Machine Learning Engineer Nanodegree

Capstone Project

Dimitri Tarasowski
March 10th, 2020

I. Definition

Project Overview

The project is about creating an offer recommendation engine for the Starbucks marketing team. Every few days, Starbucks is sending out offers to users of the mobile app. There are mainly three main types of offers: discount, BOGO (buy one get one) and informational offers. Not all Starbucks customers receive the same offer.

The dataset is the simplified version of the real Starbucks app data. The dataset contains three files: portfolio.json, profile.json, and transcript.json. The task is to combine transaction data with the demographic and offer data to identify which type of customers fit well to which type of offers.

Problem Statement

The problem of the project is to find a solution that will take customer attributes and offer attributes into account and suggest if the offer will be successful or not. By doing so Starbucks team will be able to check to which offer a customer will respond. By finding the right offer for the right customer Starbucks will be able to increase the marketing ROI. Also Starbucks will be able to target only customers that will respond to the offers, instead of sending the same offer to everyone and customers may become annoyed by Starbucks's advertising.

Metrics

As the main evaluation metric, the accuracy score will be used. For this type of problem, the accuracy score is perfectly fine. There is no huge imbalance in the dataset, also the imbalance will be fixed by oversampling. In order to fine-tune the model further, the f1, precision and recall metrics will be used to dig deeper into models performance.

As a benchmark model, a Naïve Classifier will be used. A Classifier that will simply mark all offers as successful. By using the Naïve Classifier we can see how well the new model will perform in comparison to a random classification. The dataset will be combined and cleaned for a binary classification approach. Mainly the goal will be to predict if the user should receive the BOGO or the discount offer.

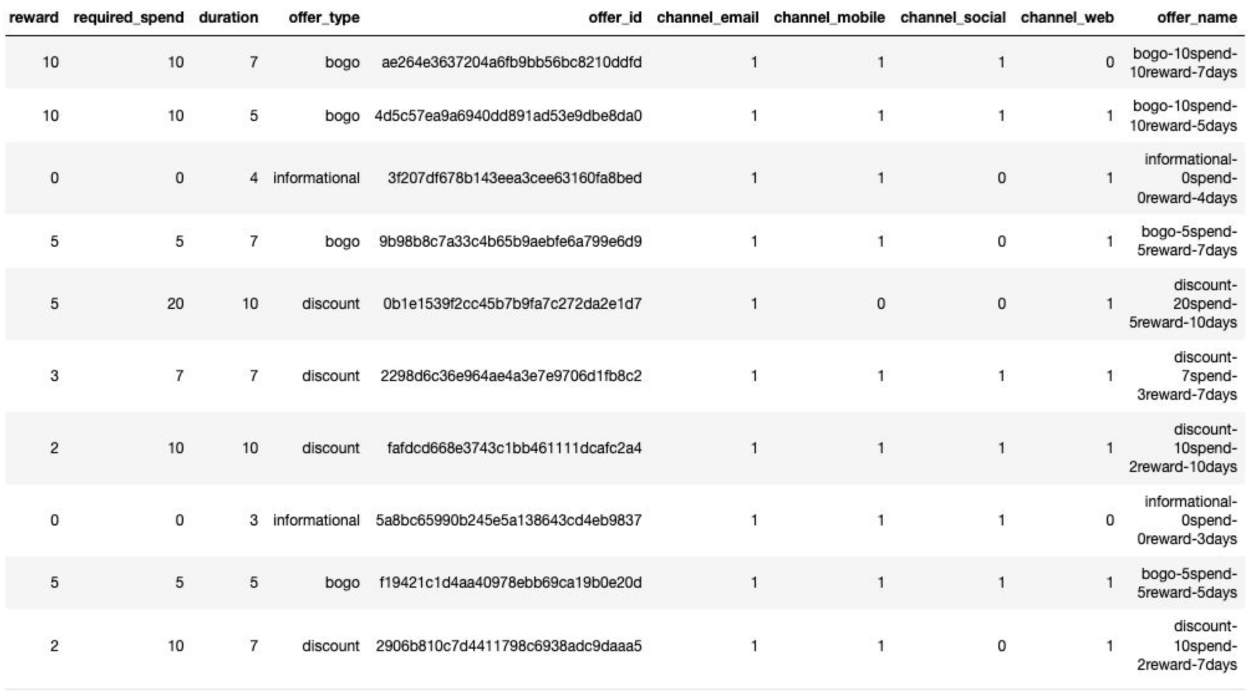
II. Analysis

Data Exploration

As already mentioned there are three files provided.

- portfolio.json: contains information about the offer attributes
- profile.json: contains information about the customer attributes
- transcript.json: contains information about events such as offer views and transactions

Figure 1.0 shows an already transformed portfolio.json data frame. In order to simplify the view for later analysis, the channel column was dropped and converted into one-hot-encoded values for channels. Also for later steps in the analysis, a specific offer_name column was constructed to have a clearer naming.



reward	required_spend	duration	offer_type	offer_id	channel_email	channel_mobile	channel_social	channel_web	offer_name
10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1	0	bogo-10spend-10reward-7days
10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1	1	bogo-10spend-10reward-5days
0	0	4	informational	3f207df678b143eea3cee63160fa8bed	1	1	0	1	informational-0spend-0reward-4days
5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	0	1	bogo-5spend-5reward-7days
5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0	1	discount-20spend-5reward-10days
3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1	1	discount-7spend-3reward-7days
2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1	1	discount-10spend-2reward-10days
0	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	1	1	1	0	informational-0spend-0reward-3days
5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1	1	bogo-5spend-5reward-5days
2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	0	1	discount-10spend-2reward-7days

Figure 1.0: Modified portfolio data frame

In the next step of the data exploratory part, the profile.json file was analyzed and modified. The profile data frame contains 17000 entries of customer profiles. Figure 1.1 shows an overview of already modified profile data frame. Here there are two main things to point out. There is an issue with the age column, there are entries with an age of 118 years which is not possible. The decision was made to drop the rows with the age of 118 years. The next issue is with the gender column. There are some rows marked as O (others), as shown in figure 1.1. there is a very low number of O values. This imbalance can cause serious issues for certain algorithms. Therefore the decision was made to drop all O values.

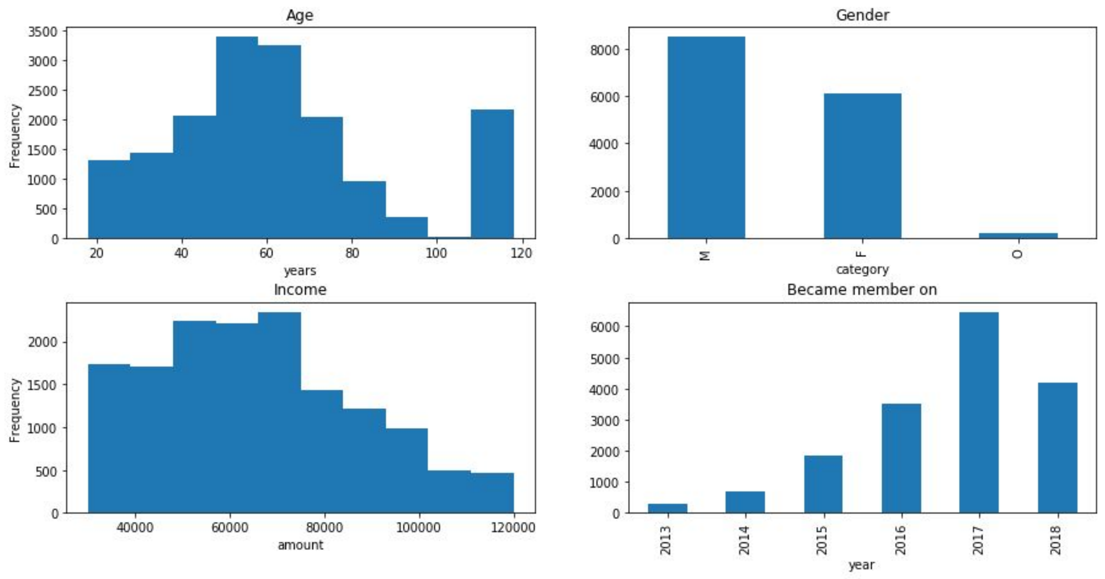


Figure 1.1: Basic visualisation of 4 columns

As shown in 1.1. Other columns such as income and became a member on (custom made column) are fine for further analysis. Figure 1.2 shows the distribution after dropping the rows of age and gender columns.

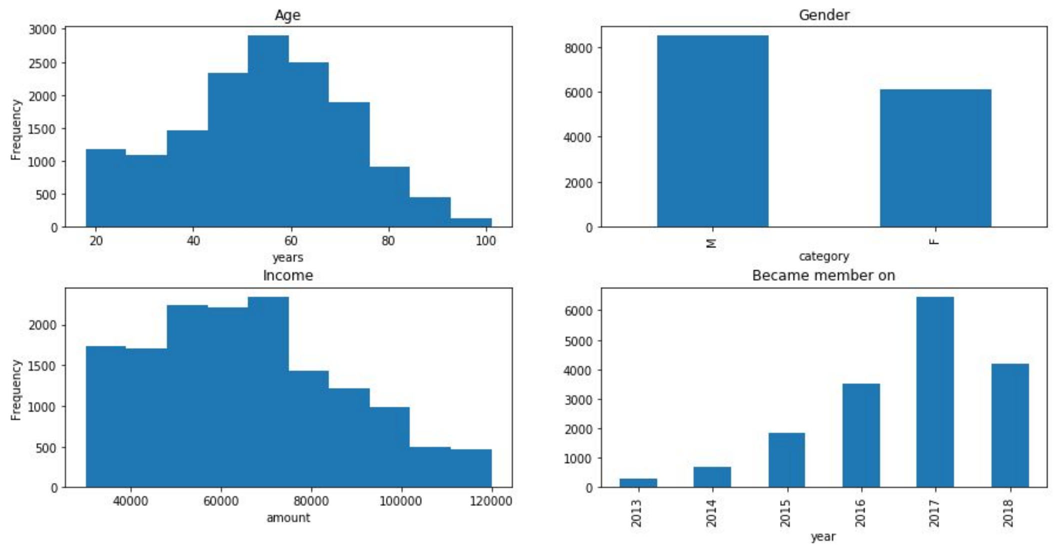


Figure 1.2: Basic visualisation of 4 columns after dropping age and gender

Next, the goal was to see if the profile data frame has some other missing values. Figure 1.3 shows a heatmap that shows a high level overview of rows with missing values. It turns out that exactly 2387 rows contain missing values in age, gender, and income columns. Since for gender or age it makes no sense to fill the values e.g. with median or any other values, the decision was made to drop the rows that contain missing values.

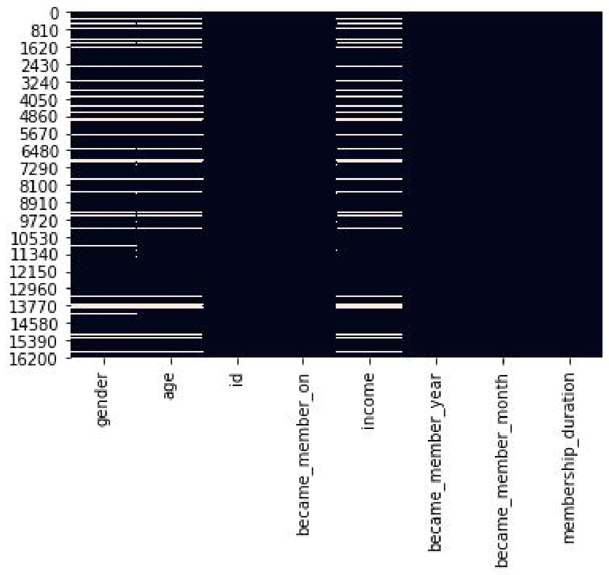


Figure 1.3: Missing values heatmap

Once the missing values were dropped a new heatmap was constructed to make sure that no missing values are present in the data frame (see figure 1.4).

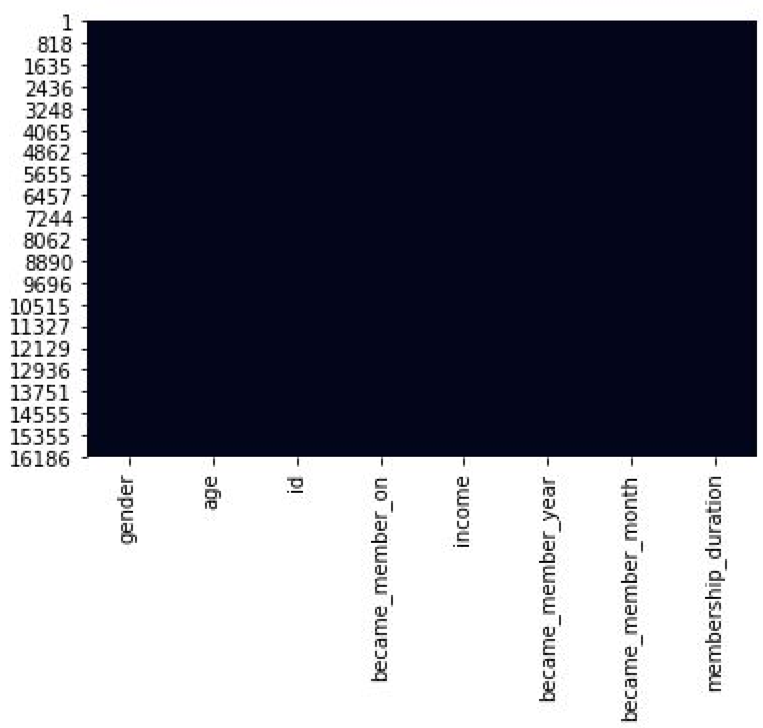
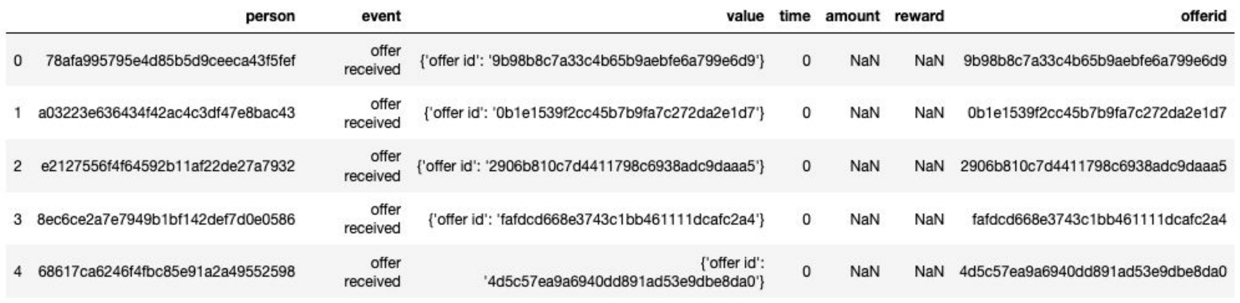


Figure 1.4: Missing values heatmap after dropping missing values

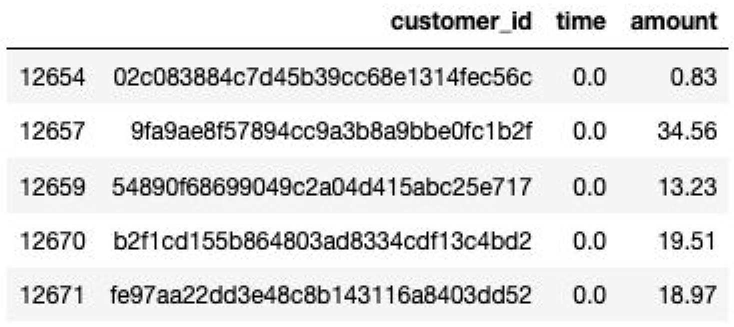
After looking into the profile data frame, removing the missing or ambiguous values in the next step of the project the transcript data frame was analyzed in detail. The transcript data frame has a column value that encodes the offer id and potential reward in case the offer was successful. At first, the column value was parsed and spread out into its own columns: amount, reward and offerid. Figure 1.5 shows already the new additional columns.



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

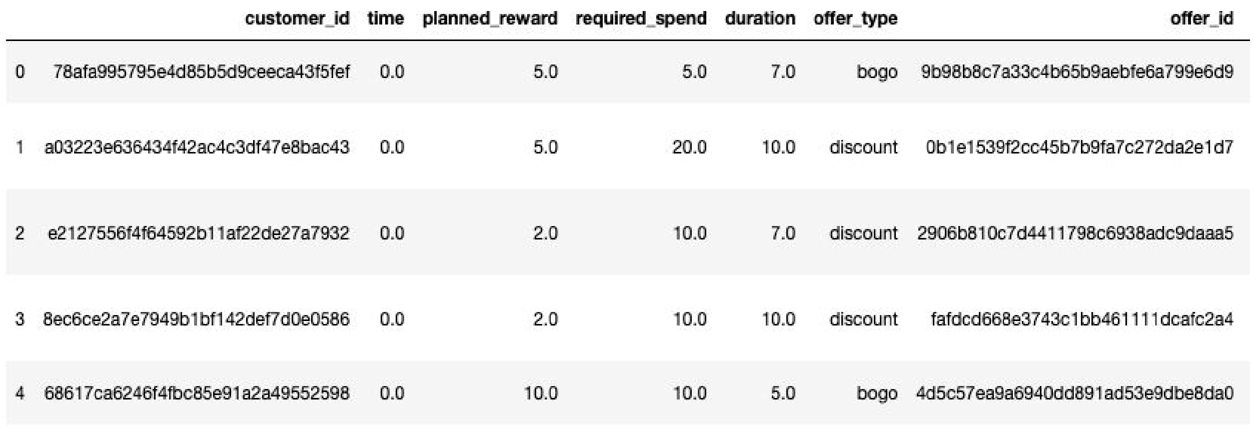
Figure 1.5: Offer data frame with new columns

The operation was needed so later the offer event data can be combined with the offer attributes from the portfolio data frame. The biggest problem with the transaction data frame is that transactions, as well as offer events, are combined into a single data frame (file). To create aggregated metrics on an offer basis it was needed to split the transcript data frame into two main parts: transactions and offer events. Figure 1.6 shows the transactions data frame and figure 1.7 shows the offer events data frame.



	customer_id	time	amount
12654	02c083884c7d45b39cc68e1314fec56c	0.0	0.83
12657	9fa9ae8f57894cc9a3b8a9bbe0fc1b2f	0.0	34.56
12659	54890f68699049c2a04d415abc25e717	0.0	13.23
12670	b2f1cd155b864803ad8334cdf13c4bd2	0.0	19.51
12671	fe97aa22dd3e48c8b143116a8403dd52	0.0	18.97

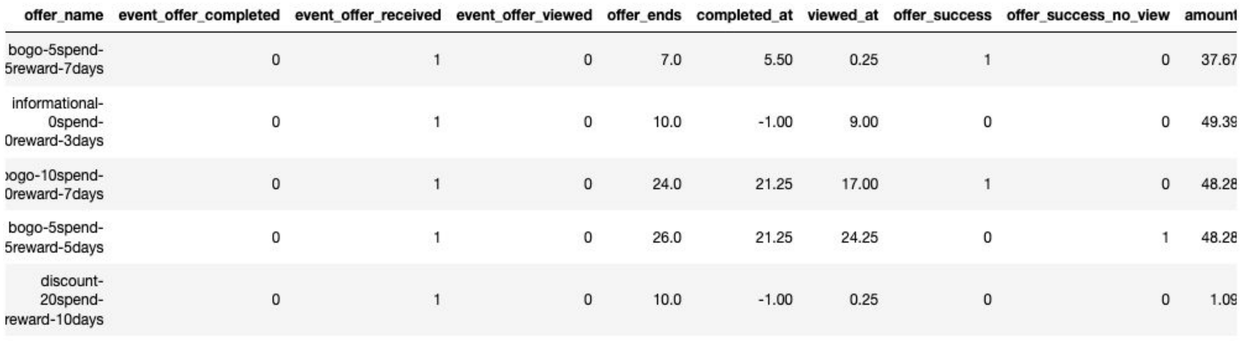
Figure 1.6: Transactions data frame



	customer_id	time	planned_reward	required_spend	duration	offer_type	offer_id
0	78afa995795e4d85b5d9ceeca43f5fef	0.0	5.0	5.0	7.0	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	a03223e636434f42ac4c3df47e8bac43	0.0	5.0	20.0	10.0	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	e2127556f4f64592b11af22de27a7932	0.0	2.0	10.0	7.0	discount	2906b810c7d4411798c6938adc9daaa5
3	8ec6ce2a7e7949b1bf142def7d0e0586	0.0	2.0	10.0	10.0	discount	fafdcd668e3743c1bb461111dcafc2a4
4	68617ca6246f4fbc85e91a2a49552598	0.0	10.0	10.0	5.0	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0

Figure 1.7: Offer events data frame

The last part of the data exploratory was dealing with the aggregation of metrics. The last part was the most time-consuming part of the project. It was important to calculate aggregated metrics such as the total amount and if the offer was successful or not with regards to timestamps of the events. Since the transactions were not stitched to specific offer ids or any other ids that can be used to merge the data, the decision was made to create a function that was comparing the timestamps of events such as offer received, offer viewed, offer completed to calculate if the offer completion can be attributed to the offer itself or not. Also, the amount spent was calculated on the window between the offer received and offer completed. Figure 1.8 shows the combined final data frame that was used for further analysis. One thing to mention here, since some offers have overlapping time windows, the amount spent can be duplicated across the offers, because the amount spent is calculated within offer received time and offer duration, during that time everything that is found in transactions is attributed to the offer. There is no other way to attribute the amount spent to offer other than doing window calculation.



offer_name	event_offer_completed	event_offer_received	event_offer_viewed	offer_ends	completed_at	viewed_at	offer_success	offer_success_no_view	amoun
bogo-5spend-5reward-7days	0	1	0	7.0	5.50	0.25	1	0	37.67
informational-0spend-0reward-3days	0	1	0	10.0	-1.00	9.00	0	0	49.39
›ogo-10spend-0reward-7days	0	1	0	24.0	21.25	17.00	1	0	48.28
bogo-5spend-5reward-5days	0	1	0	26.0	21.25	24.25	0	1	48.28
discount-20spend-reward-10days	0	1	0	10.0	-1.00	0.25	0	0	1.09

Figure 1.8: Combined transactions and offer attributes data frame

Exploratory Visualization

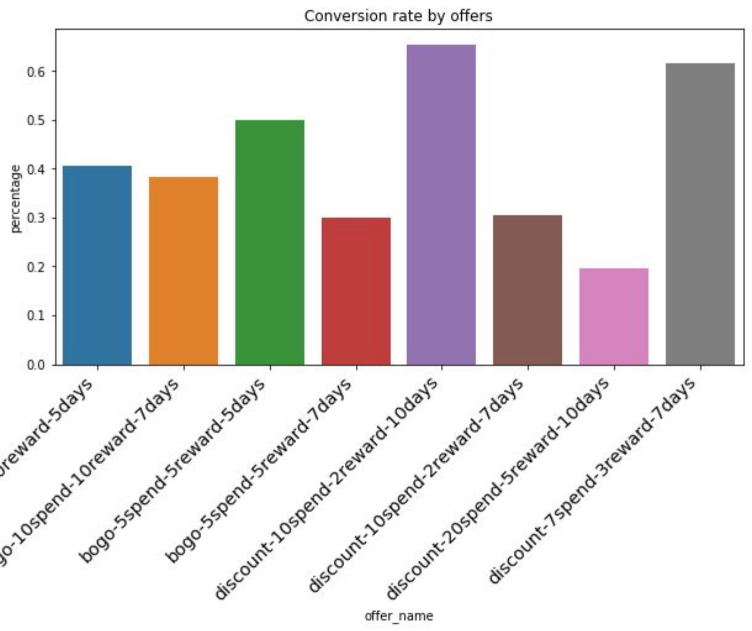


Figure 2.0: Conversion rate by offers

Figure 2.0 shows the conversion rate by offers. Based on that information a decision can be made that the discount offer to spend $10 with a $2 reward within 10 days has the highest conversion rate. Next to it, there is a discount offer to spend $7 with a $3 reward within 7 days. The least performing offer is a discount offer to spend $20 with a $5 reward within 10 days. Based on the information a decision can be made that there needs to be a balance between the amount spent and the duration of the offer.

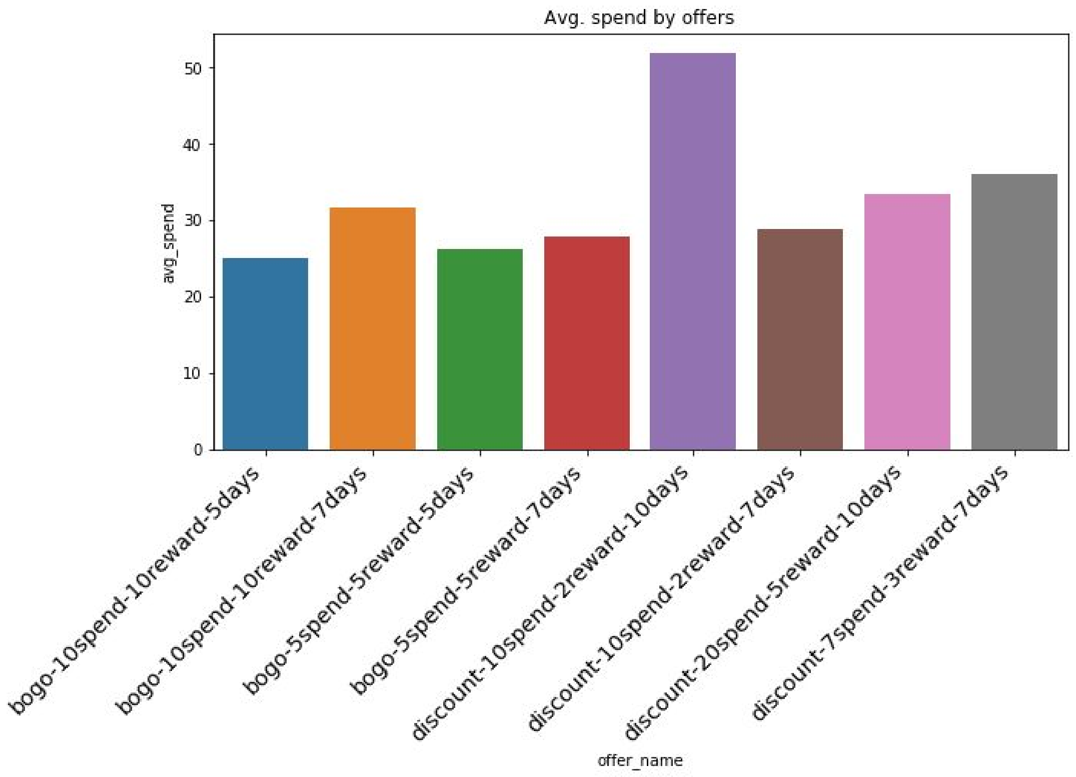


Figure 2.1: Average spend by offers

As shown in figure 2.1 the best performing offer in terms of average spending is the discount offer to spend $10 with a 2$ reward within 10 days. The offer seems to be the optimal one for the majority of the customers of Starbucks. Also, the second-best offer in terms of conversion rate, the discount offer to spend $7 with a $3 reward within 7 days performs slightly better than the average.

In figure 2.2 there is an overview of average spend by gender. The figure clearly shows that women more than men. The difference is around $10 per customer. It comes mainly from the fact that women have a higher conversion rate in comparison to men, see figure 2.3. Women have an almost 5% higher conversion rate in comparison to men. It's also to note that the current dataset is disbalanced towards men, there are almost 2000 less female customers.

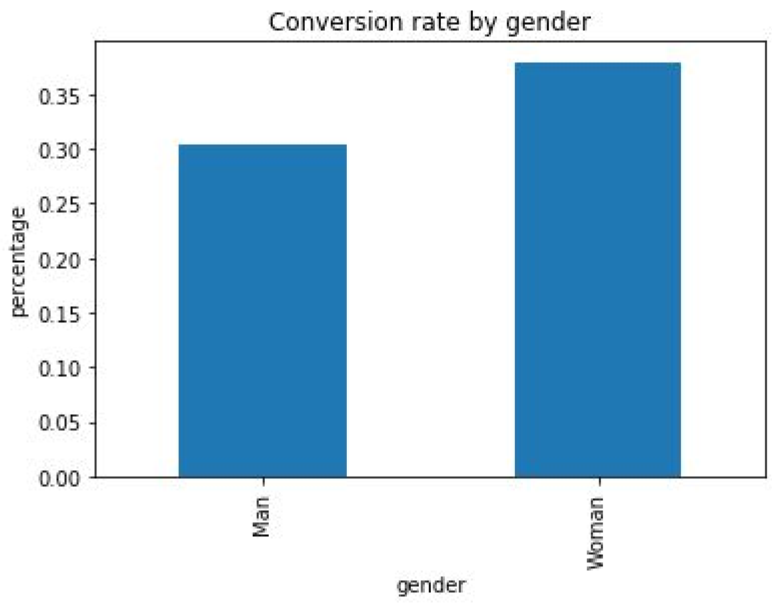


Figure 2.2: Average spend by gender

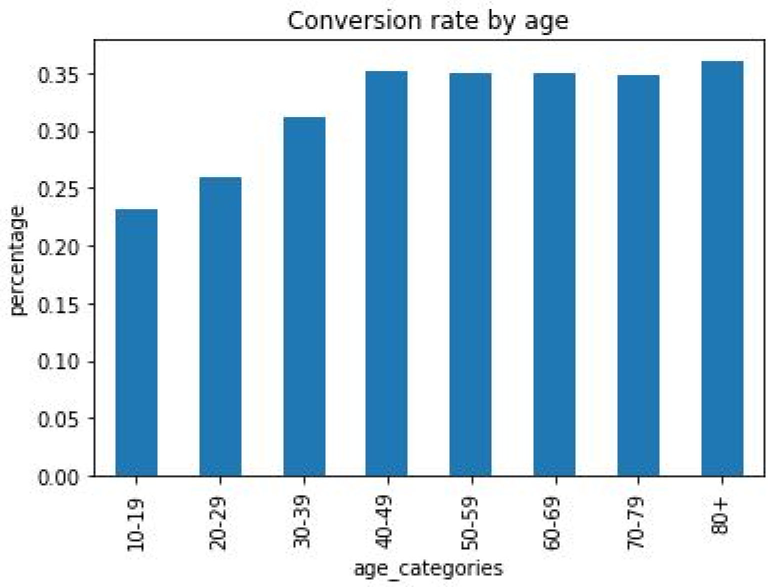


Figure 2.3: Conversion rate by gender

Figure 2.4 shows the difference in conversion rate by the offer by gender. As shown here, women perform well on both offers, the conversion rate is similar, while men tend to have a higher conversion rate on discount offers. Noted here the informational offer has no conversion rate because there is no way to complete an offer.

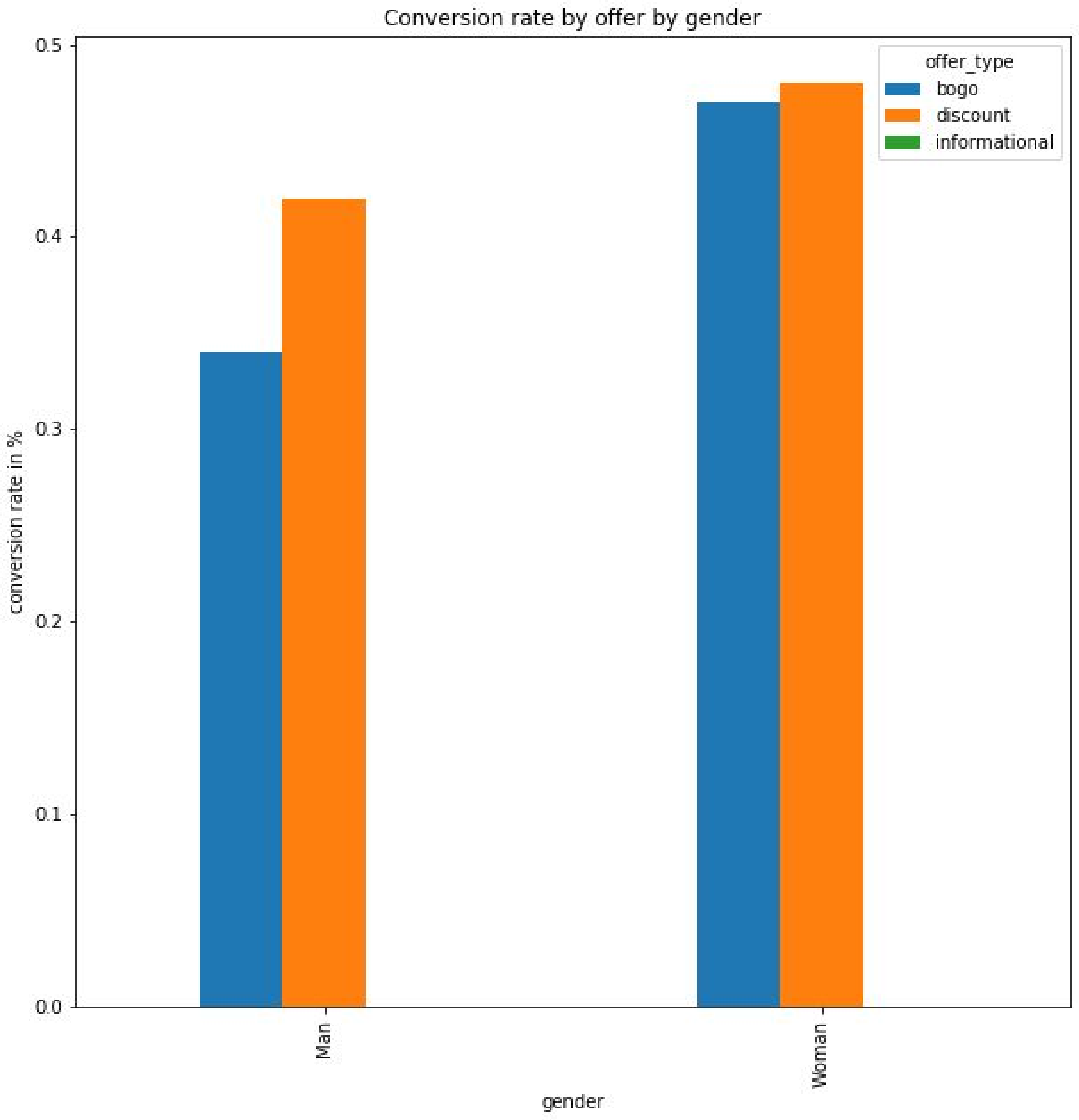


Figure 2.4: Conversion rate by offer by gender

An interesting difference to note in conversion rate is in figure 2.5. In figure 2.5 there is the conversion rate by offers split by income, as seen here people with higher income have a higher conversion on the bogo offer type, while people with less income prefer to discount. In the figure there is a pattern seen, the higher the income the higher the conversion rate for bogo, basically bogo conversion rates “correlate” to the income of a customer.

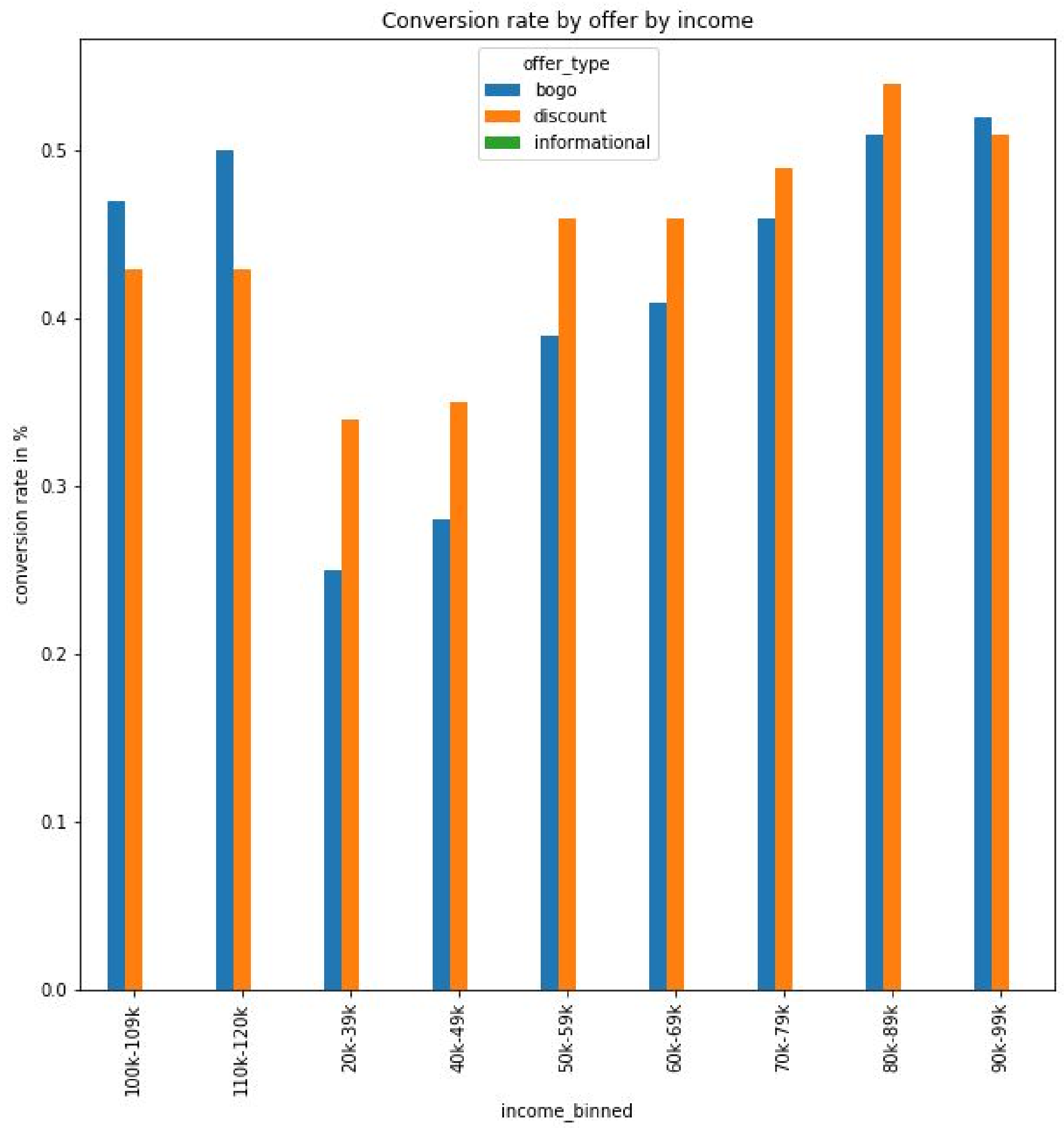


Figure 2.5: Conversion rate by offer by income

In general, there is a pattern: the higher the income the higher is the conversion rate. Figure 2.6 shows the conversion rates by income and gender. One important thing to note here, while people with the highest income have the highest conversion rate, there are not many customers in the higher income segment. Especially both segments (Man, 100k-109k) and (Man, 110k - 120k) both segments have received a low amount of offers in comparison to other segments. Since the number of people in higher-income segments are low, there is a need to target the segments carefully. Also, figure 2.6 shows that women have almost stable conversion rates across income categories. While men have fluctuating conversion rates across income categories.

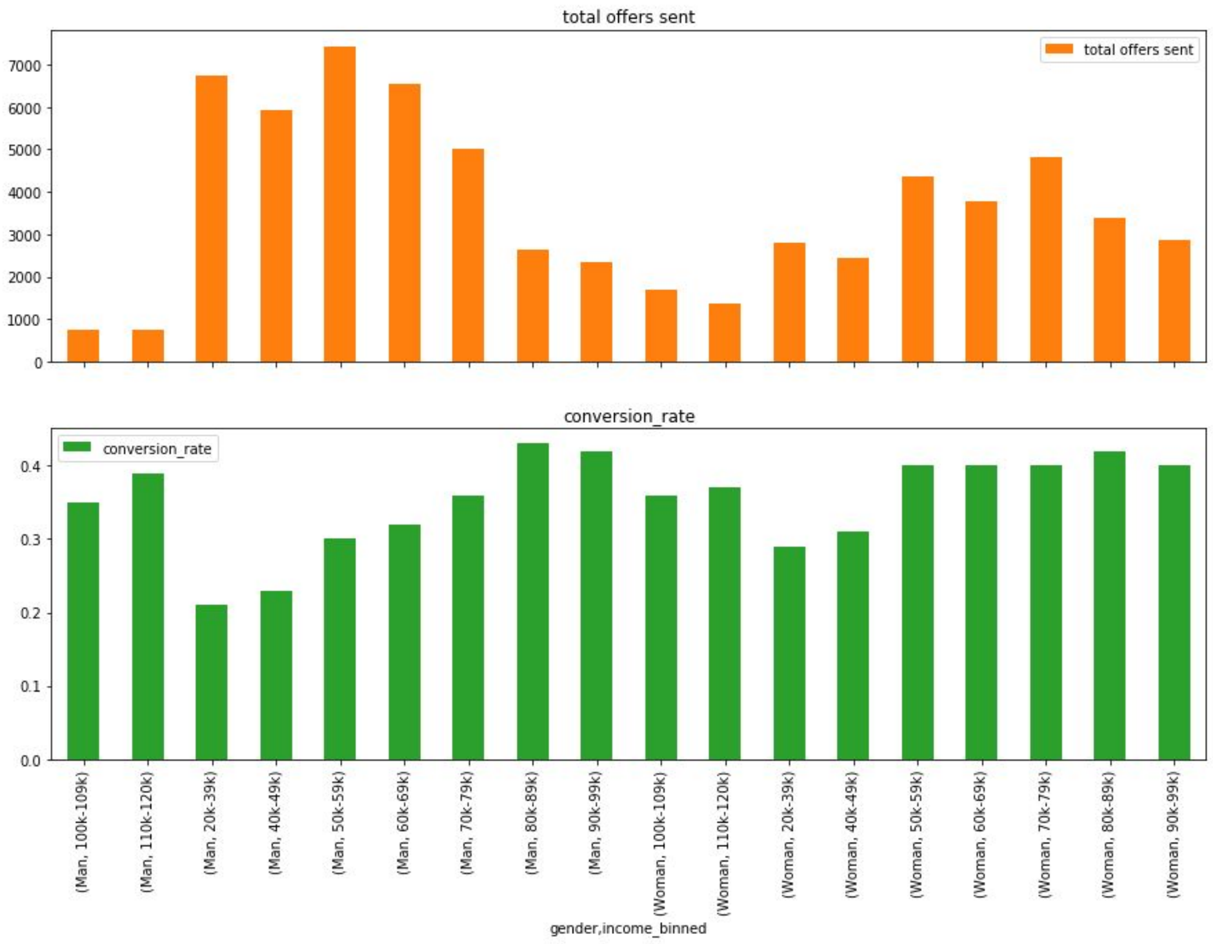


Figure 2.6: Conversion rate for offers with view by age and gender

Algorithms and Techniques

The decision was made to test different algorithms (to brute-force) to see which one will fit best. The main reason was the size of the dataset and available resources. There is a very low amount of computational resources needed to test different algorithms to make a decision. During that search of an algorithm, all tree-based algorithms showed outstanding results.

After testing several algorithms, the final decision was made to choose a stacked Random Forest Classifier. A stacked Random Forest Classifier has shown the best performance out of 10+ tested algorithms.

Benchmark

As already mentioned in the introduction part, the goal of the project was to take customer attributes and offer attributes and to predict if the offer will be successful or not. By doing so, the Starbucks team will be able to figure out which offer should be sent to which person.

As the main metric the accuracy score was chosen, but also the recall score plays an important role here. The recall score shows how many of the offers that were successful, were actually recognized as successful. The higher the recall score the better the chance to target all customers that will be able to convert. While precision score can be neglected here because sending a wrong offer to the wrong person is not a problem, to make an error here is not an issue for the company.

As a benchmark for both metrics accuracy and recall a Naïve classifier was created. The Naïve classifier marks all results as successful offers, a Naïve classifier acts as a random classifier.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

Here are some high-level data preprocessing steps:

1. Removing rows with the age of 118 years from the profile dataset
2. Removing rows gender O, since not enough observations will make the algorithms perform poorly
3. Removing rows were gender, age or income is missing
4. Combining data from income and age columns into categories for better analysis
5. Combining transcript and portfolio data to group offer attributes into a single data frame
6. Calculation of aggregated metrics for the amount and if the offer was completed via the view of the offer or not
7. Merging offer data with the customer profile data to have a single source of truth
8. Dropping all rows that contain other information than the offer received, because the offer received has all aggregated metrics about the offer, including if successful or not and amount spent.

Implementation

The next step in the project was to decide which feature should be used and which features should be dropped. There were also features created for better visualization such as income binned and age categories. All the features not needed for machine learning algorithms were dropped during the current step. Following features were included in the final data frame:

- reward
- spend
- duration
- channels: email, mobile, social, web
- age
- income
- membership year
- gender
- offer type
- offer success status

Important to note here is that the 'amount' and 'offer type informational' column were dropped from the data frame too. The amount plays an important role in classification, but since it's a posterior value, it cannot be used for prediction. The amount is known afterward not before sending the offer. The 'offer type information' has no offer completions at all, therefore it does not make sense to make a prediction. The rest of the offer types and gender were one hot encoded as well as channels that were one hot encoded in the previous parts of the project. One important issue to note is that the dataset is highly imbalanced. Figure 3.0 shows the imbalance between successful and not successful offers. There are many not successful offers.

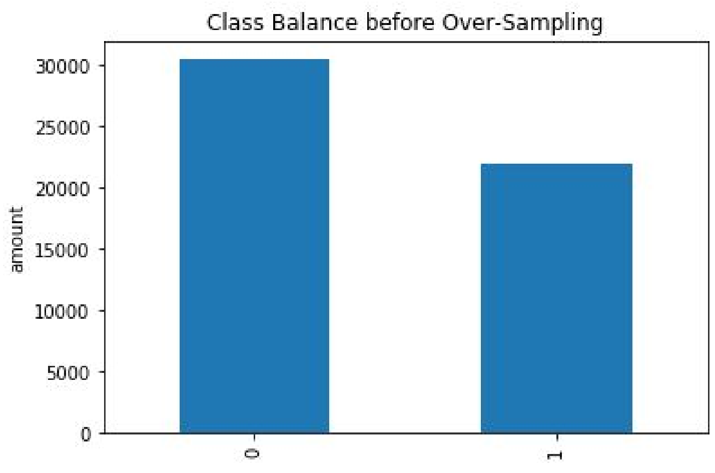


Figure 3.0: Class balance before oversampling

The decision was made to rebalance the dataset with the help of oversampling technique. Figure 3.1 shows the class balance after the oversampling. The oversampling step helped to dramatically improve the performance of tree-based algorithms.

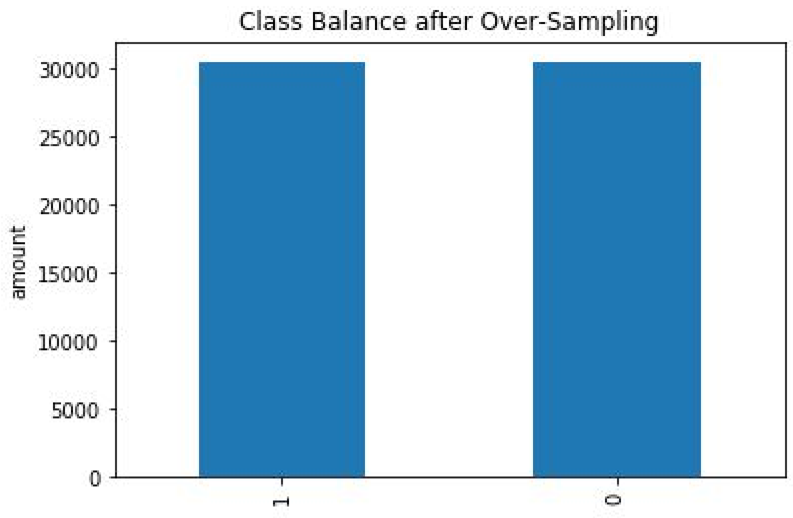
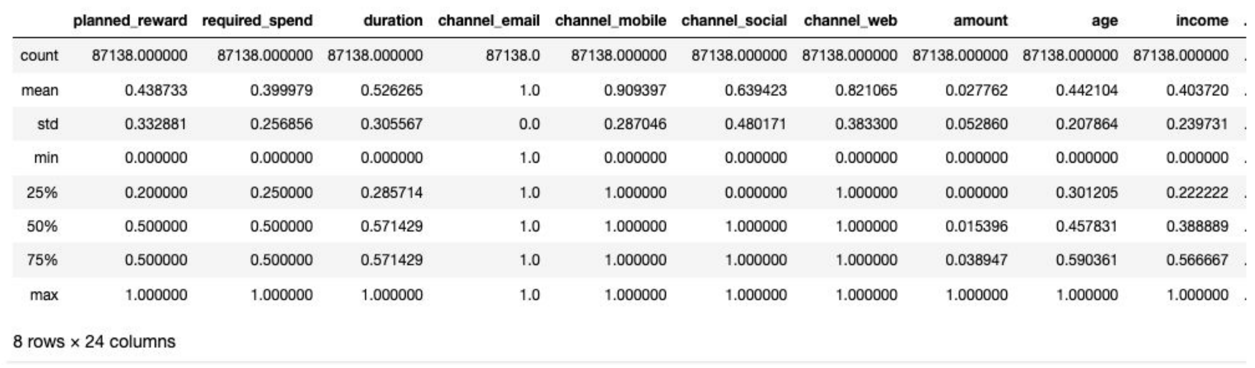


Figure 3.1: Class balance after oversampling

Many of the columns in the dataset had different numerical values. The MinMaxScaler was used to bring all features to the same scale. Feature scaling is an important step for algorithms that calculate the distance between the data points. MinMaxScaler was applied on the following features:

- reward
- spend
- duration
- age
- income
- membership year

Figure 3.2 shows some of the columns of the final data frame before algorithm selection and model training.



	planned_reward	required_spend	duration	channel_email	channel_mobile	channel_social	channel_web	amount	age	income	.
count	87138.000000	87138.000000	87138.000000	87138.0	87138.000000	87138.000000	87138.000000	87138.000000	87138.000000	87138.000000	.
mean	0.438733	0.399979	0.526265	1.0	0.909397	0.639423	0.821065	0.027762	0.442104	0.403720	.
std	0.332881	0.256856	0.305567	0.0	0.287046	0.480171	0.383300	0.052860	0.207864	0.239731	.
min	0.000000	0.000000	0.000000	1.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	.
25%	0.200000	0.250000	0.285714	1.0	1.000000	0.000000	1.000000	0.000000	0.301205	0.222222	.
50%	0.500000	0.500000	0.571429	1.0	1.000000	1.000000	1.000000	0.015396	0.457831	0.388889	.
75%	0.500000	0.500000	0.571429	1.0	1.000000	1.000000	1.000000	0.038947	0.590361	0.566667	.
max	1.000000	1.000000	1.000000	1.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	.

8 rows × 24 columns

Figure 3.2: Data frame after feature scaling

The split in test and training size is 0.2 for the test and 0.8 for training. At first, the Naïve classifier was running to establish the benchmark against which all other algorithms will be compared. Then some randomly chosen algorithms were tested:

- LogisticRegression
- LinearDiscriminantAnalysis
- KNN
- DecisionTreeClassifier
- GaussianNB

Already during the first step, it was clear that a tree-based algorithm outperforms all other algorithms. Next some ensembles such as Ada Boost, Random Forest, Gradient Boosting and XGBoost were tested. Random Forest showed here the best results. After testing ensembles the decision was made to test the stacking approach. The idea behind stacking is to use first-level models to make predictions and then use these predictions as features to the second level models

The stacking approach was built from the first level models:

- DecisionTreeClassifier
- RandomForestClassifier
- XGBoostClassifier
- AdaBoostClassifier
- GradientBoostingClassifier

The second level model was Random Forest. The previous accuracy score from Random Forest (where it was running as a standalone algorithm) was increased by almost 2%.

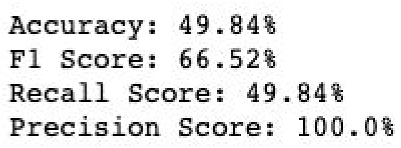
Refinement

After getting the initial results from Random Forest Classifier a grid search was used to optimize the hyper-parameters. But it seems that the default hyper-parameters of Random Forest perform already well. Grid Search didn't help to improve the metrics. The stacking of tree-based algorithms was a refinement of the model itself. It helped to increase all metrics across the board. Also here running grid search didn't help to improve the metrics.

IV. Results

Model Evaluation and Validation

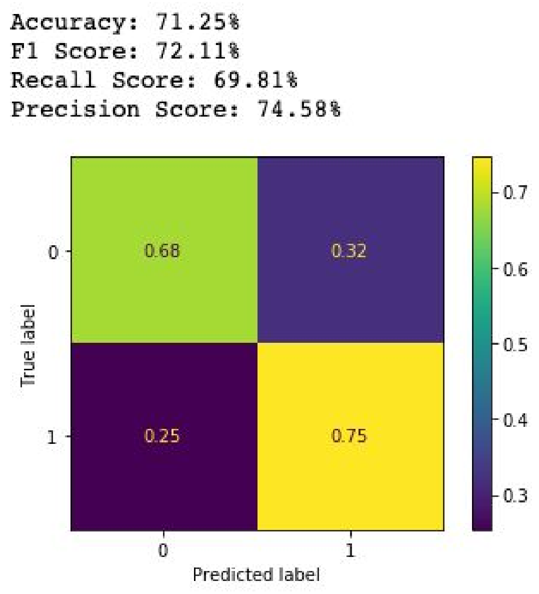
As already mentioned as a benchmark model a Naïve classifier was created. Figure 4.0 shows the results on the test set after running the Naïve classifier. All labels were set to success, basically a random model. From now on every algorithm was compared to a Naïve classifier's results.



```
Accuracy: 49.84%
F1 Score: 66.52%
Recall Score: 49.84%
Precision Score: 100.0%
```

Figure 4.0: Naïve Classifier Scores

During the next step, different algorithms via a brute-force method were analyzed. After validating different algorithms, the tree-based algorithms showed the best results. The next step was to choose ensembles to see if the results from the previous steps can be improved. Figure 4.1 shows the metrics from the Random Forest Classifier.



```
Accuracy: 71.25%
F1 Score: 72.11%
Recall Score: 69.81%
Precision Score: 74.58%
```

Figure 4.1: Random Forest Results

An additional run of grid search didn't help to lift the metrics. The decision was made to use stacking to see if the results can be improved further. Figure 4.2 shows the overview of the stacking and algorithms included in the process. Again, Random Forest was chosen as the second layer a so-called meta learner.

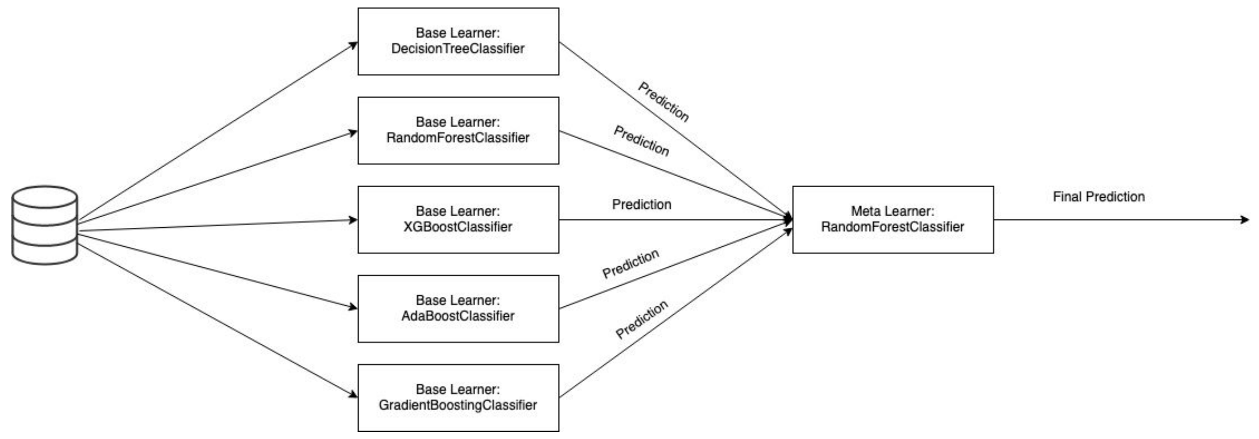

Base Learner: DecisionTreeClassifier
Base Learner: RandomForestClassifier
Base Learner: XGBoostClassifier
Base Learner: AdaBoostClassifier
Base Learner: GradientBoostingClassifier
Prediction
Prediction
Prediction
Prediction
Prediction
Meta Learner: RandomForestClassifier
Final Prediction

Figure 4. 2: Stacking Overview

Figure 4.3 shows the final results after training and evaluating the stacking model. In order to avoid overfitting and provide generalization, a four-fold validation approach was used for training of the stacked model. The test results were generated on previous unseen test data.

```
Accuracy: 73.05%
F1 Score: 74.55%
Recall Score: 70.41%
Precision Score: 79.22%
```

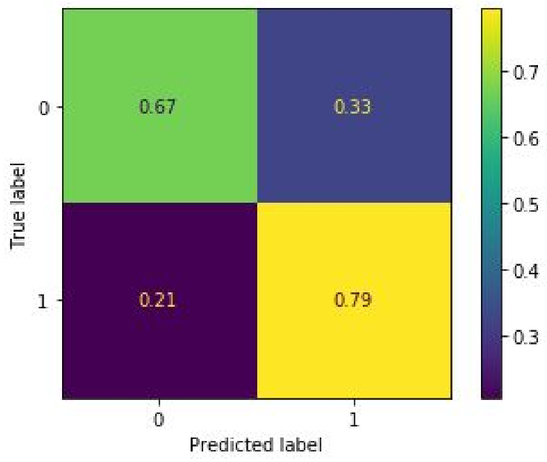


Figure 4.3: Stacking Random Forest Results

Again, running grid search didn't help to improve the metrics, it seems that Random Forest has already very well defined default hyper-parameters. One important aspect to note is that the most predictive features are age and income. Both features accumulate almost 80% of the weights. The information is very important for the customer acquisition campaigns and should be used during research and planning of marketing campaigns.

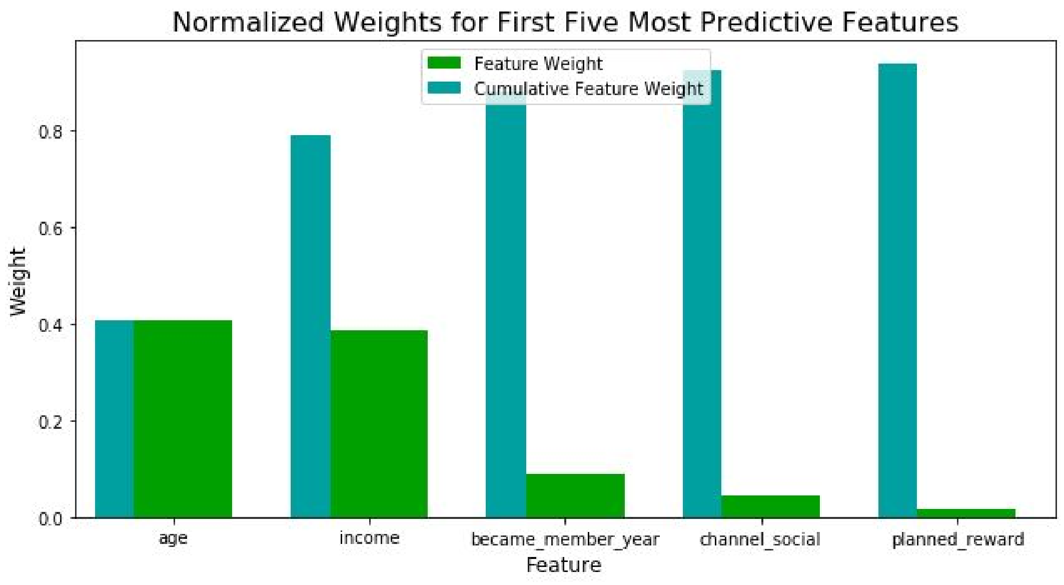


Figure 4.4: Feature Weights for Random Forest

V. Conclusion

Reflection

One of the biggest mistakes I did during the project was to start to explore the data in a freestyle modus. I was simply trying to get some insights into the data, to see if there are some interesting insights hidden, so I can decide what I want to set as a goal for the project. But the problem was that without a clear goal, there are so many ways to explore the data, so it made the whole process very cumbersome. But once I had set the goal the whole project became much more effective.

Also while investigating the algorithms I saw some flaws in the design and thinking. For example I was using amount as one of the features for the model. But the problem was that the amount is a posterior value, when you want to send a campaign you don't have the amount yet. Amount plays an important role in forecasting if the offer will be successful or not. Also, there is an offer type called informational, all informational offers don't have offer completions, so when you have informational offers as a feature, the algorithms learns fast that this offer will be not successful. While from the standpoint of prediction such features like amount and informational offers are very good, from the logical standpoint it's useless. Once amount and offer type information were removed, the results dropped by 20%. Figure 5.0 shows the results with amount and informational offer type included.

```
Accuracy: 91.58%
F1 Score: 92.04%
Recall Score: 87.45%
Precision Score: 97.14%
```

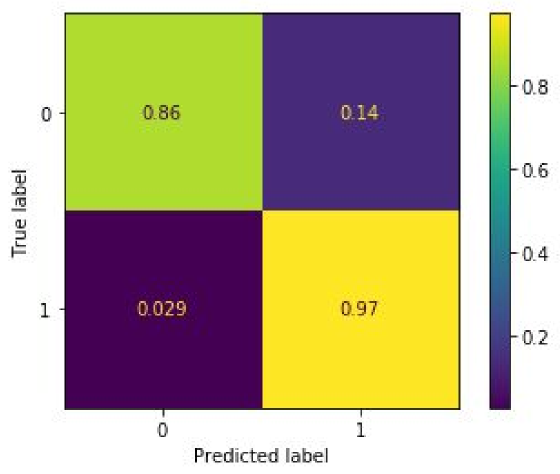


Figure 5.0: Stacked Random Forest with Amount Included

One of the most complicated parts of the project was the aggregation of metrics on an offer level. Since the offer data was a transaction log, there were a couple of modifications needed to get the metrics right. First it wasn't clear how to group the data properly, how to attribute the amount to the offers. The decision was made to use the time window from offer received, viewed and completed, every transaction that happened in between was attributed to the offer. But the problem is that some offers overlap in time, so the transactions will be double counted. It's an important point that needs further investigation. But since amount was removed for training the models, it wasn't an issue for predictions.

Improvement

I would try to run some deep learning algorithms to see if the results can be improved. The main problem was that I didn't have access to GPU. It's possible to train the model e.g. on Sagemaker, but it still very time consuming to test different algorithms, hyper-parameters and so on. I tested some deep learning algorithms on my local machine, they didn't improve the original Random Forest results, but I couldn't fine-tune the hyper-parameters because of the time constraints. It would simply take too long to figure out the right settings.