Capstone Project

Machine Learning Engineering Nanodegree

✓ Definition

Project Overview

The task is basically a case in **customer segmentation**. It's going to help Starbucks to create better and more targeted promotional offers for customers; this will lead to better customer satisfaction, and eventually more revenue for the company. I've used some EDA and simple decision tree classifier to find segments of customers that are more likely to actually use the promotional offers when they get them.

Problem Statement

We have a dataset of about one month for a single product in Starbucks. This includes various offers that have been made and customers' response to those offers as well as other purchases that customers have made without using any promotional offer.

The dataset is composed of three sources:

- **portfolio.json** - containing offer ids and meta data about each offer (duration, type, etc.)
- **profile.json** - demographic data for each customer
- **transcript.json** - records for transactions, offers received, offers viewed, and offers completed

I'm going to find a series of rules, based on my exploratory data analysis and decision tree classifiers, to find how certain types of customers respond to certain kind of offers.

Metrics

I have used *accuracy* as a metric to train my decision trees in order to find various segments of customers that responds differently to the offers. Accuracy is defined as follows:

Accuracy = (sum of predictions that match the actual labels) / (total number of cases/labels)

I'm not after finding the most accurate classifier here; rather I'm using the metric as a guide to guide my classifier to find different partitions of customers that may reply differently to various ads, or offers.

✓ Analysis

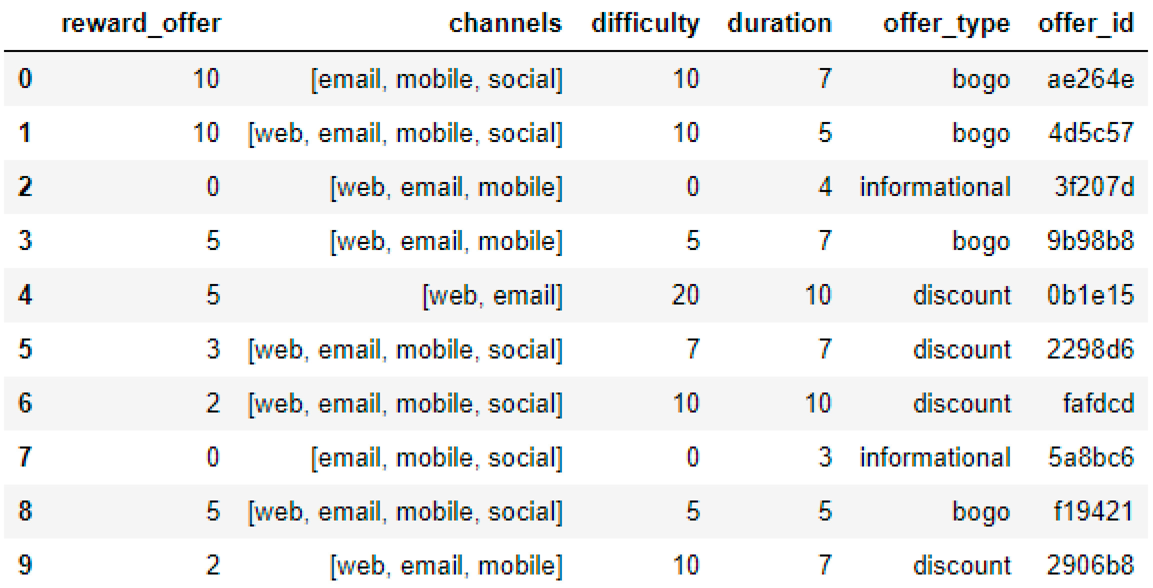
Data Exploration

The dataset is composed of 3 different sources:

1. A **portfolio** of various offers and their characteristics, including their duration, the channel that offer was promoted etc.
2. A **profile** of the customers: their age, income, their tenure etc.
3. And the **transcript** data which includes about 300K events: offer received, viewed, transaction, and completed.

Overall, we have about 17,000 customers and more that 300,000 transactions made by them. There are 10 different offers, which are various degrees of 3 offer types: a **bogo** is bye-one-get-one, a **discount**, and a simply **informational offer** which is an advertisement about the product.

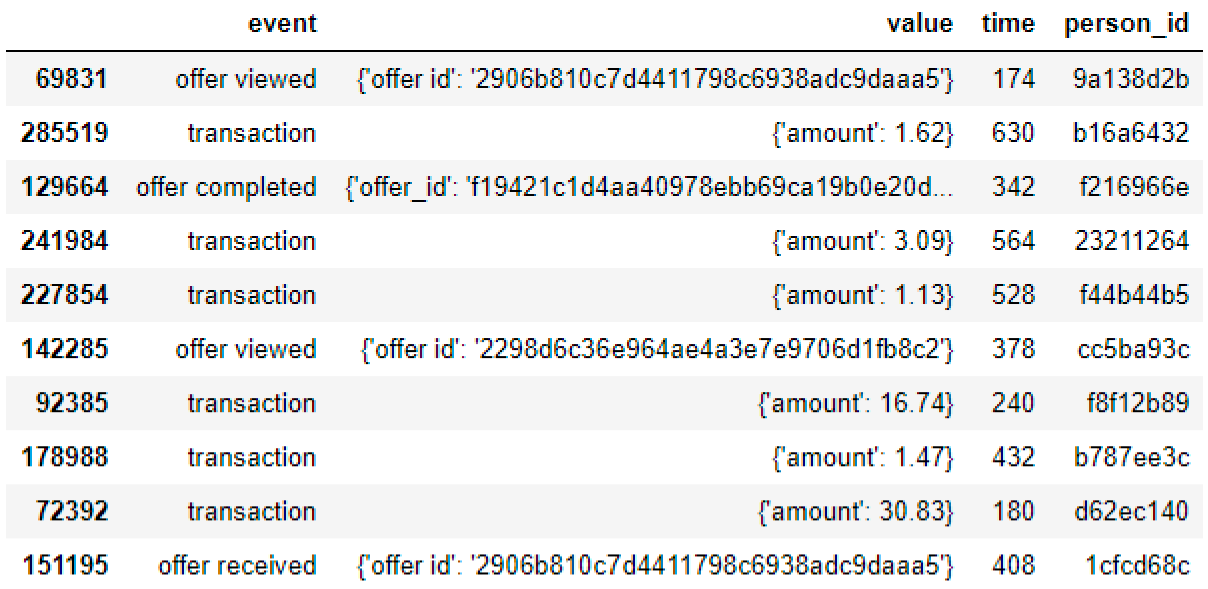
In the following picture you could see various offers and their characteristics.



	reward_offer	channels	difficulty	duration	offer_type	offer_id
0	10	[email, mobile, social]	10	7	bogo	ae264e
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57
2	0	[web, email, mobile]	0	4	informational	3f207d
3	5	[web, email, mobile]	5	7	bogo	9b98b8
4	5	[web, email]	20	10	discount	0b1e15
5	3	[web, email, mobile, social]	7	7	discount	2298d6
6	2	[web, email, mobile, social]	10	10	discount	fafdcd
7	0	[email, mobile, social]	0	3	informational	5a8bc6
8	5	[web, email, mobile, social]	5	5	bogo	f19421
9	2	[web, email, mobile]	10	7	discount	2906b8

- 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)

A sample of transactions made during that month is as follows:

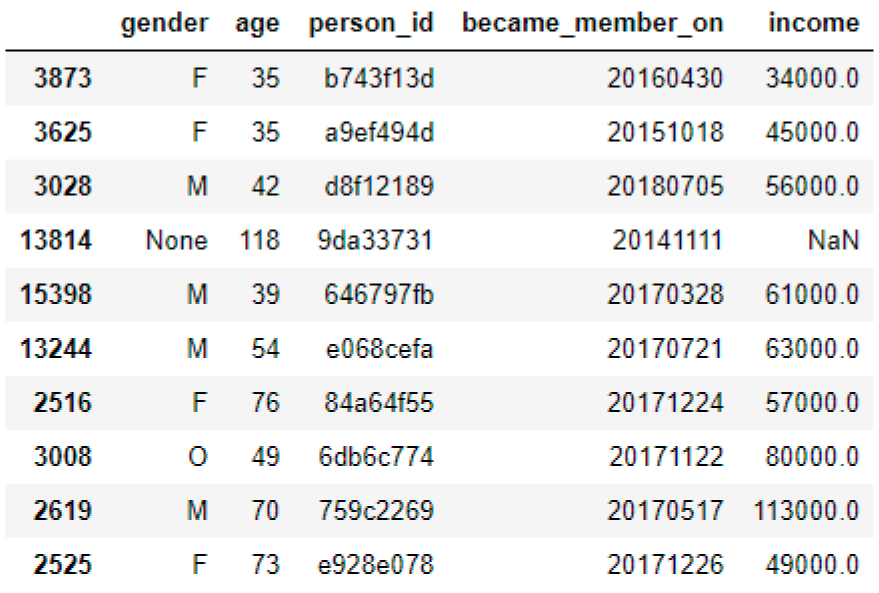


	event	value	time	person_id
69831	offer viewed	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	174	9a138d2b
285519	transaction	{'amount': 1.62}	630	b16a6432
129664	offer completed	{'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d...	342	f216966e
241984	transaction	{'amount': 3.09}	564	23211264
227854	transaction	{'amount': 1.13}	528	f44b44b5
142285	offer viewed	{'offer id': '2298d6c36e964ae4a3e7e9706d1fb8c2'}	378	cc5ba93c
92385	transaction	{'amount': 16.74}	240	f8f12b89
178988	transaction	{'amount': 1.47}	432	b787ee3c
72392	transaction	{'amount': 30.83}	180	d62ec140
151195	offer received	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}	408	1cfcd68c

- event (str) - record description (i.e., transaction – purchases made without any offer, offer received, offer viewed and offer completed)
- person id (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dictionary of strings) - either an offer id or transaction amount depending on the record

In order to be able to use the information in the *value* column, I had to do some preprocessing to extract various information, which I'll explain later.

A random sample of profile data is as follows:



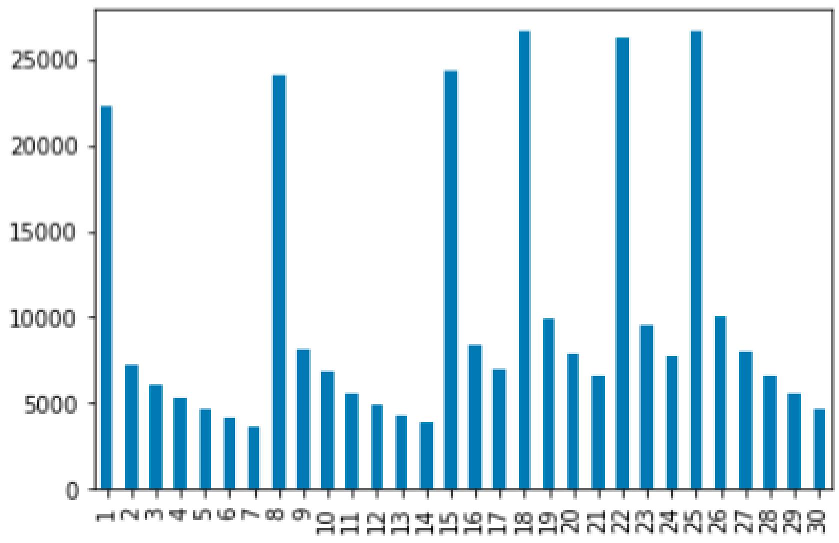
	gender	age	person_id	became_member_on	income
3873	F	35	b743f13d	20160430	34000.0
3625	F	35	a9ef494d	20151018	45000.0
3028	M	42	d8f12189	20180705	56000.0
13814	None	118	9da33731	20141111	NaN
15398	M	39	646797fb	20170328	61000.0
13244	M	54	e068cefa	20170721	63000.0
2516	F	76	84a64f55	20171224	57000.0
3008	O	49	6db6c774	20171122	80000.0
2619	M	70	759c2269	20170517	113000.0
2525	F	73	e928e078	20171226	49000.0

- 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

About 40% of the customers are female. Half of them is male, and the rest not identified.

Exploratory Visualization

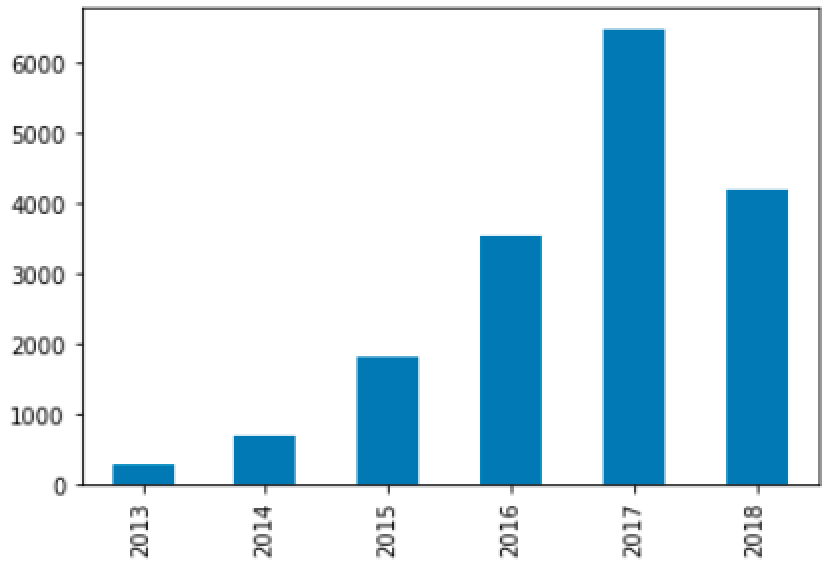
A curious aspect of the dataset is the frequency with which the offers have been made:



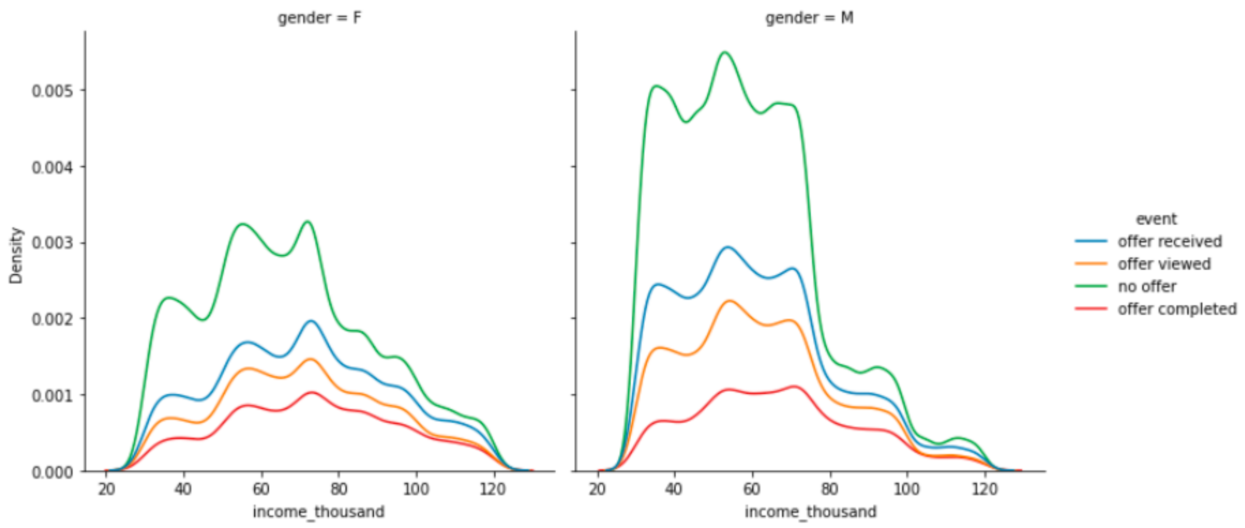
As you see, almost every 7 days a new round of offers is made, most of the *actions* happen during that same day – it could be that customers act upon the offer of simply view it, or even ignore – and then it dies down.

An initial suggestion is to change this policy if possible, rather than every other 7 days, a policy of twice a week would be more beneficial, as we could see in fact when offers have been made in days of 18 and 22 and 25 of that month.

Customer tenure – since when they've joined the app – also varies withing customers:



Naturally income-base of customers varies among them:

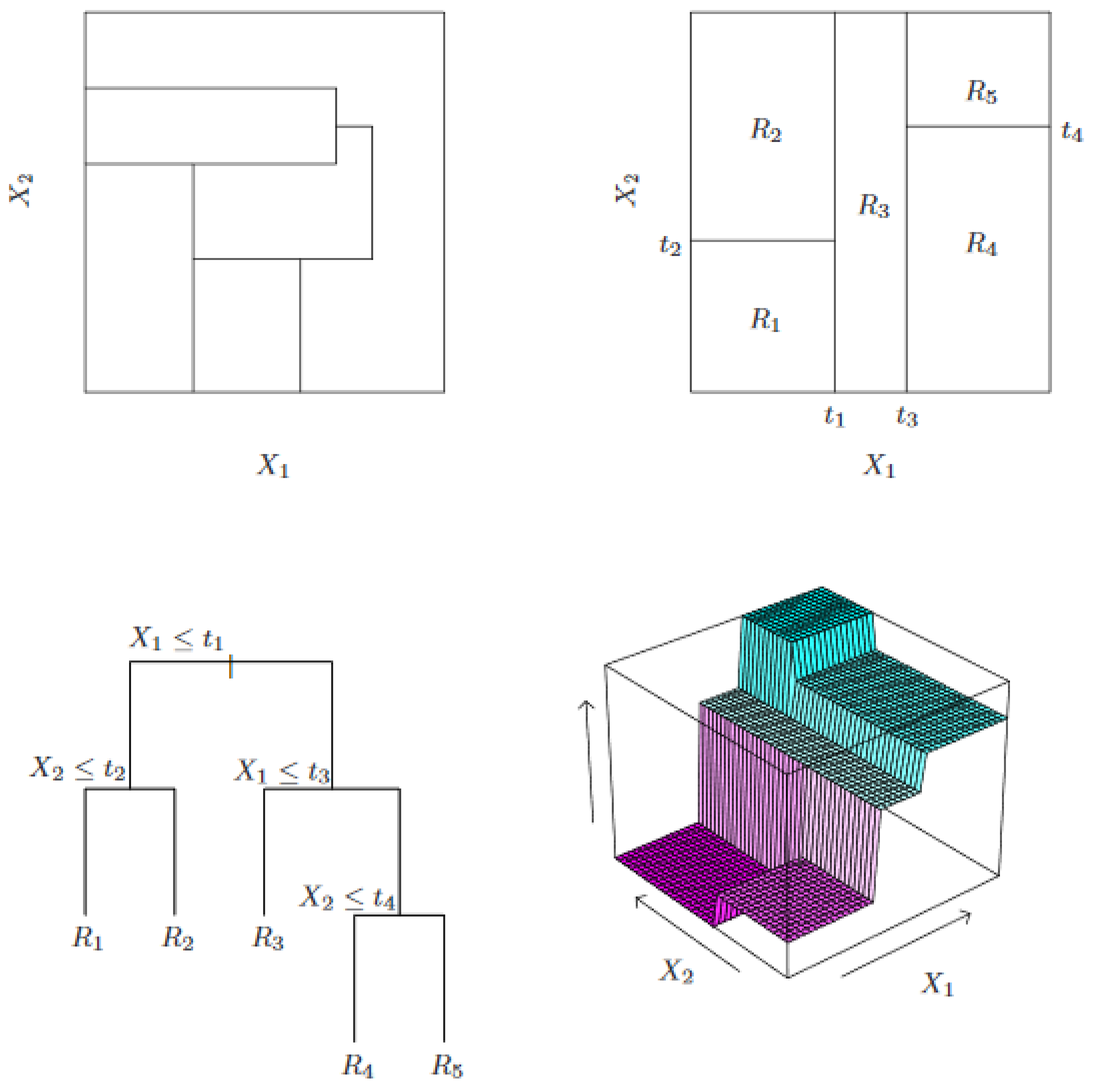


Here, we see that most of the transactions are made buy customers who are male and earn less than 80K a year.

Algorithms and Techniques

As mentioned above I'll be using exploratory data analysis as well as decision-tree classifiers to find clusters of customers and justify their behaviors to the offers.

Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant when dealing with a regression task, or majority vote in classification) in each one. This makes them a powerful technique especially suited for our task. Because the outcome is a set of if-then-rules that could be used as a recommendation for our advertising and promotion strategy. A short demonstration of the algorithm is as follows:



In the above diagram, top right panel shows a partition of a 2D feature space by recursive binary splitting. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel – to see the tree plot for our model please check out the graphs in the image directory, and a perspective plot of the prediction surface appears in the bottom right panel.[1]

This partitioning makes decision trees powerful for our purpose, because they'll be less prone to suffer from outliers. Other strength of decision-tree-based algorithms that justifies my choice is that they exclude unimportant features, that is, they perform some form of feature selection which is really important for customer segmentation tasks. They also can be interpreted without a mathematical background. This is important if we want to communicate with the marketing or product team.

On the other hand, decision trees have their weaknesses: most importantly, they have trouble modeling some relationships due to reliance on axis-parallel splits, as seen on the diagram. Also, they are prone to overfit, as small changes in the data could result in large changes to the decision logic.

I've used two various decision trees; they differ based on how they split in each round. One is the default which finds the best fit, and the second one will do it randomly. I'll justify my approach later on in the analysis. Also, I've found that **entropy** is a better choice for this dataset as **criterion** for split in each node.

Some preprocessing also had to be done, as decision trees do not handle missing values in the input data.

Benchmark

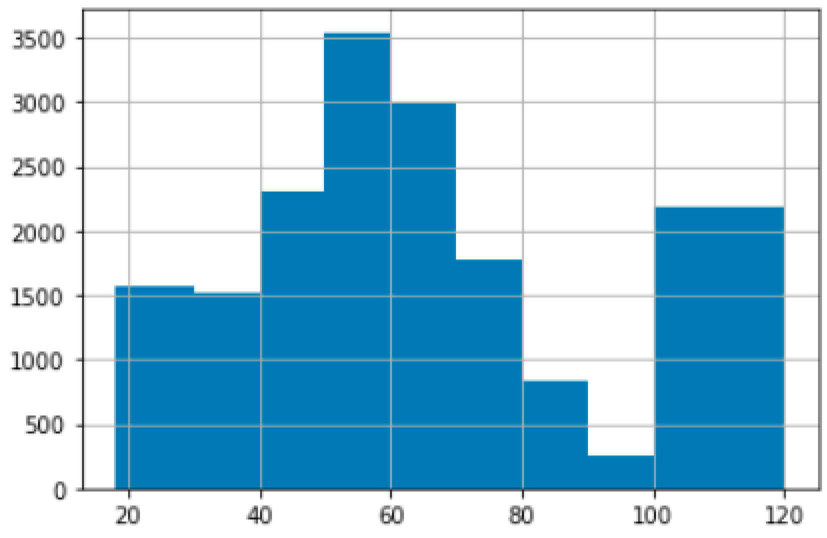
As a benchmark for this project few ideas come to mind, one would be to simply assume that all offers will be completed by the customers, which is obviously not realistic. A more realistic offer would be to go by the rule of third: out of all the offers, a third of them will be viewed by our customers, and a third of those will be completed, i.e., 1/9 of the offers will be completed. Almost 11%.

My hope is to shed some light about the customer profile and how certain types of customers act on various offers, and perhaps to beat the benchmark as well.

✓ Methodology

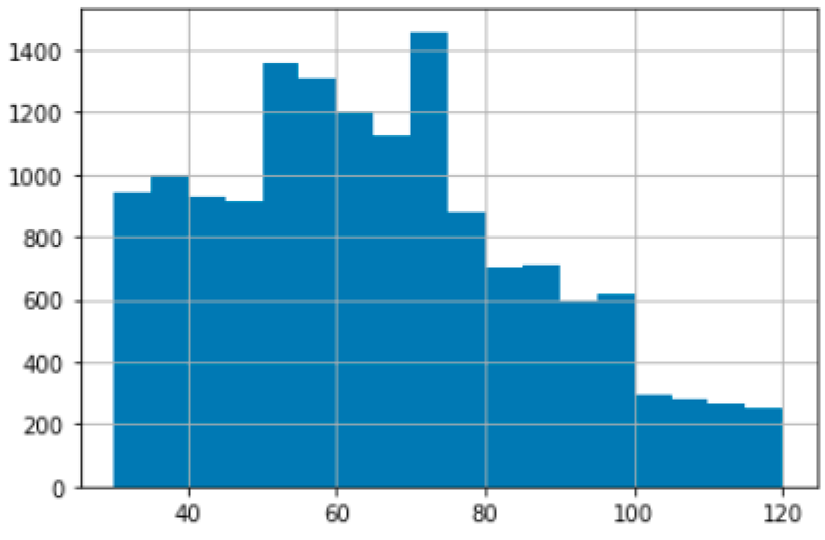
Data Preprocessing

A curious aspect of the data was encoding the missing values in **age**.



As you see, there are quite of customers that are above 90-year-old. It turns out that missing values are encoded as 118 in this dataset. I've also treat the values above 90 as missing values since, in the absence of any more information, I'd find that information not so reliable.

Decision trees are not affected by feature scale since they simply split on the feature space, however I've divided the original income data by 1000 for ease of visualization.



We can divide customers into two categories of bellow and above 80K.

As mentioned previously, I also had to extract features from *value* column in the profile data, as the column was coded as a dictionary which could not be used as is in the analysis.

```
dfinfo.iloc[129664]
```

```
event                                                    offer completed
value          {'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d...
time                                                                 342
person_id                                                       f216966e
Name: 129664, dtype: object
```

```
dfinfo.iloc[129664]['value']
```

```
{'offer_id': 'f19421c1d4aa40978ebb69ca19b0e20d', 'reward': 5}
```

For customers that have **completed** the offer, the dictionary involved two values, **offer_id** and **reward** which was offered by engaging the promotional offers. Here, we cannot know what is the actual amount that customers have purchased, we simply know that they've made a purchase that enabled them to be eligible to use the offer, i.e., we only know the minimum amount that is required to make a purchase to use the offer.

For customers that made a purchase without any offer we have the actual purchase amount though.

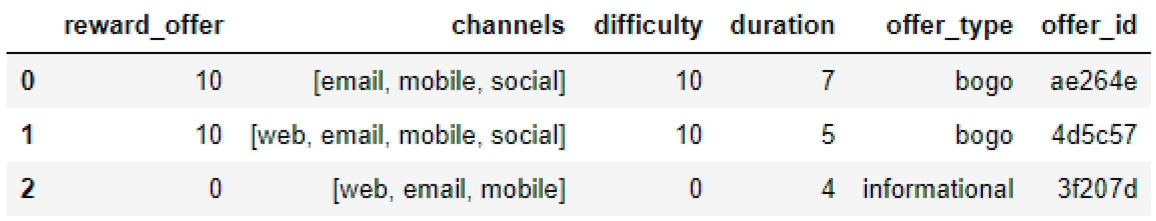
Implementation

I've used the **event** as my target variable of choice, since I want to know given a certain profile of a promotional offer, what the reaction of a certain customer would be. The feature space is the following variables: **difficulty, duration, member_year age, income_thousand, reward_offer, gender**.

As a preprocessing technique, I've used a naïve imputer to impute missing values with most frequent values, and used one-hot-encoding for converting categorical features to numeric, so that they will be used by the model.

A complication of the dataset is that like so the data is **imbalanced** as majority of transactions are made without usage of any offers, and **completed offers** are in the minority category. As a result, to fit the algorithm better for out final goal – i.e., finding the customer clusters – I've had to use **class_weight = 'balanced'** in the decision trees.

Another complication of data is that in the dataset, offers have duration, during which they are valid:



	reward_offer	channels	difficulty	duration	offer_type	offer_id
0	10	[email, mobile, social]	10	7	bogo	ae264e
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57
2	0	[web, email, mobile]	0	4	informational	3f207d

This way, algorithm will always have easy time to find customers that make purchase without any offer being used, since the **duration** value for those purchases will be zero, as I coded in the dataset. Or perhaps I could have coded as NA, but then it would not help my case anyway. To circumvent this issue, I've tried fitting a decision tree without this feature, but it did not help.

A third complication is handling missing values, as I mentioned above, I've used a naïve imputer: filling missing values with the most frequent value. Perhaps more advanced imputation techniques could help. Or perhaps we could talk to the product/marketing team to see if we could gather more information to fill the missing values accurately.

In the end, the model accuracy was 65.7% on the training set, and 33.4% on the test set.

Refinement

To better fit the classifier to the data, I tweaked the initial hyper-parameters in two ways. First, I found that **entropy** helps better as opposed to **gini** for the criterion to split. I honestly don't know why that is.

Also, again because of the **duration** – which will be zero if the transaction made is without using any offer – made learning from data much harder. I found that using **randomization**, i.e., splitting randomly, helped finding better clusters.

Results

Model Evaluation and Validation

Since the central problem of the project is to find customer segments and their probable response to various offers, I've used a decision tree classifier in combination with simple exploratory data analysis.

My final model's accuracy was a bit over 1/3, which would be barely an improvement above the benchmark – 3 class classification.

Am I confident enough to make firm recommendations based on this model? I would rather prefer to continue reading the literature, or talking to domain experts to find either better features or better algorithms.

All in all, the followings are some insights I've found during my analysis:

1. Cheaper buy-one-get-one offers are better in demand, especially by those earning bellow 80K.
2. Making the offers more frequent – twice a week perhaps – is highly likely to improve the final outcome of desires, i.e., offers getting completed rather than simply seen or ignored.
3. There is not a major distinction base on gender.
4. Tenure counts. The older customers are more likely to complete an offer.
5. The higher the reward of an offer the more likely that an offer gets completed.

Please refer to the image **model-02** in the **images** directory for the plot of decision trees. You may need to zoom to read the labels.

Justification

Major choice of hyperparameters was picking random split. I've also picked a naïve imputer – filling with most frequent values – for missing values imputation.

Some findings made sense, like gender not playing a major role. But the fact that more recent customers are more likely to complete an offer was counter intuitive to me. I would have expected otherwise.

Another surprising factor was that income did not play a major role. Only based on my own EDA, I found out that people with less than 80K earnings per year are more likely to prefer cheaper **bogo offers**, as explained above.

Overall, this was a nice first attempt at the problem as we got 33.4% accuracy on the test data, which beats the 11% benchmark. However, I'm not ready to make firm recommendations based on these findings. Perhaps more knowledge from marketing, or domain experts would help my analysis.

Conclusion

Reflection

The process used for this project can be summarize as follows:

- ✓ Cleaning the data and encoding the feature names properly.
- ✓ Deriving features from abnormally encoded features like *value* in the transcript data.
- ✓ Performing EDA to find interesting features of dataset.
- ✓ My benchmark of use would be simple classification since the model is 3-class-classification – offers viewed, offers received, and offers completed.
- ✓ Performing preprocessing to make the data ready for the learning algorithm.
- ✓ Fitting variations of decision tree classifiers to find various customer segments.

I found the project more challenging that expected. There were no easy-to-find clusters to be found. And features like *duration* my it more difficult for the algorithm to actually find interesting patterns, despite even trying to circumvent the issue by *splitting randomly* on each node.

Improvement

Perhaps some advanced feature engineering could help, for example using k-nearest-neighbors-based features. However, I couldn't implement such features, and also I am not even sure if this would help my case, since I'm not after the most accurate classifier, rather one that could identify interesting clusters of customers that respond differently to offers as expected.

Also, I would have loved to have the actual amount purchased for **offer completed** events. I think this would help the overall assessment and analysis of the problem.

Reference

Hastie et all, *The elements of statistical learning: data mining, inference, and prediction*, 2008, Springer