Starbucks's Capstone Challenge Report

Ferenc Török
ferike.trk@gmail.com
github.com/ferenctorok/starbucks_capstone_challenge

April 29, 2020

Table of Contents

1 Definition

This project was carried out as a part of Udacity's Machine Learning Engineer Nanodegree.

1.1 Project Overview

The Starbucks Corporation is an American multinational coffeehouse chain. The company was founded in 1972 and today operates in more than 30000 locations in 77 countries worldwide. As other companies, Starbucks keeps in touch with its costumers mostly via its marketing system. It sends promotions, advertisements to the costumers via various channels. With the amount of data gathered in recent times and with the advance of Machine Learning techniques and Artificial Intelligence a new era in advertising raises. This is the era of personal advertisement where based on data about costumers, one is able to send relevant and only the most relevant promotions and advertisement to them. This stimulates purchasing and hence the company is able to gain larger profit.

The dataset given is a "toy" dataset that mimics Starbucks's own gathered during their marketing campaigns. It was created by simulation and mimics purchasing decisions and how those decisions are influenced by promotional offers in a 30 day period. Each person in the simulation has some hidden traits that influence their purchasing patterns. In the simulation people produce various events, including receiving offers, opening offers, and making purchases.

Of course in real life, Starbucks has a wide variety of products to offer, however for the sake of simplicity the simulation includes only one product. There are however 10 types of promotions about this one product, such as 'BOGO' (buy-one-get-one) or simple advertisement. The dataset contains records of some basic personal data of the costumers, the transactions with time-stamped data (transactions, when did the costumer receive an offer, when did he view it etc.) and the details of the offers. Detailed explanation of the dataset can be found later.

1.2 Problem Statement

Carrying out successful marketing campaigns is an absolutely not straight forward and vastly expensive task. Hence for Starbucks, and for any company, it is crucial to minimize the chance for campaigns to be unsuccessful. These days technology provides tools for marketing which if well used can increase the number of successful campaigns by allowing to target different people with different materials. This is a huge opportunity, however it poses further tasks to be solved. For example to identify costumer groups to target with different campaigns. Machine Learning techniques tend to play an important role in these tasks due to their capability of recognizing patterns in data. Machine Learning techniques and the vast amount of available data together provide the opportunity to carry out personal advertisement with high success rates.

The aim of personal advertisement is to send each costumer promotions which are the most probable to make them satisfied. Also it is important not to send them promotions which will most probably not interest them since these might even have a negative effect on the consumption. Also in some situations the people fulfill some promotions without even noticing that they existed. These are also situations to avoid since in this situation the company gave these people some discount although it was not necessary.

In this respect the aim of the project is to classify an offer that is given to a costumer into one of the following categories:

- Will not be viewed nor completed accidentally
- Will not be viewed but will be completed
- Will be viewed but will not be completed
- Will be completed and viewed

For being able to classify offers sent to a person, personal data and consumer history until the point of receiving the offer are going to be used. Hence one important task of the solution is to extract valuable information form the raw dataset which can be useful for representing the given costumer and his/her habits, traits.

After an offer is classified into the groups above one can decide to send or not to send that particular offer to the costumer. A reasonable choice would be to send someone an offer if it falls into the category 4 and not to send otherwise. Another option could be to also send it if it falls into 3 and 4. In this case it is not sure that the costumer will fulfill the offer, however it has viewed the offer, so at least the information got through.

After creating the training data and training the model we hope to achieve a prediction accuracy around 65% - 70%.

1.3 Metrics

As the task is multiclass classification, accuracy is the suitable performance measure. (False positive, False negative etc. is not defined for the multiclass case.) The accuracy is going to be calculated with the usual formula:

$$Accuracy = \frac{\sum_{i=1}^{N} \mathbb{I}\left(\hat{y}(x) = y(x)\right)}{N}$$

where $\hat{y}(x)$ and $y(x)$ are the predicted and real class respectively, N is the number of data points and $\mathbb{I}(statement)$ is the indicator function which is 1 if *statement* is true and 0 otherwise.

2 Analysis

This section introduces the used dataset in detail. 2.1 subsection introduces the structure of the provided dataset. In 2.2 we provide a thorough insight into the data mainly through visualization. Here the features and anomalies which will have to be cured are shown. In 2.3 we provide the set of algorithms which will be used for this purpose and finally in 2.4 we provide a benchmark model, with which our solution is going to be compared.

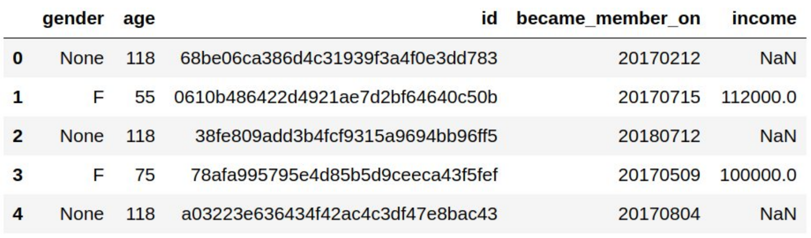
2.1 Data Structure

The provided dataset is stored in 3 '.json' files: 'profile.json', 'portfolio.json' and 'transcript.json'.

profile.json

The profile.json file contains personal data about costumers in the following fields (17000 costumers):

- 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)



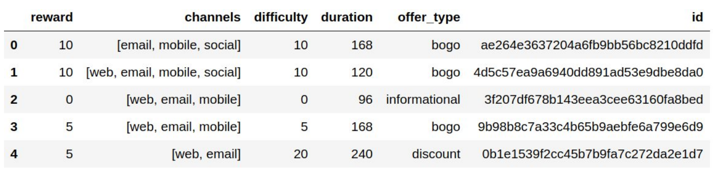
	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
4	None	118	a03223e636434f42ac4c3df47e8bac43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

Figure 1: Sample from the portfolio dataset

portfolio.json

The portfolio.json file contains the data of the offers sent during the test period in fields (10 offers):

- reward: (numeric) money awarded for the amount spent
- channels: (list) web, email, mobile, social
- difficulty: (numeric) money required to be spent to receive reward
- duration: (numeric) time for offer to be open, in days
- offer_type: (string) BOGO, discount, informational
- id: (string/hash)



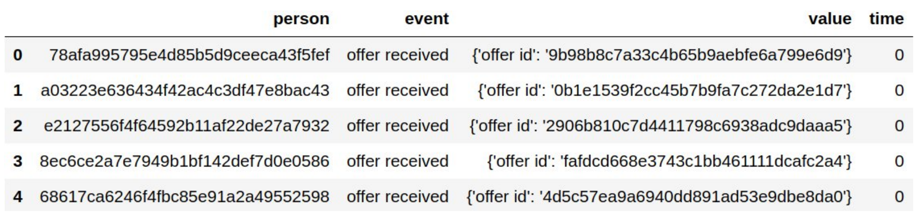
	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	168	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	120	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	96	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	168	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	240	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7

Figure 2: Sample from the profile dataset

Transcript.json

The transcript.json file contains timestamped data about transaction and offers in the following fields ((306648 events):

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

Figure 3: Sample from the transcript dataset

This data in itself is not very informative, so serious effort has to be made to extract meaningful features from it. The feature extraction / engineering is going to be shown in section 3

2.2 Data Exploration

Dealing with missing values

As it can be seen in Figure 1, the portfolio dataset is not complete. By examining the data we have established that there are two kinds of data points: One in which all data is provided and one in which gender, age and also income is missing. This can be due to several reasons, one of which is that these people did not agree with the company’s privacy policy.

A total number of 2175 datapoints are effected by missing values. One have to deal with these datapoints. A potential solution would be to discard these datapoints from the dataset. However these people possibly form an interesting group in the dataset, so we decided to keep these datapoints as well. These datapoints will be clearly differentiated from the other datapoints by giving them a gender 'U' (unknown). The age and income values will be filled with the respective mean values. The unknown gender category seems to be a reasonable choice for the following reason: Filling up the missing values with the respective means seriously changes data distribution. By the extra gender group we hope to separate these points from all the other ones sufficiently.

Profile Data Exploration

There are 8484 Male and 6129 Female datapoints in the dataset. In this respect the number male and female datapoints is quite well balanced, so no care has to be taken on compensating it in any direction. The number of people who classify themselves into other genders is however very small. These people are decided to be kept in the dataset but we assume that the predictions on them will not be very accurate.

The membership length is calculated for the people counting from the date of the last person who got a membership. This is an arbitrary choice, however since we are only interested in the relative lengths of the memberships (the data is going to be normalized / standardized) it works totally fine. After this, the age, income and membership length distribution of the costumers can be analyzed. These distributions, also separated based on genders can be seen in Figure 4.

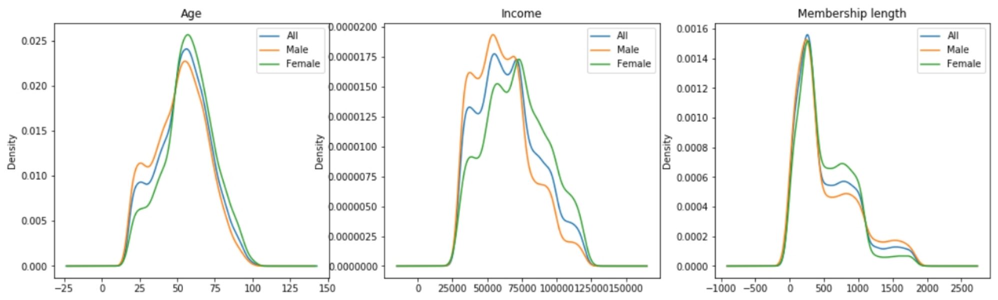


Figure 4: Distributions of the age, income and membership data in the dataset

From the above plots we derive the following conclusions:

- The age distribution is relative nice. It has got a small peak around 25 years, but we assume that it will not pose a serious problem during training.
- The income is not perfectly well distributed, it has got some local peaks at certain points. However it will possibly be sufficient for training. There are some differences between the male and female distributions. The male costumers tend to have lower income than the female costumers.
- The membership length on the other hand is quite badly distributed. It has a huge peak between 0 and 500. Other than that, there tend to be some intervals where the distribution is almost constant. This left skewed distribution will probably require some transformation before using it for training.

Based on these experiments we conclude that it will probably be sufficient to only standardize the age and income distributions and only the membership length data needs to be transformed to behave better during training.

Offer Data Exploration

For being able to explore the transcript data, first the field ”value” has to be extracted. 3 new columns, called ”offer_id”, ”amount” and ”reward” are created and are filled with the values from the ”value” column’s dictionaries.

As it will be seen later, we mostly create our features from statistical data about the offers the costumers receive. For this reason firs we observed some properties of the offers related data in the transcript dataset. We checked the number of received, viewed and completed offers from each type. (From now on we use the phrase ”offer” both for actual offers and for simply informational material.) Obviously an informational material can not be completed, however it can be viewed. The distributions can be seen in Figure 5. (The numbers 0-9 correspond to different offers. These were assigned to the offers based on their order in the portfolio dataset)

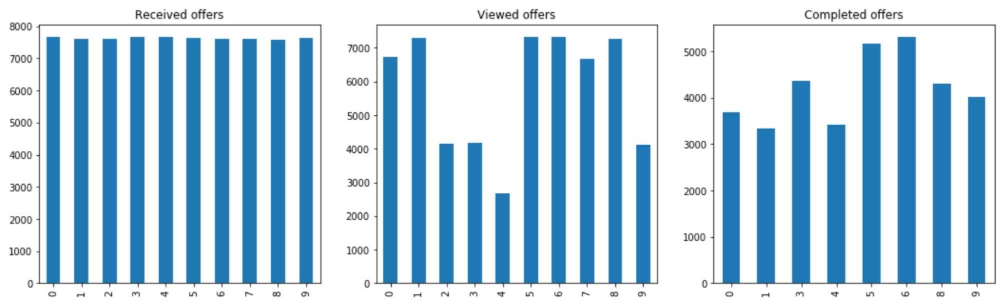


Figure 5: Number of offers received, viewed and completed

The following observations can be made based on these figures:

- The amount of sent out promotions is almost the same for every offer type.
- The amount of viewed promotions varies stronger and by far the least people viewed the promotion with the highest difficulty and longest duration.
- The completion is also relatively homogeneous. (The offer types 2 and 7 correspond to the informational type offers which can not be completed, hence these are missing from the table.)

After this we have examined how the distribution of the success of the offers look like. For this we have calculated for every person in the dataset, how many times he/she a) viewed, b) viewed and completed and c) not viewed but still completed an offer. The resultant numbers can be examined in Figure 6. (The x axis of the plots represent the amount of offers per person that fulfills the property indicated in the plot title and the y axis represents the number of people.)

These plots look quite like what we expected:

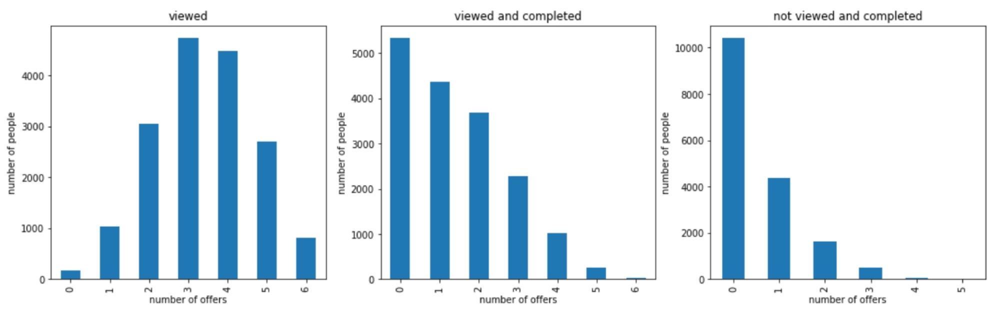


Figure 6: Sample from the transcript dataset

- The number of viewed offers is quite normally distributed. It is not very common that people never look at offers not that they view every single offer they receive.
- The number of people who view and then complete an offer several times decreases linearly with the number of occurrences.
- The number of people who view and then complete an offer several times decreases in an exponential-like fashion. It is very improbable, that someone would complete several offers accidentally without knowing about the offer.

Transaction data Exploration

A huge part of the transcript dataset are transactions. In the followings we derive some conclusions about these data.

During exploration of the transactions data we have quickly realized that there are some outliers which seriously distort the statistics about the data. The issue can easily be seen in Figure 7 where the distribution of the amount of a single transaction is shown:

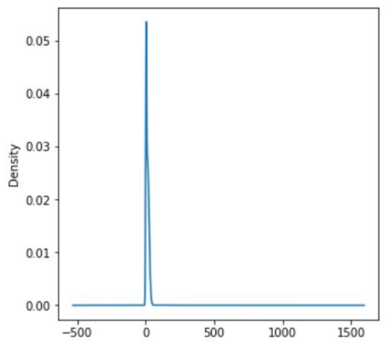


Figure 7: Effect of outlier transaction

There is an outlier transaction with an amount of 1062.28$ although the mean for the whole dataset is around 12.7$. Most of the data is in a normal range, however there are some unlikely high values in the dataset.

A very similar can be said about the cumulative amount of money a person sent during the 30 day period. The mean value for the cumulative spent amount is around 104$, however the maximum value is 1608$. Also there are a total number of 422 people who did not spend any money during this period. In Table 2.2 we show the number of people who spent more than a specified threshold.

	> 100	> 200	> 300	> 500	> 1000
number of people	6722	2349	738	272	48

Table 1: Number of people in the dataset who spent more than different thresholds during the 30 days.

We decided that some people should be excluded from the data for the following two reasons: Either they are not representative for the whole group, or they are not the target who the company should consider when they plan their advertisement campaign. By not being representative we mean, that a portion of the group spends much more money than the rest of the group. They are outliers in the dataset and would just reduce the performance of the model. The second group is the people who have not spent any money during the 30 day period. These people should not be considered as potential costumers and hence can be discarded from the dataset.

In Figure 8 the distribution of the cumulative amount spent by a person can be seen if a group of people whose purchases are over a specified threshold are discarded from the data.

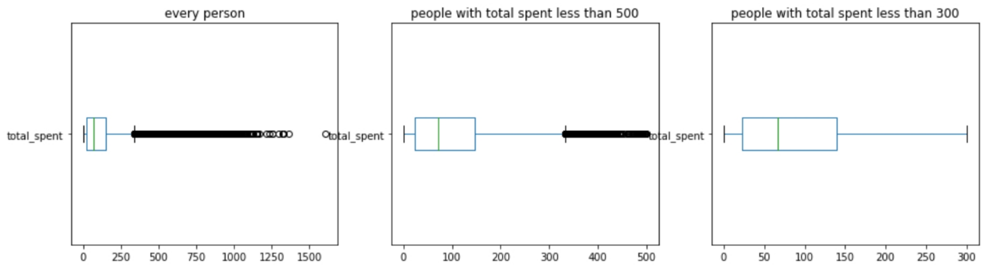


Figure 8: Distribution of cumulative money spent during the 30 day period

On the first two figures one can see the effect of outlier data in the dataset. Based on these experiments we have decided to discard the people (and all data related to them) from the dataset, who either have spent 0$ or more than 300$ during the 30 day period. This means that a total number of 1160 people will be dropped from the data. This is only 6.8% of the people and we assume that around the same amount of data will be lost from the transcript dataset.

2.3 Algorithms and Techniques

It is a crucial step to preprocess the data that later will be fed to the model during training. To be able to decide which preprocessing steps are required for which features data visualization is unavoidable. After we have obtained some characteristics of the data through visualization we concluded that the following algorithms are going to be use through preprocessing:

The age and income features are relatively nicely distributed and hence will only have to be standardized. We would like to carry out the standardization before filling up the missing values with the respective means, since otherwise these would distort the data strongly. We have chosen sklearn's StandarScaler algorithm since it is able not to consider "None" type values. The scaler standardizes the data using the following formula:

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the feature calculated for the whole dataset.

The membership length is a strongly left skewed distribution and hence has to be transformed. For this we have chosen to use sklearn's QuantileTransform algorithm. This method transforms the features to follow a uniform or a normal distribution using quantiles information. Therefore, for a given feature, this transformation tends to spread out the most frequent values.

Due to the reasons detailed in the previous subsections, the people and all data related to them, who tend to be outliers because of their spending are going to be removed from the dataset.

An important part of the solution that we have chosen is the generation of training data from the provided dataset. This algorithm is going to be detailed later in the Methodology section.

Chosen model

we are going to train a feed forward neural network to predict into which category an offer is going to fall after it is sent to a costumer. Feed forward neural networks are relatively easy to train and are universal approximators in the sense, that an infinitely complex neural network can approximate any given nonlinear function. This does not necessarily mean that neural networks are the best choice for every task. However they are quite usual choices for data where one suspects very tricky underlying distribution as it is in this case.

The neural network is going to be implemented using the PyTorch framework.

2.4 Banchmark Model

We have chosen an easily implementable and easy to evaluate model as benchmark model. A kNN (k Nearest Neighbor) model is going to be used. As we suspect that similar people react similarly to offers a kNN classifier will possibly perform quite well for the task and hence will be a good benchmark model to compare the performance of the neural network to.

3 Methodology

In the previous section some issues were identified with the dataset. The *Algorithms and Techniques* subsection details the algorithms that were used to remedy these issues. In this section we show the results that were acquired during data preprocessing.

The other important contribution of this section is the feature engineering part. This provides details about how the features from the dataset were extracted.

3.1 Preprocessing the Portfolio Dataset

We do not use much data about the portfolio dataset. We do realize that some interesting features could have also been extracted from this data, however we decided to focus on other features of the data. In the full created feature vectors the offer that was received is one-hot-encoded to be able to feed it to the neural network. Also the duration of the offers is converted to hours to be able to use it with the transcript dataset where the timestamp is provided in hours instead of days.

3.2 Preprocessing the Profile Dataset

Due to the reasons detailed in the previous section we removed the people who have spent more than a total of 300$ or did not spend anything. With this we have removed 6.8% of the profile and and 7.3% of the transcript data. These are not very large percentages, so there are plenty of data remaining after this selection.

After this, we have one-hot-encoded the gender into the following categories: "F", "M", "O" and "U".

Before filling up the "None" type values in the dataset, the age and income fields are standardized using sklearn's StandardScaler algorithm. The features before and after standardization can be seen on Figure 9

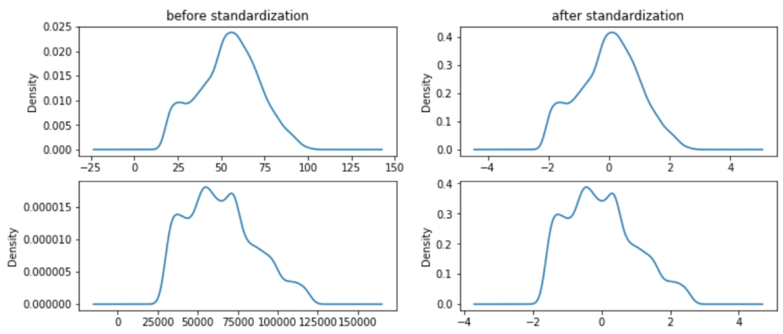


Figure 9: Age and income before and after standardization

After standardizing the age and income data, the missing values can be filled with the respective means.

Next the membership length is calculated and since it was shown that it has a quite distorted distribution, it will be transformed using sklenarn's Quintile-Transform algorithm. After that the resultant data is also standardized. The data before and after the transformation can be examined in Figure 10.

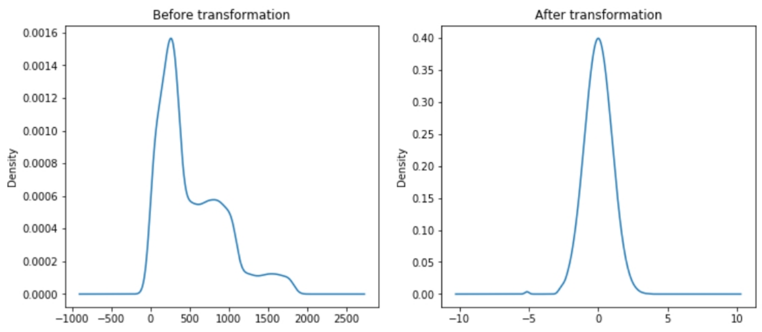


Figure 10: Membership length before and after the transformation and standardization.

After the data was preprocessed and cleaned it is ready for handcrafted feature creation.

3.3 Feature Engineering

A crutial part of out solution is the way the features and the corresponding labels are handcrafted from the raw data. In this subsection we define how the feature vector is composed and briefly introduce the algorithm that manufactures it from the raw dataset.

Our main goal is to be able to predict that if an offer would be sent to a specified costumer, what the costumers most probable reaction for receiving the offer would be. We are trying to classify each received offer into the 4 groups that were introduced int section 1.2. For this in a real life scenario we can not use data from the "future", but one has to make its prediction based on past experiences with the costumer. It is also possible that the costumer is fresh, that is the company does not yet have a record about this person yet, however we would also like to make predictions about that person.

To achieve this, features are required which we assume that represent the costumers in detail and also are usable for costumers with different recorded history. To achieve this goal, we have chosen to create the features with the following procedure:

First we pruned the datapoints from the transcript data where the offer was received so late, that its expiration date is out of the scope of the 30 day period. These offers are not representative in our point of view. Then we create a training point for every received offer, since every single received offer can be examined, how the costumer reacted to it. Hence, for every received offer we check, into which category it falls in the categories of section 1.2. After that we take the history (if there is any) of the person who received the offer only until that point when the offer was received. Than we create some representative features based on this history segment. The features that we create are as follows:

- Personal data: Age, income, gender, membership length. These are the data that we have about every costumer irregardless of their recorded history length. (For new costumers these are the only relevant features that we have.)
- General costumer history: Some statistical data about the costumer's activity until the point the offer was received. It includes the following features:
  average amount spent
  number of offers received
  fraction of viewed and received offers
  fraction of completed and viewed offers
  fraction of completed but not viewed offers
- Costumer history in connection with this product. Some measuring numbers about the costumer's history with this received product. It includes the following features:
  number of offers received from this type
  fraction of viewed and received offers of this type
  fraction of completed and viewed offers of this type
  fraction of completed but not viewed offers of this type
- type of the offer in one-hot-encoding

.

These features allow to represent a costumer quite in detail. With this method we have created 61324 training points for the model. The number of training points for the different classes can be seen in Figure 11

The differences in the number of labels is going to be compensated during training.

We have also checked the correlation matrix of the generated features. We have found that the features are not much correlated, they are going to be sufficiently independent for training.

3.4 Implementation

In this section we are going to provide some details how te project was implemented. Most of the implementation details are spread in this document, because we have found that the report is better readable in this way. Here however we provide some details on the technical workflow of implementation.

During implementation we have been working along the following workflow:

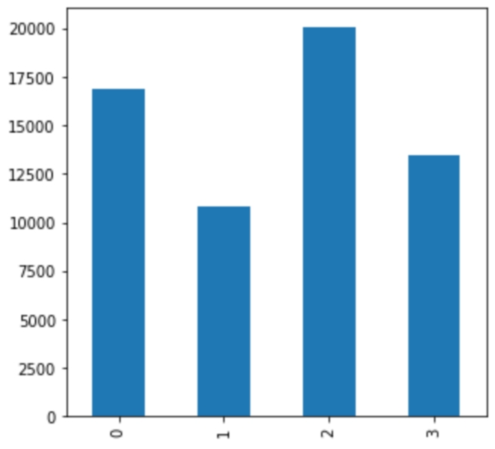


Figure 11: Number of labels in the 4 classes.

- Data Exploration
- Feature Engineering
- Preparing the data for training
- splitting the data into training (80%), validation (10%) and test sets (10%)
- Neural network implementation
- Iterating between hyperparameter tuning, relevant feature selection and training until we have found the best setup based on validation accuracy
- Testing the accuracy on the test set
- comparing the performance if the trained and the benchmark models.

We have mostly used python notebooks during implementation. The Data Exploration and the Feature Engineering parts can respectively be found in their notebook. The rest of the steps, which are all related to training, are implemented in a third training notebook. For better readability more complicated functions are imported from .py scripts which can be found in the *source* directory. Some implemented functions use cache data if it is available, which are stored in the *cache* folder. The trained model's state dictionary are stored in the *models* directory.

The neural network is implemented in an own script file. The network was implemented using the PyTorch framework. During implementation care was taken to be able to define architectures with different depths and hidden layer sizes. This allows to discover the best network architecture for the task. To achieve this, we have used PyTorch's ModuleList, which is a custom list for storing layers and also allow to integrate these layers into the module. (The layer parameters appear in the module for optimization.)

We have also implemented a "NN_Solver" class, for being able to train the neural network. This class provides various methods, including a "train" method, which trains the network with the given parameters. It also provides validation loss and accuracy information after each epoch. The model with the best validation accuracy is saved after training into the *model* directory. It is also returned for further use (testing). For training we have used the following hyperparameters and techniques:

- Architecture: Feed forward neural network.
  input dimension: 25
  hidden dimensions: [256, 512, 512, 128]
  output dimension: 4
  activation: Leaky ReLU
  output activation: Softmax
- Oprimization algorithm: ADAM solver
  $\beta_1 = 0.9$, $\beta_2 = 0.99$
  ε =1e-8
  weight decay = 0
- Loss function: Cross entropy loss with weights for compensating the different number of occurrences of the labels.
- Regularization: dropout with probability of 0.2
- Learning rate: 1e-4

The implementation was checked by the usual method: We have tried to overfit a single data instance in the dataset. This verifies that there are probably no implementation errors in the neural network definition and that the optimization works. We have managed to overfit the single data instance even with a very small architecture, so from this we deduce that the implementation is correct.

For the benchmark model we have used sklearn's KNeighborsClassifier. We have fed it the same training data and checked its accuracy with several neighbor numbers.

As it was stated previously, we used *accuracy* as the only metric for evaluating the performance of the multiclass classifier. After finding the best hyperparameter setting for training using the validation set, we have trained the model and tested it on the test set. The performance was then compared with the kNN's performance. The results are going to be shown in section 4

3.5 Refinement

During code development there are always issues which occur only during solving the problem. These issues require some refinement during development. In this subsection we summarize the main refinement and development related steps that were taken.

Feature Engineering

The feature engineering algorithm is probably the most complex algorithm that was developed by us during this project. It needed to be tested, whether it works correctly or not. This testing was impossible on the original dataset due to its size and complexity. Instead, we created a toy transcript dataset, which only includes around 15 datapoints. During developing this toy dataset, care has been taken that every possible situation in the original dataset is modeled. After running the feature extractor algorithm on this toy transcript data, we could identify bugs in the code. After some iterations we then were able to verify that the algorithm is implemented correctly.

We have also experimented extracting some further features from the data, and we have trained the models on this data as well. We have found out however, that the most relevant features for training are the ones introduced in section 3.3.

Hyperparameter Tuning

A crucial requirement to successfully train a neural network is to correctly chose hyperparameters. This usually includes some systematic search in the hyperparameter space together with some intuitions. During our search we have experimented with different network architectures (depth, hidden layer size), batch size parameters, regularization techniques, regularization strengths, activation functions and learning rate. During these experiments we have found that the best parameter setting for our purpose is the one stated in section 3.4

4 Results

After carrying out carefully the data preprocessing, feature engineering and training steps, we have obtained the trained classifier. In this section we are going to present the results of the training.

4.1 Model Evaluation and Validation

After validating by overfitting to a single training example that the model is implemented correctly and is able to learn we carried out a hyperparameter search in the parameter space. The hyperparameters were chosen with which the highest validation accuracy was acquired. These parameters are provided in section 3.4. The model was then fully trained with these parameters. The training and validation loss during training can be examined in Figure 12

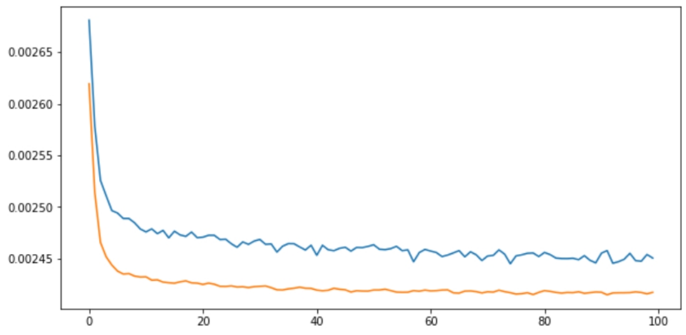


Figure 12: Training (blue) and validation (orange) loss during training.

From this figure one can conclude that the training has converged to a local optima. The regularization is also sufficient as the model does not overfit to the data. (Does not memorizes the labels of the training set.) The phenomena that the validation loss is smaller than the training loss is thanks to the regularization technique. Since dropout is only used during training and is switched off during validation, the validation loss becomes lower during training.

After training, the model was evaluated on the test set which was never shown to the model before. On the test set the model was able to reach an accuracy of 52%.

The benchmark model was also evaluated with different number of neighbors. It was able to produce an accuracy of 46% with k=10 neighbors.

Hence we conclude that the trained neural network was able to outperform the kNN classifier slightly. Further analysis and justification of the results is provided in the next subsection.

4.2 Justification

The successfully trained model was tested on the test set which was never seen before. It acquired 52% accuracy on this set. This is a slight improvement compared to the accuracy of the benchmark kNN classifier (46%).

Although this accuracy does not seem to be very high, let us place it into context. We have also expected a somewhat higher accuracy than this number. 52% accuracy means that only in slightly more than half of the cases the future of a received offer were predicted successfully. In other words, the classifier is only able to predict the reaction of every second costumer.

However one have to consider the following properties of the situation to be able to evaluate the performance in its context. First of all, the received offers are classified into 4 classes. This means that a random classifier would achieve an accuracy around 25%. Our classifier has reached an accuracy more than twice as high. Also a very important aspect that one has to keep in mind is how dynamically change the behavior of the people. It really is a stochastic environment. It is totally possible, that someone have completed a certain offer 5 out of 5 times before and still does not completes it the 6^{th} time. People behave so stochastically that it is impossible to tell their reaction to certain offers with a very high accuracy.

In this respect we conclude that although this accuracy is not very high, it is still a surprisingly good result considering the complexity and randomness of the task it solves. Being able to predict the exact outcome of an advertising campaign for every second person could be a very powerful tool for companies such as Starbucks.

4.3 Summary

In this report we have detailed our solution for Starbuck's Capstone Challenge. Our chosen aim was to try to predict, into which of the following four classes a received offer is going to fall given some basic data about the person who receives the offer and its costumer history until receiving the offer.:

- Will not be viewed nor completed accidentally
- Will not be viewed but will be completed
- Will be viewed but will not be completed
- Will be completed and viewed

We have explored the provided data thoroughly, identified the issues with it and provided solutions to remedy them. We have preprocessed the dataset to make it sufficiently well behaving for training and developed a feature representation which from our point of view describes the costumers and their costumer history until receiving the offer accurately. We have extracted the features from the raw dataset and created the corresponding labels.

We have implemented a feed forward neural network for the above classification purpose. After searching the sufficient hyperparameters, we have trained the network with the training data. We then compared its results with our benchmark model, which is a kNN classifier with 10 neighbors.

We have reached an accuracy of 52% on the testing set, which is a slight improvement on the kNN classifier which have reached 46%. Although these accuracy numbers do not seem to be very high, considering the randomness on how people act in the real word, these numbers are quite impressive in our point of view. For every second person one is able to predict exactly what the future of an offer is going to be.

This can be a powerful tool for companies. Considering the accuracy of the model, a useful application of such a model would be to check the success of a campaign for a large number of costumers and obtain whether for most of the people it would be successful or not.

4.4 Outlook

After training the model, we have also trained an other network for a slightly different purpose. We have re-classified the received offers only into successful and not successful categories. We considered the offers to be successful if they were viewed and completed or if they were plain informational offers and they were viewed. We have trained a neural network with the same features for this binary classification task.

We were able to obtain the following results with it:

- accuracy: 65%
- precision: 49%
- recall: 75%
- false negative rate: 25%

From these measures the high recall and low false negative rate are especially appealing. This means that the model mostly classifies successful outcomes correctly and most of the failures are made by classifying some offers to be successful although they will not be successful in fact. This is the smaller problem. A bigger problem would be if customers who would otherwise not receive the offer do not receive the offer at all.

In this respect a further improvement in this direction could be an interesting field of further research.