encoded as 118
- id: (string/hash)
- became_member_on: (date) format YYYYMMDD
- income: (numeric)

	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN

This dataset presents gender as a categorical feature which should be mapped to columns. There are some missing values in gender feature: in the same rows, income is missing too, and age is always equal to 118. We can suppose that informations about these customer has been lost, so we can safely drop these records: there remain 14,825 records.



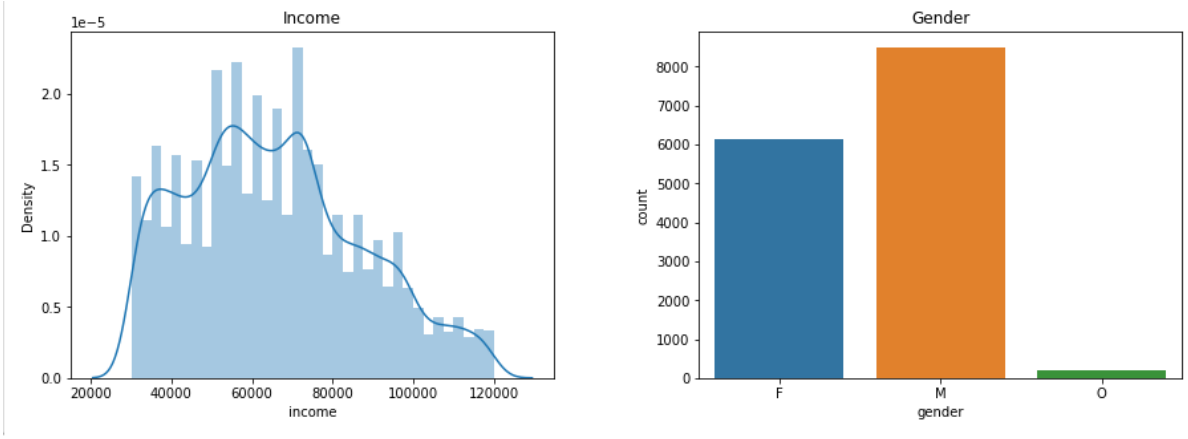



```
Membership start dates range from 2013-07-29 to 2018-07-26.
```

- Analyzing the graphs above, we can see that although the age is normally distributed among the population, there is a local peak around the interval 20

~ 25. Apparently, this deviation is not a problem to be handled beforehand. However, this is a point to be taken into consideration if networks have difficulty to converge. Hence, simply standardization for this feature seems to be good enough, as shown in the graphs below.

- the customers are quite new members, since the memberships are concentrated from the last part of 2017 onwards



- there is a third gender O, with very few records
- These graphs show us that income is not a well distributed feature.

transcript

Event log containing records for transactions, offers received, offers viewed, and offers completed.

Size: 306648 events x 4 fields

- person: (string/hash)
- event: (string) offer received, offer viewed, transaction, offer completed
- value: (dictionary) different values depending on event type
- offer id: (string/hash) not associated with any "transaction"
- amount: (numeric) money spent in "transaction"
- reward: (numeric) money gained from "offer completed"
- time: (numeric) hours after start of test

	person	event	time	amount	offer_id	reward
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	NaN	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN
1	a03223e636434f42ac4c3df47e8bac43	offer received	0	NaN	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN
2	e2127556f4f64592b11af22de27a7932	offer received	0	NaN	2906b810c7d4411798c6938adc9daaa5	NaN
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	0	NaN	fafdcd668e3743c1bb461111dcafc2a4	NaN
4	68617ca6246f4fbc85e91a2a49552598	offer received	0	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN
12650	389bc3fa690240e798340f5a15918d5c	offer viewed	0	NaN	f19421c1d4aa40978ebb69ca19b0e20d	NaN
12651	d1ede868e29245ea91818a903fec04c6	offer viewed	0	NaN	5a8bc65990b245e5a138643cd4eb9837	NaN
12652	102e9454054946fda62242d2e176fdce	offer viewed	0	NaN	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN
12653	02c083884c7d45b39cc68e1314fec56c	offer viewed	0	NaN	ae264e3637204a6fb9bb56bc8210ddfd	NaN
12655	be8a5d1981a2458d90b255ddc7e0d174	offer viewed	0	NaN	5a8bc65990b245e5a138643cd4eb9837	NaN
12654	02c083884c7d45b39cc68e1314fec56c	transaction	0	0.83	NaN	NaN
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	transaction	0	34.56	NaN	NaN
12659	54890f68699049c2a04d415abc25e717	transaction	0	13.23	NaN	NaN
12670	b2f1cd155b864803ad8334cdf13c4bd2	transaction	0	19.51	NaN	NaN
12671	fe97aa22dd3e48c8b143116a8403dd52	transaction	0	18.97	NaN	NaN
12658	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	offer completed	0	NaN	2906b810c7d4411798c6938adc9daaa5	2.0
12672	fe97aa22dd3e48c8b143116a8403dd52	offer completed	0	NaN	fafdcd668e3743c1bb461111dcafc2a4	2.0

In the representative dataset above, we notice that:

- offer received and offer viewed have an associated offer id
- transaction has an amount value that indicates how much the customer spent
- offer completed is associated to a reward value
- every event is informed with person and time
- distinct events can occur to the same person at the same time. So that, we need to check if these statements are always true.

Is there missing information?

	person	event	time	amount	offer_id	reward
offer received	True	True	True	False	True	False
offer viewed	True	True	True	False	True	False
transaction	True	True	True	True	False	False
offer completed	True	True	True	False	True	True

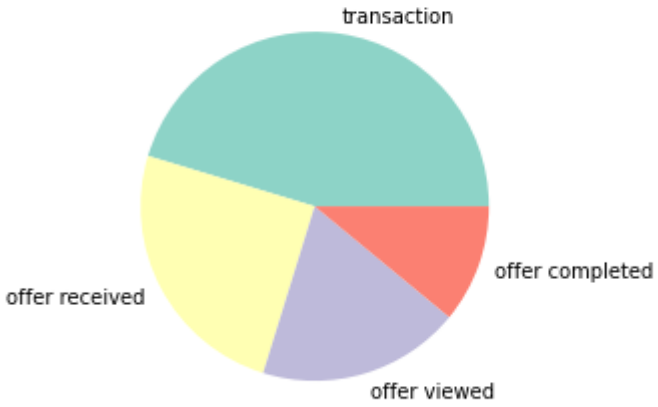
Is there event with extraneous information provided?

	person	event	time	amount	offer_id	reward
offer received	False	False	False	True	False	True
offer viewed	False	False	False	True	False	True
transaction	False	False	False	False	True	True
offer completed	False	False	False	True	False	False

According to these verifications, there is no missing nor extraneous information in this dataset.

Now, we need to verify if there is some inconsistent values in it.

	transaction	offer received	offer viewed	offer completed
event	138953	76277	57725	33579



id	reward	channels	difficulty	duration	offer_type	received	viewed	completed
ae264e3637204a6fb9bb56bc8210ddfd	10	[email, mobile, social]	10	7	bogo	7658	6716	3688.0
4d5c57ea9a6940dd891ad53e9dbe8da0	10	[web, email, mobile, social]	10	5	bogo	7593	7298	3331.0
3f207df678b143eea3cee63160fa8bed	0	[web, email, mobile]	0	4	informational	7617	4144	NaN
9b98b8c7a33c4b65b9aebfe6a799e6d9	5	[web, email, mobile]	5	7	bogo	7677	4171	4354.0
0b1e1539f2cc45b7b9fa7c272da2e1d7	5	[web, email]	20	10	discount	7668	2663	3420.0
2298d6c36e964ae4a3e7e9706d1fb8c2	3	[web, email, mobile, social]	7	7	discount	7646	7337	5156.0
fafdcd668e3743c1bb461111dcafc2a4	2	[web, email, mobile, social]	10	10	discount	7597	7327	5317.0
5a8bc65990b245e5a138643cd4eb9837	0	[email, mobile, social]	0	3	informational	7618	6687	NaN
f19421c1d4aa40978ebb69ca19b0e20d	5	[web, email, mobile, social]	5	5	bogo	7571	7264	4296.0
2906b810c7d4411798c6938adc9daaa5	2	[web, email, mobile]	10	7	discount	7632	4118	4017.0

There is no anomaly detected in the data related to events. Besides that, we note a correlation between views and channels. Not surprisingly, the more channels used to deliver the offer, the better.

Additionally, using social media seems to be the most effective channel. Furthermore, there is some correlation between offer completion and features reward, difficulty, and duration. However, it seems more related to the number of offers received.

Analyze expense

Total

	amount
count	138953.00
mean	12.78
std	30.25
min	0.05
25%	2.78
50%	8.89
75%	18.07
max	1062.28

Count by person

	amount
count	17000.00
mean	8.17
std	5.12
min	0.00
25%	4.00
50%	7.00
75%	11.00
max	36.00

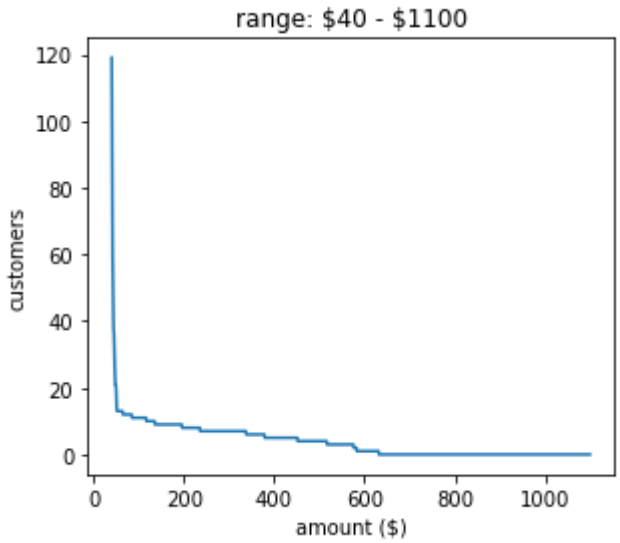
Sum by person

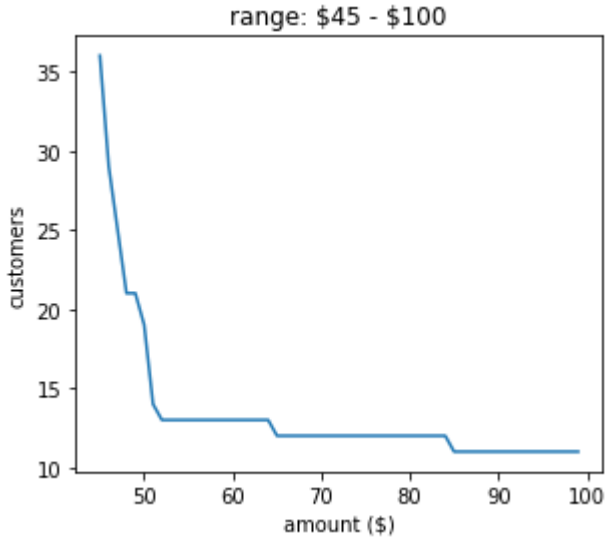
	amount
count	17000.00
mean	104.44
std	125.92
min	0.00
25%	21.82
50%	69.41
75%	148.78
max	1608.69

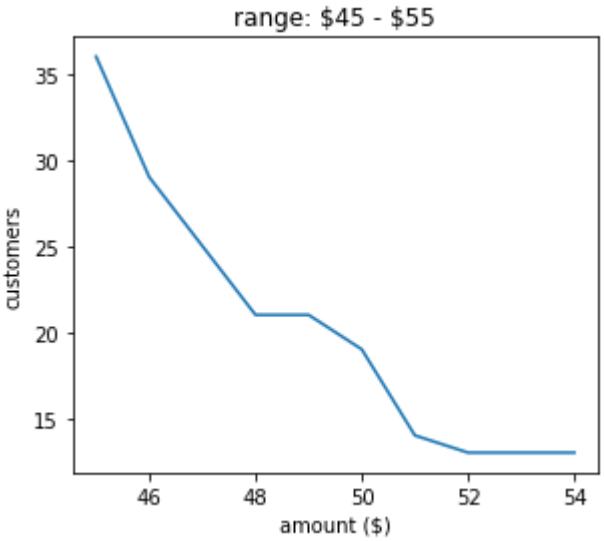
These description tables show us some interesting points.

- There are customers who did not make one single purchase in the period.
- There are anomalous values in the amount feature

The three graphs below show the number of customers who repeat a transaction with a value higher than a certain amount. Around the value 50, that number tends to stabilize.







Number of customers with more than 1 transaction with a value >= $50,00: 19
Number of customers with transactions with a value >= $50,00: 687

Number of customers with expense less than 50 dollars are max. Above graph gives idea on what range and offer should be.
It is quite clear that one purchase with a value of $1,062.28 in this dataset is an anomaly. However, it is especially tricky finding the value that correctly classifies outliers. I tried to define this value by finding the number of times one customer repeats transactions with a similar amount.

The three graphs on the top show the number of customers who repeat a transaction with a value higher than a certain amount. Around the value 50, that number tends to stabilize. By setting the threshold to $50, there are 668 customers with transactions higher than this amount, and only 14 of them repeat a similar transaction.

The three graphs on the bottom present transactions with value clipped to $50.

Algorithms and Techniques

Analysis

From data analysis above we can say that, the company may not be too concerned about accuracy and perhaps expects to experience a fairly large number of incorrect predictions. The initial intuition is that Starbucks would want to minimize lost sales – it is worse for the company to lose out on a potential sale by not providing an offer than to hand out an offer that is not used. Perhaps the revenues from a few sales more than make up for the costs of many offers given to customers.

With this primary motivation, Starbucks may place more weight on recall, to reduce the number of false negatives, meaning if the model predicts the customer will not make a purchase after being shown an offer, the customer does not make a purchase. Another way to frame it is that the company would like to correctly identify as many of the people who would buy as possible and is willing to overshoot on identifying customers as potential buyers.

So Starbucks may be willing to tolerate the model scoring lower on precision, meaning it would allow for the model to produce more false positives.

Aim is to find the offers which have highest rate of acceptance. To achieve this, I will take each kind of offer (bogo, discount and advertisement for a drink) and make separate predictions and find which is best offer for a customer.

The data points that I will be using for the binary classification are quite balanced. Hence, I think accuracy along with precision and recall will be helpful in this case Data which I am going to use is well Balanced. I am interested in finding

1) Total completed offers: 33579
2) Total number of viewed offers: 57725
3) How many viewed the offer and completed it too: 29456
4) How many viewed the offer but didn't complete it: 28269

I will be performing binary classification on this data and since the data points are quite balance the accuracy will be helpful in this case.

```
In [64]: viewed_offers_df[viewed_offers_df['offer_used_label'] == 0].count()
         #29456
         #28269

Out[64]: person              28269
         event               28269
         value               28269
         time                28269
         offer_used_label    28269
         dtype: int64
```

```
In [65]: viewed_offers_df[viewed_offers_df['offer_used_label'] == 1].count()

Out[65]: person              29456
         event               29456
         value               29456
         time                29456
         offer_used_label    29456
         dtype: int64
```

Algorithm

I want to create a binary classification feedforward models having hidden layers in between the input and the output layers. After every hidden layer, an activation function is applied to introduce non-linearity. When you have more than two hidden layers, the model is also called the deep/multilayer feedforward model or multilayer perceptron model (MLP).

Choice is between feed forward and convolutional NN (counters overfitting). But right now, the data set looks quite simple so convolutional NN may not be needed.

Now to achieve, this I plan to use **sklearn's Linear Support Vector Classification(SVC)** and compare the results with **PyTorch's neural network that performs Binary Classification**.

Benchmark

Main model to be used will be a deep feedforward neural network having three or more hidden layers and a convolutional neural network to avoid overfitting through regularization.

A benchmark model I could use will likely be a model outside the category of neural networks, relatively simple, and one that has been widely used in academia and industry in the past.

Another good candidate may be a linear model such as a support vector machine with a linear kernel.

Methodology

Data Preprocessing

In the section Data Exploration and Exploratory Visualization, some issues were identified in the datasets. In Algorithms and Techniques, it was explained how to tackle those problems. This section presents the results obtained for those features.

Portfolio

Although the features in this dataset have been analyzed and no issues found, no transformation is necessary since this project is concerned in determine whether an offer would be completed.

Profile

Index the profile data using id feature. This will help later in joining it with portfolio and transcript to enrich the data.

Resultant dataset

```
profile_id_as_index
```

id	gender	age	became_member_on	income
68be06ca386d4c31939f3a4f0e3dd783	None	118	20170212	NaN
0610b486422d4921ae7d2bf64640c50b	F	55	20170715	112000.0
38fe809add3b4fcf9315a9694bb96ff5	None	118	20180712	NaN
78afa995795e4d85b5d9ceeca43f5fef	F	75	20170509	100000.0
a03223e636434f42ac4c3df47e8bac43	None	118	20170804	NaN
...	...	...	...	...
6d5f3a774f3d4714ab0c092238f3a1d7	F	45	20180604	54000.0
2cb4f97358b841b9a9773a7aa05a9d77	M	61	20180713	72000.0
01d26f638c274aa0b965d24cefe3183f	M	49	20170126	73000.0
9dc1421481194dcd9400aec7c9ae6366	F	83	20160307	50000.0
e4052622e5ba45a8b96b59aba68cf068	F	62	20170722	82000.0

17000 rows × 4 columns

Transcript

Filter out the customer who have completed the offers. This is important as we are interested in customers completing their offers.

Add a new column offer_id. This extracts the value from the key:value paring in the feature called 'value'. As in current form its unusable.

Filter out the customer who have viewed the offers. This is needed as we will be joining it with the customer who have completed the offers.

Finding customer who have viewed and also completed the offers. Returns dataframe with customer who have viewed offers and also completed them. Create valid offers data.

Create a new dataframe and enrich it with more information. Done by joining 'profile_id_as_index' and 'viewed_offer' dataframes.

Bit of cleaning is done. Fill the income values with 0 if it contains null.

To train out model we are going to use only 'age', 'time' and 'offer_used_label'. Drop every other feature from the dataframe.

Final data looks like below. This will be used to create train and test datasets.

```
valid_offers_df
```

	age	income	time	offer_used_label
12650	65	53000.0	0	1
12651	53	52000.0	0	0
12652	69	57000.0	0	1
12653	20	30000.0	0	0
12655	39	51000.0	0	0
...	...	...	...	...
14048	53	100000.0	0	0
14049	118	0.0	0	0
14050	78	57000.0	0	1
14051	64	98000.0	0	1
14052	118	0.0	0	0

1000 rows × 4 columns

Implementation

All the data pre-processing and the model building are developed under the Amazon Web Services infrastructure, specifically the Sagemaker component. This is a cloud Machine Learning platform to develop, tune, version, query and deploy Machine Learning models. In particular we will use the Python API of Sagemaker:

- the data processing component, via SKLearnProcessor
- the hyperparameter tuning with the HyperparameterTuner component
- the model development with the Estimator object
- finally, the model query for result using the Batch Transformation part

General

Since this project aims to define the offers most likely to be completed, precision is the metric more relevant. Precision indicates how many positive labels were correctly classified against those classified as positive but it is a negative case actually.

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

$$\text{precision} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{tp}{tp + fn}$$

Where:

- $\hat{y}$ is the predicted value of the i-th sample and y is the corresponding true value.
- tp (True Positive) is the number of correctly classified sample when the category is *positive*.
- fp (False Positive) is the number of wrongly classified sample when the category is *negative*.
- fn (False Negative) is the number of wrongly classified sample when the category is *positive*.

Model 1 : Linear SVC

The first model I built and trained is the sklearn's Linear SVC. As the final data with the label is quite balanced and Linear SVC is good file for it.

Main model to be used will be a deep feedforward neural network having three

or more hidden layers and a convolutional neural network to avoid overfitting through regularization. A benchmark model I could use will likely be a model outside the category of neural networks, relatively simple, and one that has been widely used in academia and industry in the past. Spo the good candidate is be a linear model such as a support vector machine with a linear kernel.

Created the train.py file and defined the model and called the train method like below:

```
# SageMaker parameters, like the directories for training data and saving models; set automatically
# Do not need to change
parser.add_argument('--output-data-dir', type=str, default=os.environ['SM_OUTPUT_DATA_DIR'])
parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])
parser.add_argument('--data-dir', type=str, default=os.environ['SM_CHANNEL_TRAIN'])

## TODO: Add any additional arguments that you will need to pass into your model

# args holds all passed-in arguments
args = parser.parse_args()

# Read in csv training file
training_dir = args.data_dir
train_data = pd.read_csv(os.path.join(training_dir, "train.csv"), header=None, names=None)

# Labels are in the first column
train_y = train_data.iloc[:,0]
train_x = train_data.iloc[:,1:]

# Define a model
model = LinearSVC()

# Train the model
model.fit(train_x, train_y)
```

Next created the Sagemaker's sklean estimator:

```
from sagemaker.sklearn.estimator import SKLearn

estimator = SKLearn(entry_point='train.py',
                    source_dir='/home/ec2-user/SageMaker/Capstone-Starbucks-Project/source_sklearn',
                    role=role,
                    py_version="py3",
                    framework_version='0.23-1',
                    train_instance_count=1,
                    train_instance_type='ml.m5.large')
```

Then I trained the model and deployed it to an endpoint for testing.

```
# Train estimator on S3 training data
estimator.fit({'train': uploaded_data_s3})
```

```
%%time

# deploy model to create a predictor
predictor = estimator.deploy(initial_instance_count=1, instance_type='ml.m5.large')
```

```
-------------!CPU times: user 189 ms, sys: 23.3 ms, total: 213 ms
Wall time: 6min 31s
```

Model 2: Feed forward neural network (Binary Classifier)

For main model, Choice is between feed forward and convolutional NN (counters overfitting). But the data set looks quite simple so convolutional NN is not used and instead feed forward model is chosen to be the main model.

I want to create a binary classification feedforward models having hidden layers in between the input and the output layers. After every hidden layer, an activation function is applied to introduce non-linearity. When you have more than two hidden layers, the model is also called the deep/multilayer feedforward model or multilayer perceptron model (MLP).

Feed Forward Neural Network

- Input: 3 features
- 2 hidden layers fully-connected
- Activation function: ReLU, after each hidden layer

As a first step created model.py and used ReLU. And Sigmoid to output as I am doing binary classification.

```
def forward(self, x):
    """
    Perform a forward pass of the model on input features, x.
    :param x: A batch of input features of size (batch_size, input_features)
    :return: A single, sigmoid-activated value as output
    """

    # define the feedforward behavior of the network
    out = F.relu(self.fc1(x))
    out = self.drop(out)
    out = self.fc2(out)
    return self.sig(out)
```

Then using BinaryClassifier trained the model:

1.

```
# Build the model with input params.
# Model moved to designated device (gpu/cpu).
model = BinaryClassifier(args.input_features, args.hidden_dim, args.output_dim).to(device)

# Define an optimizer and loss function for training
optimizer = optim.Adam(model.parameters(), lr=args.lr)
criterion = nn.BCELoss()

# Train the model
train(model, train_loader, args.epochs, criterion, optimizer, device)
```

2.

```
# Provided training function
def train(model, train_loader, epochs, criterion, optimizer, device):
    """
    This is the training method that is called by the PyTorch training script. The parameters
    passed are as follows:
    model        - The PyTorch model to be trained.
    train_loader - The PyTorch DataLoader used during training.
    epochs       - The total number of epochs to train for.
    criterion    - The loss function used for training.
    optimizer    - The optimizer to use during training.
    device       - Where the model and data should be loaded (gpu or cpu).
    """

    # Training loop
    for epoch in range(1, epochs + 1):
        model.train() # Make sure that the model is in training mode.

        total_loss = 0

        for batch in train_loader:
            # Get data
            batch_x, batch_y = batch

            batch_x = batch_x.to(device)
            batch_y = batch_y.to(device)

            optimizer.zero_grad()

            # Get predictions from model
            y_pred = model(batch_x)

            # Perform backprop
            loss = criterion(y_pred, batch_y)
            loss.backward()
            optimizer.step()

            total_loss += loss.data.item()

        print("Epoch: {}, Loss: {}".format(epoch, total_loss / len(train_loader)))
```

Created pytorch estimator:

```
# import a PyTorch wrapper
from sagemaker.pytorch import PyTorch

# specify an output path
output_path = 's3://{}/{}'.format(bucket, prefix)

# instantiate a pytorch estimator
estimator = PyTorch(entry_point='train.py',
                    source_dir='source_pytorch',
                    role=role,
                    py_version="py3",
                    framework_version='1.0',
                    train_instance_count=1,
                    train_instance_type='ml.m5.large',
                    output_path=output_path,
                    sagemaker_session=sagemaker_session,
                    hyperparameters={
                        'input_features': 3, # number of features
                        'hidden_dim': 2,
                        'output_dim': 1,
                        'epochs': 80
                    })
```

Train the pytorch model:

```
%%time
# train the estimator on S3 training data
estimator.fit({'train': uploaded_data_s3})
```

Then instantiate the Sagemakers Pythorch Model and deploy.

```
from sagemaker.pytorch import PyTorchModel

# Create a model from the trained estimator data
# And point to the prediction script
model = PyTorchModel(model_data=estimator.model_data,
                     role=role,
                     py_version="py3",
                     framework_version='1.0',
                     entry_point='predict.py',
                     source_dir='source_pytorch')
```

```
%%time
# deploy and create a predictor
predictor = model.deploy(initial_instance_count=1, instance_type='ml.m5.large')
```

Results

Model Evaluation and Validation and justification.

Having trained and compared many, I chose as the final model that one which reached a high recall value.

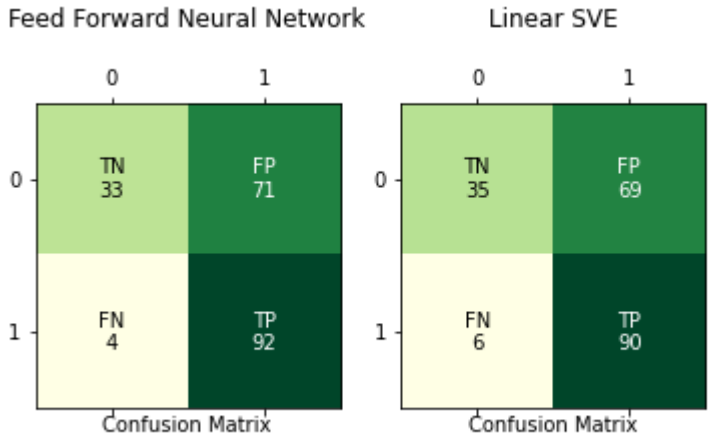
The final model was evaluated by using unseen data.

	Feed Forward NN	Linear SVC
accuracy (%)	62.50%	62.50%
precision	0.564	0.56
recall	0.958	0.94

As shown above, I got exactly 62.50 % accuracy from Linear SVC and the Feed forward neural network. But the intention is to get high recall value. The precision for feed forward classifier is bit higher than that of Linear SVC.

Finally, the recall is more than 95%. For feed forward classifier recall value is 0.958 which is higher than Linear SVC. So, feed forward neural network has performed better or equal in all the relevant parameters.

Comparing confusion matrix:



Side by side comparison of confusion matrix shows that feed forward classifier got higher true positive (92). This was the main aim of my modelling. That is to get higher recall or true positives.