Capstone Project
Machine Learning Engineer Nanodegree

Lucas Braun
June 2020

Definition

Project Overview

The domain of this project is the exploration of customer behavior, particularly the customer behavior when presented with advertisement or product offers. It is very helpful to have an idea of how a customer will react to an offer that is presented to him or her, in order to maximize the effectiveness of the offers. An offer would be effective if a customer both views and completes the offer, as this would mean that the customer was actually influenced by the offer. Offers that are viewed and not completed or not viewed and not completed would either be ineffective or fall in the category of informational offers, as these types of offers cannot be completed. An offer that is not viewed but completed, however, is actually harmful to the business. The customer receives a discount even though the customer would have spent more money if the offer was not made. In this case, presenting an offer to a customer cannibalizes revenue. It should therefore be avoided to send such offers to a customer.
In this specific context the behavior of customers within the Starbucks rewards app will be investigated. Customers that are registered in the Starbucks Rewards app frequently get offers from the app. These offers can be purely informational or actual offers like discounts and buy-one-get-one-free offers. In total, there are ten different offers of different types and difficulties, which are given in the dataset portfolio.json.
During a test period of one month, the customer behavior of Starbucks Rewards customers was recorded. These records include the offers received by customers, the offers viewed by customers, the offers completed by customers and all transactions made by the customers. All of this information is given in the dataset transcript.json. In total there are well over 300 thousand recorded datapoints. These datapoints can be attributed to 17,000 customers. The id of each customer as well as some further information like age and gender are recorded in the profile.json dataset.
With the help of these datasets it can be analyzed how customers react to the offers they received in the Starbucks Rewards App. Based on this analysis a model can be built to help Starbucks optimize, which offer should be sent to customers and identify which offers are harmful to Starbuck's revenue.
right offer to the right customer. For every offer sent there are 4 possible outcomes based on whether the offer is viewed and completed.

Problem Statement

For this project a model will be built that predicts how a customer will react to an offer that was sent to him or her. Any offer can fall into one of four categories on the viewed/completion matrix:

	Completed	Not Completed
Viewed	1	2
Not Viewed	4	3

The value of each category can be ranked from most desirable (1) to least desirable (4). The problem investigated in this project is therefore building a model that can predict the category of the viewed/completed matrix into which an offer that is presented to a customer is most likely to fall. Based on this prediction, a simple recommendation function can be built to suggest an offer based on the attributes of a specific customer.

Metrics

The model that has to be built to solve the problem statement is a multi-class classification model. This means that the model will assign one of the four categories as an output.
The main metric that will be used to evaluate the effectiveness of the model will be accuracy. Accuracy in this case is defined as follows:

$$Accuracy = \frac{True\ Positives}{Total\ Number\ of\ Predictions}$$

True Positives are the cases in which the model correctly predicts the category of the observation. As the problem is a multi-class classification problem and not a binary classification problem, there are no true negatives – which in a binary classification problem also have to be taken into account when determining accuracy.
The Total Number of Predictions as the name suggest refer to the total amount of predictions made.

Accuracy was chosen as a metric for this problem, as the metrics of recall and precision do not make a lot of sense in this multi-class classification problem. Firstly, the recall and precision cannot be calculated as one total metric but rather has to be calculated between each of the categories. Secondly, it is not evident that the importance of either recall or precision outweighs the importance of the other. Therefore, it is helpful to take a look at these metrics to see if there are any red flags, but accuracy obvious main metric by which the model will be evaluated.

Analysis

Data Exploration

The data exploration of the initial input datasets described in the project overview does not give great insight into the problem, as the datasets have to be significantly edited before obtaining a dataset that can actually be used to solve the problem statement.
Nevertheless, it helps to understand the features of each of the initial datasets in order to understand the processed dataset that will be used to train the model. The schema and features of the three datasets are as follows:

portfolio.json
- id (string) - offer id
- offer_type (string) - type of offer ie BOGO, discount, informational
- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) - time for offer to be open, in days
- channels (list of strings)

profile.json
- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str) - customer id
- income (float) - customer's income

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

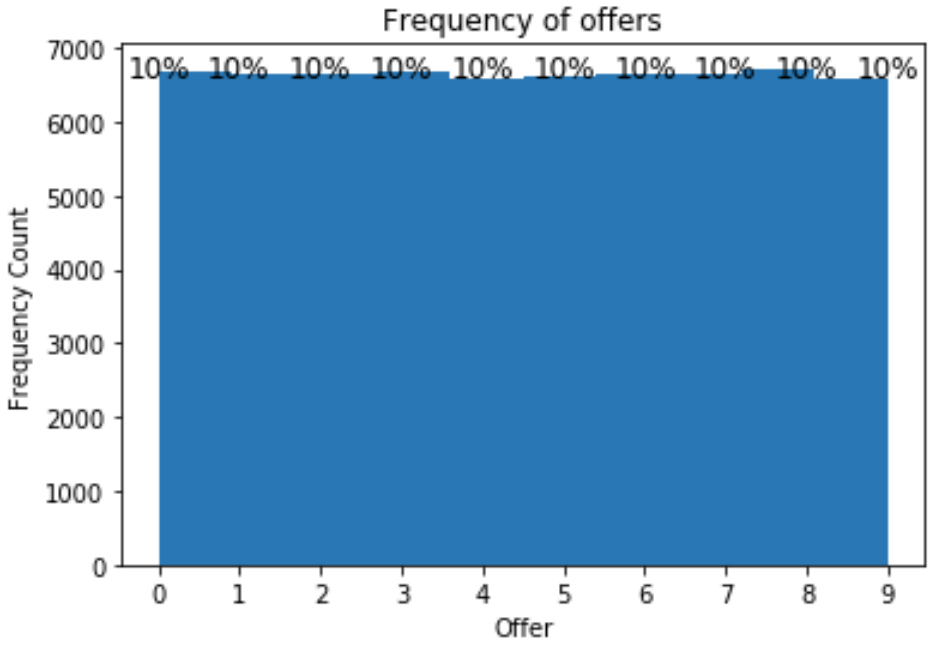
In the following the features of the transformed dataset will be outlined and explained. To be able to follow along with this part it is suggested to first scim over the ‘Data Preprocessing’ section in the ‘Methodology’ chapter.
The transformed dataset used to train the model records each offer that was received by a customer and whether this offer was viewed and completed. The label corresponds to the categories defined in the viewed/completed matrix above. Additionally, for every offer it holds all the features of the offer from the portfolio dataset and all the features of the customer the offer was sent to from the profile datasets (except the became_member_on feature). All features were then transformed to be either binary or numerical. This results in a dataset with a total of 32 columns (including the label) and 66,501 rows.
The feature list reads as follows:

Transformed_offer_df:

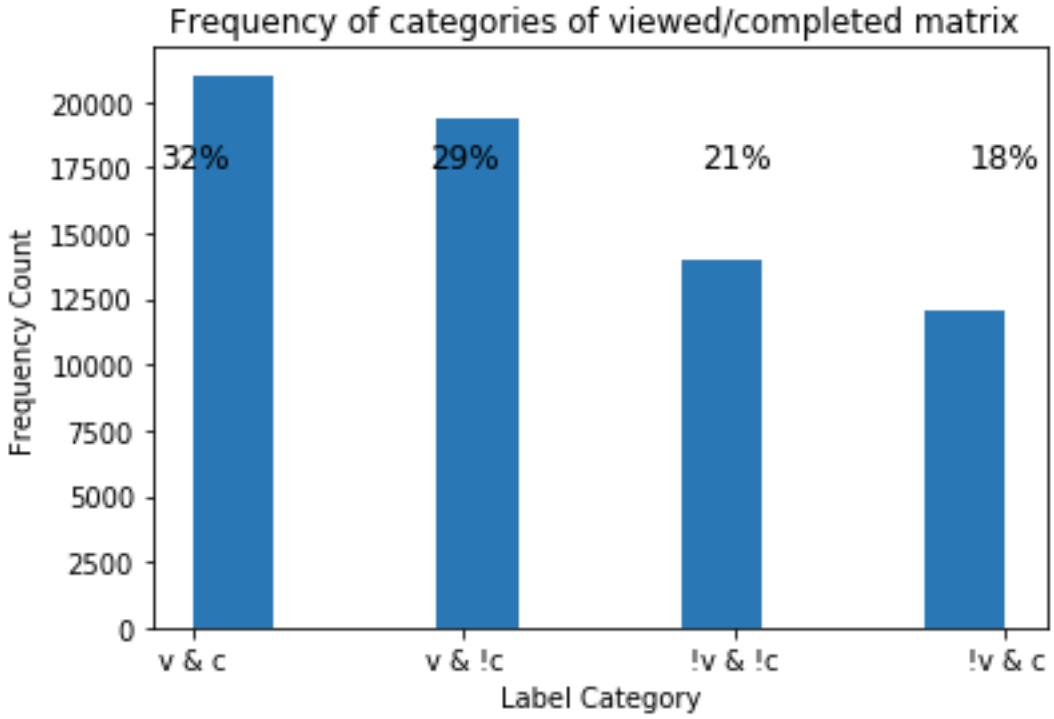
- label – (int) category of the viewed/completed matrix the observation falls into (0,1,2,3)
- cust_age – (float) scaled age of customer
- cust_income – (float) scaled income of customer
- cust_spend_wk_1 – (float) scaled spending volume in week one of test period
- cust_spend_wk_2 – (float) scaled spending volume in week two of test period
- cust_spend_wk_3 – (float) scaled spending volume in week three of test period
- cust_spend_wk_4 – (float) scaled spending volume in week four of test period
- offer_channel_web – (int) binary variable, 1 if offer is sent via web
- offer_channel_email – (int) binary variable, 1 if offer is sent via email
- offer_channel_mobile – (int) binary variable, 1 if offer is sent via mobile
- offer_channel_social – (int) binary variable, 1 if offer is sent via social media
- offer_type_bogo – (int) binary variable, 1 if offer type is buy-one-get-one-free
- offer_type_discount – (int) binary variable, 1 if offer type is discount
- offer_type_informational – (int) binary variable, 1 if offer type is informational
- offer_reward_0 – (int) binary variable, 1 if offer reward is 0
- offer_reward_2 – (int) binary variable, 1 if offer reward is 2
- offer_reward_3 – (int) binary variable, 1 if offer reward is 3
- offer_reward_5 – (int) binary variable, 1 if offer reward is 5
- offer_reward_10 – (int) binary variable, 1 if offer reward is 10
- offer_difficulty_0 – (int) binary variable, 1 if offer difficulty is 0
- offer_difficulty_5 – (int) binary variable, 1 if offer difficulty is 5
- offer_difficulty_7 – (int) binary variable, 1 if offer difficulty is 7
- offer_difficulty_10 – (int) binary variable, 1 if offer difficulty is 10
- offer_difficulty_20 – (int) binary variable, 1 if offer difficulty is 20
- offer_duration_3 – (int) binary variable, 1 if offer duration is 3
- offer_duration_4 – (int) binary variable, 1 if offer duration is 4
- offer_duration_5 – (int) binary variable, 1 if offer duration is 5
- offer_duration_7 – (int) binary variable, 1 if offer duration is 7
- offer_duration_10 – (int) binary variable, 1 if offer duration is 10
- cust_gender_F – (int) binary variable, 1 if customer gender is Female
- cust_gender_M – (int) binary variable, 1 if customer gender is Male
- cust_gender_O' – (int) binary variable, 1 if customer gender is Other

Exploratory Visualization

There are two features of the dataset that are interesting to investigate before proceeding. Firstly, it is important to see how often each of the 10 offers of the portfolio.json dataset were sent to customers. The desirable property here is that the 10 offers are sent out with equivalent frequency. From the transformed_offer_df dataset a graph can be constructed that nicely shows that this property is fulfilled.



As can be seen from the graph each offer appears in 10% of the observations of the dataset. This means that there is no imbalance that has to be accounted for.
The second feature characteristic that is interesting to investigate is the frequency of the categories of the viewed/completed matrix, i.e., the label of the observation. If the frequencies are strongly imbalanced this would have to be accounted for. The following graph visualizes the frequencies of the labels in the dataset.



It can be seen that the category viewed & completed is the most frequent category with a share of 32%. This is a nice property, as this is the most desirable outcome. The category viewed &

completed has a share of 29% and the category not viewed & not completed has a share of 21%. This shows that a majority of offers fall into categories that are not harmful to Starbucks revenue. However, a share of 18% falls into the category not viewed & completed, which is still quite high considering that Starbucks loses potential profit with each of the roughly 12000 sent offers that fall into this category.
Even though the categories are slightly imbalanced, the imbalance is not great enough to cause concern for using the dataset to train a supervised learning model to solve the problem.

Algorithms and Techniques

The problem statement and the created dataset clearly show that the model used to solve this problem has to be a supervised learning classification model. Since the dataset includes both binary and numeric variables a Naïve Bayes Classifier should not be used. Of the frequently used supervised learning classification algorithms, this leaves the options of a linear learner, a decision tree and a neural network. All of these will be tested as part of this project. The model that performs best will be used in a sample recommendation function.
As a linear learner, the Sagemaker built-in algorithm will be used. This algorithm uses stochastic gradient decent to find the optimal linear functions to split the dataset into the four classification categories. This algorithm works best for datasets with few features that do not have polynomial properties. For the dataset used in this problem, these properties are likely to be fulfilled. While there are 31 input_features, most of them refer to the features of the offer and it is likely that a simple algorithm like a linear learner will be able to do a decent job of categorizing the labels. Therefore, the linear learner is used as somewhat of a second benchmark to see how a simple model compares to the decision tree and neural network.
As decision tree classifier, the Sagemaker built-in algorithm XGB Boost will be used. XGB boost is an algorithm which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker decision tree models. A decision tree tries to classify an observation based on the values of the features of the observation. Simplified this looks something like: if feature 1 > 0.5 then assign label = 1 and otherwise assign label = 0. As there are a variety of input features it is likely that the XGB Boost algorithm will be better than the linear learner as it might be able to identify more nuanced differences in the assignments of labels.
As neural network, a custom Multi-Classification Network will be built using Pytorch. The neural network will have 31 input nodes and 4 output nodes. Each note corresponds to one of the categories in the viewed/completed matrix. The loss function that will be used is Cross Entropy loss. The output of the neural network can be evaluated using a softmax function and returns the probabilities with which the network would assign each of the four categories. The predicted label is then the node that delivers the highest probability.

Benchmark

As a benchmark for the model a random assignment of the categories from the viewed/completed matrix was chosen. Based on the Exploratory Visualization it was shown that the labels occur with frequencies of 32% - label = 0; 29% - label = 1, 21% - label = 2; 18% - label = 3. Based on these frequencies the benchmark model randomly assigns a label to the test dataset. As this is likely to

be the current process of how the offers are sent, this approach seems like an appropriate benchmark.
As a result, the benchmark model should get about 1 in 4 label assignments right and therefore have an accuracy of 25%. It turns, out that the accuracy of this approach used on the test set is even slightly higher at 27%. Nevertheless, this is not a very good accuracy and should definitely be improvable with an appropriate ML model.

Benchmark Model Accuracy = 27%

Methodology

Data Preprocessing

The data preprocessing in this specific case encompasses the creation of an entirely new dataset using the three given datasets.
The first step is to rid the portfolio, profile and transcript datasets of any incomplete rows. Only, the profile dataset actually has such observations, however, when deleting a row of the profile dataset all the corresponding observations in the transcript dataset also have to be deleted. In total, this data cleaning set encompasses deleting 2175 entries from the profile dataset and 33772 entries from the transcript dataset.
In the second step, the transcript dataset will be used to create a dataset that holds a record of each offer that was received by a customer and whether this specific offer was viewed and completed. Here it is important to account for the fact that customers can view offers after they complete them and therefore, an offer can only count as viewed if it was viewed before completion. This preprocessing step yields a dataset that holds the customer id and offer id of each offer that was received by a customer and an entry for whether it was viewed and completed. In total there were 66,501 offers sent to customers. From this dataset each offer can be assigned with its corresponding category from the viewed/completed dataset.
In the third step, the new dataset is extended by adding the offer information from the portfolio dataset to each of the 66,501 entries of the dataset.
In the fourth step, the customer information is extended. From the transcript dataset it can be constructed how much each customer spent in each week of the test period. Therefore, four variables holding the transaction amount of a customer in the weeks 1 to 4 is added to the profile dataset.
In the fifth step, the customer information is added to the offer dataset and each of the 66,501 entries of the dataset is extended by the available information about the customer, to whom the offer was sent.
In the sixth step, the id columns are deleted from the dataset and all features are either scaled if they are numeric or one-hot-encoded using the pd.get_dummies function.
This results in the transformed_offer_df dataset described in the section ‘Data Exploration’ of the ‘Analysis’ chapter.

The final step in the data preprocessing is splitting the transformed_offer_df into training and testing sets and uploading the datasets to S3.

Implementation

For the implementation of the linear learner the Sagemaker built-in algorithm was used. The specification looks as followed:

```
# define Linear Learner
from sagemaker import LinearLearner
output_path = 's3://{}/{}'.format(bucket, prefix)

ll_estimator = LinearLearner(role=role,
                             train_instance_count=1,
                             train_instance_type='ml.c4.xlarge',
                             predictor_type='multiclass_classifier',
                             num_classes = 4,
                             output_path=output_path,
                             sagemaker_session=sagemaker_session,
                             epochs=15)
```

The implementation is pretty straight forward, yet there are a couple of things to look out for. Firstly, the *predictor_type* has to be set to *'multiclass_classifier'* and the number of classes has to be specified as *num_classes = 4*. This lets the linear learner know that there are four possible output labels.
Secondly, the input data has to be formatted in a record set format, which is done before training the algorithm.

For the implementation of the XGB Boost algorithm the Sagemaker built-in algorithm was used. The specification looks as followed:

```
#get container name
from sagemaker.amazon.amazon_estimator import get_image_uri
container = get_image_uri(sagemaker_session.boto_region_name, 'xgboost')

#create hyperparameters and other input data necessary
hyperparameters = {"max_depth":"7",
                   "eta":"0.2",
                   "gamma":"4",
                   "min_child_weight":"6",
                   "objective":"multi:softmax",
                   "num_class": "4",
                   "subsample":"0.8",
                   "silent":"0",
                   "early_stopping_rounds" : "10",
                   "num_round":"500"}

output_path = 's3://{}/{}/output/xgb'.format(bucket, prefix)
```

```
#create XGB estimator and train estimator
xgb = sagemaker.estimator.Estimator(container,
                                    role,
                                    hyperparameters=hyperparameters,
                                    train_instance_count=1,
                                    train_instance_type= 'ml.m5.2xlarge',
                                    output_path=output_path,
                                    sagemaker_session=sagemaker_session
                                   )


s3_input_train = sagemaker.s3_input(s3_data=train_location, content_type='csv')
xgb.fit({'train': s3_input_train})
```

The implementation of the XGB Boost algorithm requires the specification of a number of hyperparameters. Particularly, important is that the *objective* is set to '*multi:softmax'* and the number of classes is set to *num_class = 4.* The other parameters have to be tested to see what values deliver the best output.
For the training of the XGB Boost algorithm the data stored in s3 in form of a csv file was used directly as the format is compatible with the algorithm input.

For the implementation of the Neural Network, a custom pytorch implementation was chosen. This encompasses that the model architecture and training is specified in separate python files model.py and train.py. For this particular task a Neural Network with 4 layers was chosen. An input layer with 31 input features, a hidden_layer_1 with 128 nodes, a hidden layer_2 with 64 nodes, a hidden_layer_3 with 32 nodes and lastly an output layer with 4 nodes – one for each of the possible output labels. It is likely that this Network is overly complex for the problem at hand, however, this design was chosen to see if the neural network is able to pick up nuances in the data that the two other models were not able to find.

Refinement

During the refinement of the implementation the accuracy was improved by:
Linear Learner:

- changing the number of epochs of the linear learner

XGB Boost:

- Changing the hyperparameters of the model such as max_depth or num_rounds

Neural Network:

- Changing the model architecture, i.e, adding a layer and changing number of nodes
- Changing the epochs and the learning rate

Results

Model Evaluation and Validation
The trained models achieved the following accuracies:

Benchmark Model Accuracy = 27%
Linear Learner Model Accuracy = 59.4%
XGB Boost Model Accuracy = 63.8%
Neural Network Accuracy = 61%

While the models perform significantly better than the benchmark model, the accuracy does not seem to go higher than 64%, regardless of how the three models are changed. While this is a very significant improvement to the benchmark, it seems like there is still some untapped potential.

This is why a second step of the model evaluation was testing how the models faired when actually used in a recommendation function. For this purpose, a function was written which predicts the

category of the viewed/completed matrix for each of the 10 offers given a specific customer_id from the profile dataset.
The results of this function was that all of the models learned to identify the offer that was sent to them and almost always returned the same label for a specific offer, regardless of who the customer is. This did not change even when adjusting the input dataframe to include much less information about the actual offer.
An example of how the assignment to the different labels looks:

	reward	difficulty	duration	offer_type	id	channel_web	channel_email	channel_mobile	channel_social	label
6	2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1	1	0
7	0	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	0	1	1	1	1
1	10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1	1	1
8	5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1	1	1
0	10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	0	1	1	1	1
5	3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1	1	1
2	0	0	4	informational	3f207df678b143eea3cee63160fa8bed	1	1	1	0	2
9	2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	1	0	2
4	5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	1	0	0	2
3	5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	1	0	3

It can be seen that one offer is most likely to fall into category 3 – the harmful category.

While the behavior of the models seems detrimental to the promise of having personalized recommendation based on the category of the viewed/completed matrix, this has a very significant and immediate business impact, which will be explained in the justification section.

Justification

Given the benchmark model, the best model of this project (XGB Boost) performed significantly better. Therefore, it is to some extend successful in completing the objective of the project.
However, a more significant finding of the project is an unexpected one. Namely that the model Does not have a custom recommendation for each customer but rather seems to make recommendations for an offer as a whole. The algorithm can actually show which offers are particularly vulnerable to fall into the harming category – not viewed and completed. Therefore, there is a very significant finding, that Starbucks could probably very easily improve its revenue by discontinuing some of the offers, which have a high likelihood of being classified in the harmful category.
Therefore, the model is not only able to significantly improve upon the accuracy of the benchmark model (the XGB Model is 37 percentage points better than the benchmark), but it is also able to give concrete business insight on how effective the offers are.

Conclusion

Reflection

Reflecting on the project it was shown that the approach of categorizing the offers made to customers based on the viewed/completed matrix was somewhat successful. The major finding of this project, however was that this approach is not great for making personalized recommendation, but very useful for analyzing the effectiveness of an offer. The models clearly showed which offers are likely to be harmful to the profit of Starbucks because they unnecessarily take away revenue. Therefore, even though the result was different than expected, from a data analysis standpoint it was still a highly valuable result.

Improvement

The project could be improved on two fronts.
Firstly, the dataset could be adjusted to actually include all the customers that did not provide all the customer information. There is potentially a way to include them, for example by first clustering the customers using k-means and using the cluster the customer falls into instead of the more detailed information.
Secondly, the two informational offers do not quite comply to the viewed/completed matrix as defined above, as they cannot be completed. Therefore, these offers could be separated from the other offers and analyzed separately. This might allow for a more nuanced recommendation function.