Machine Learning Engineer Nanodegree

Capstone Project – Starbucks Offers

David Morcuende Cantador

January 2nd, 2020

I. Definition

Project Overview

This data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an **advertisement** for a drink or an actual offer such as a **discount** or **BOGO** (buy one get one free). Some users might not receive any offer during certain weeks.

Not all users receive the same offer, and that is the challenge to solve with this data set.

Your task is to combine **transaction, demographic and offer data** to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

Every offer has a **validity period** before the offer expires. As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, you can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

You'll be given transactional data showing user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase. This transactional data also has a record for each offer that a user receives as well as a record for when a user actually views the offer. There are also records for when a user completes an offer.

Keep in mind as well that someone using the app might make a purchase through the app without having received an offer or seen an offer.

Example

To give an example, a user could receive a discount offer buy 10 dollars get 2 off on Monday. The offer is valid for 10 days from receipt. If the customer accumulates at least 10 dollars in purchases during the validity period, the customer completes the offer.

However, there are a few things to watch out for in this data set. Customers do not opt into the offers that they receive; in other words, a user can receive an offer, never actually view the offer, and still complete the offer. For example, a user might receive the "buy 10 dollars get 2 dollars off offer", but the user never opens the offer during the 10-day validity period. The customer spends 15 dollars during those ten days. There will be an offer completion record in the data set; however, the customer was not influenced by the offer because the customer never viewed the offer.

Cleaning

This makes data cleaning especially important and tricky.

You'll also want to consider that **some demographic groups will make purchases even if they don't receive an offer**. From a business perspective, if a customer is going to make a 10-dollar purchase without an offer anyway, you wouldn't want to send a buy 10 dollars get 2 dollars off offer. You'll want to try to assess what a certain demographic group will buy when not receiving any offers.

Problem Statement

You can solve the problem in three different ways depends on what are you looking for: **multi-label, multiclass** or as a **binary classifier**.

1. multiclass: in multi-class problems the classes are mutually exclusive
2. multi-label: in multi-label problems each label represents a different classification task, but the tasks are somehow related (so there is a benefit in tackling them together rather than separately)
3. binary classifier: is the task of classifying the elements of a given set into two groups (predicting which group each one belongs to).

1)

For the multiclass problem the goal is to get the offer that will give the maximum benefit to the company for a user

2)
For the multi-label problem, the goal is to get the offers that a certain user will complete

3)
For the binary classifier problem, the goal is to answer if a certain user will complete an offer or not.

Classifier	Problems
Custom ANN	1,2
XGboost	3
Naive Bayes, BinaryRelevance	1
Naive Bayes, LabelPowerset	1
Naive Bayes, MLkNN	1
Naive Bayes, ClassifierChain	1

Metrics

The metrics used to measure the performance of the models are, **Binary Cross Entropy** and **Accuracy classification** (Jaccard similarity)

For the custom ANN the loss criterion is **Binary Cross Entropy** (is defined on probability estimates). It is commonly used in (multinomial) logistic regression and neural networks, as well as in some variants of expectation-maximization, and can be used to evaluate the probability outputs of a classifier instead of its discrete predictions.

For binary classification with a true label y {0,1} and a probability estimate p = Pr(y=1), the log loss per sample is the negative log-likelihood of the classifier given the true label:

$$L_{\log}(y,p) = -\log \Pr(y|p) = -(y\log(p) + (1-y)\log(1-p))$$

For the other classifiers the metric used is **accuracy score** that computes the accuracy, either the fraction or the count of correct predictions.

In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

If y^i is the predicted value of the i-th sample and yi is the corresponding true value, then the fraction of correct predictions over n samples is defined as

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

II. Analysis

Data Exploration

The data is contained in three files:

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

Here is the schema and explanation of each variable in the files:

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

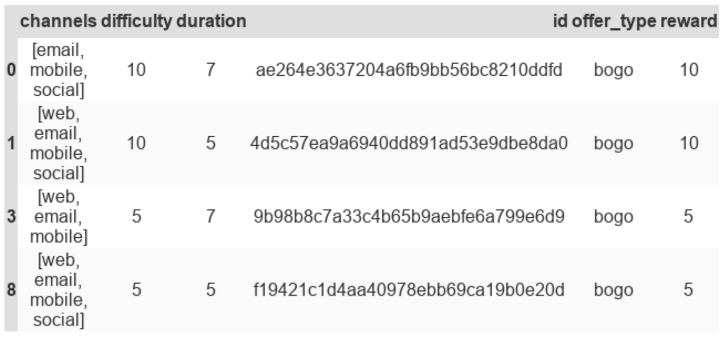
transcript.json

- 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

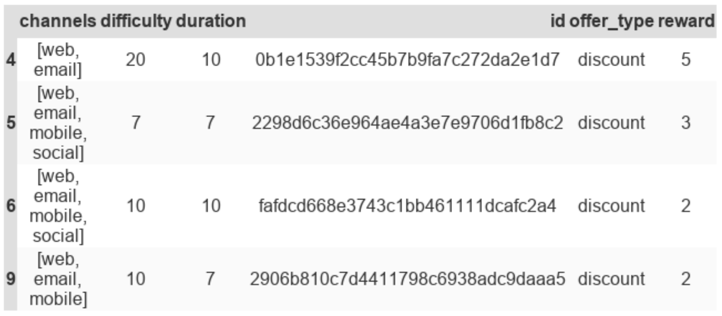
Exploratory Visualization

Offers

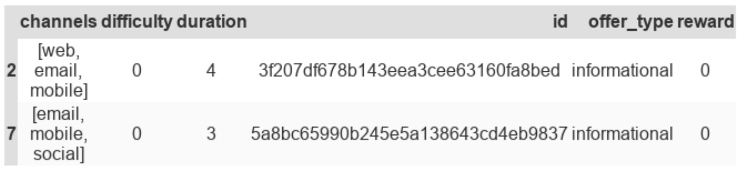
There is 3 kind of offers with 4 different channels and each offer has a difficulty and a reward associated.



	channels	difficulty	duration	id	offer_type	reward
0	[email, mobile, social]	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10
1	[web, email, mobile, social]	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10
3	[web, email, mobile]	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5
8	[web, email, mobile, social]	5	5	f19421c1d4aa40978ebb69ca19b0e20d	bogo	5



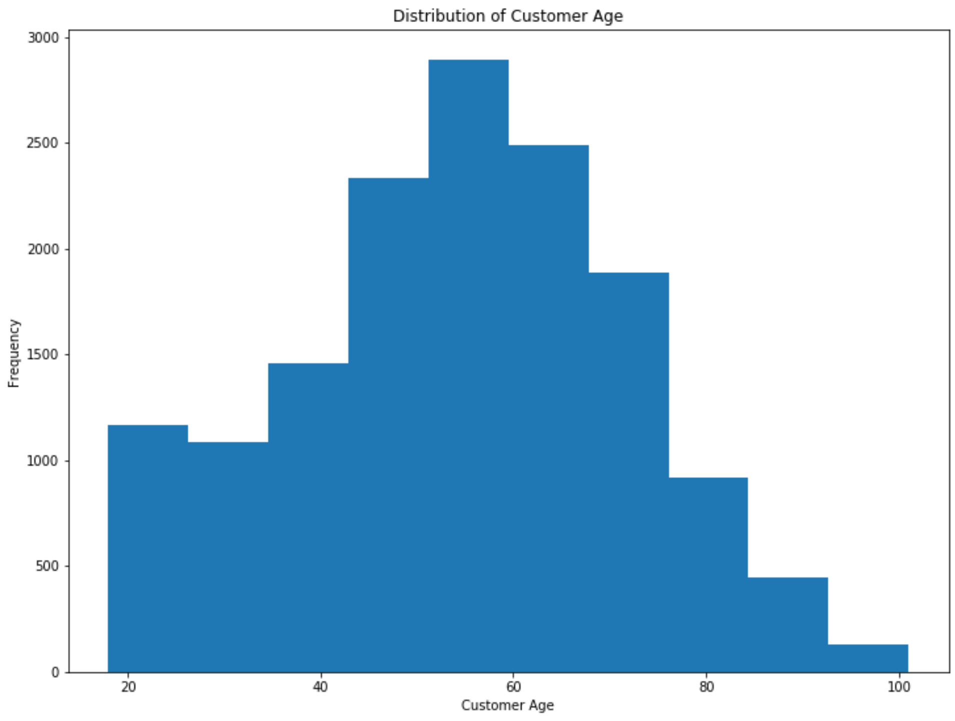
	channels	difficulty	duration	id	offer_type	reward
4	[web, email]	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5
5	[web, email, mobile, social]	7	7	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	3
6	[web, email, mobile, social]	10	10	fafdcd668e3743c1bb461111dcafc2a4	discount	2
9	[web, email, mobile]	10	7	2906b810c7d4411798c6938adc9daaa5	discount	2



	channels	difficulty	duration	id	offer_type	reward
2	[web, email, mobile]	0	4	3f207df678b143eea3cee63160fa8bed	informational	0
7	[email, mobile, social]	0	3	5a8bc65990b245e5a138643cd4eb9837	informational	0

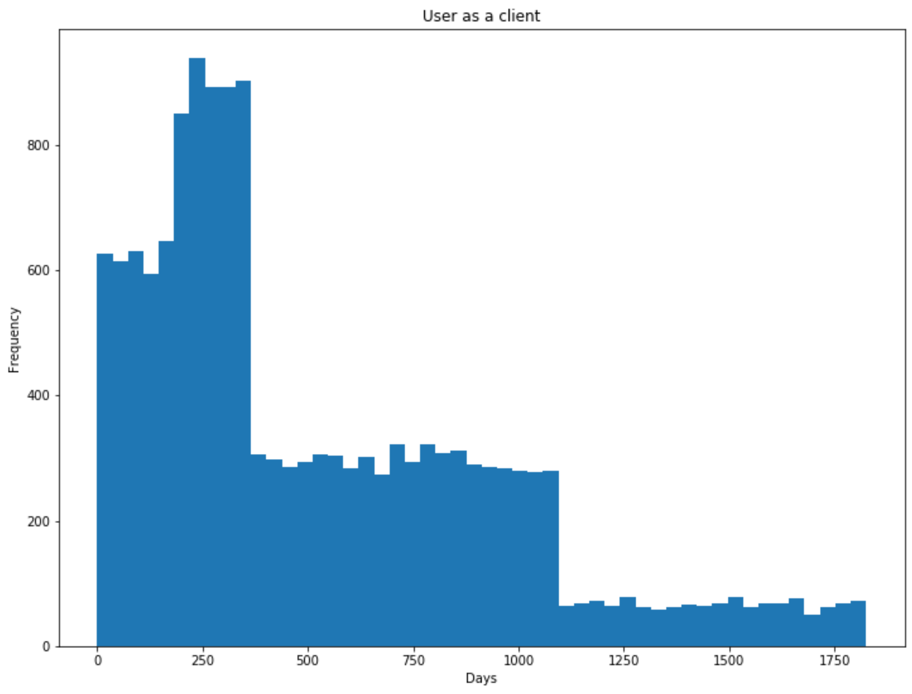
Distribution of customer age

Most of the users is between **40-65** followed by the range 20-40.



User as a client

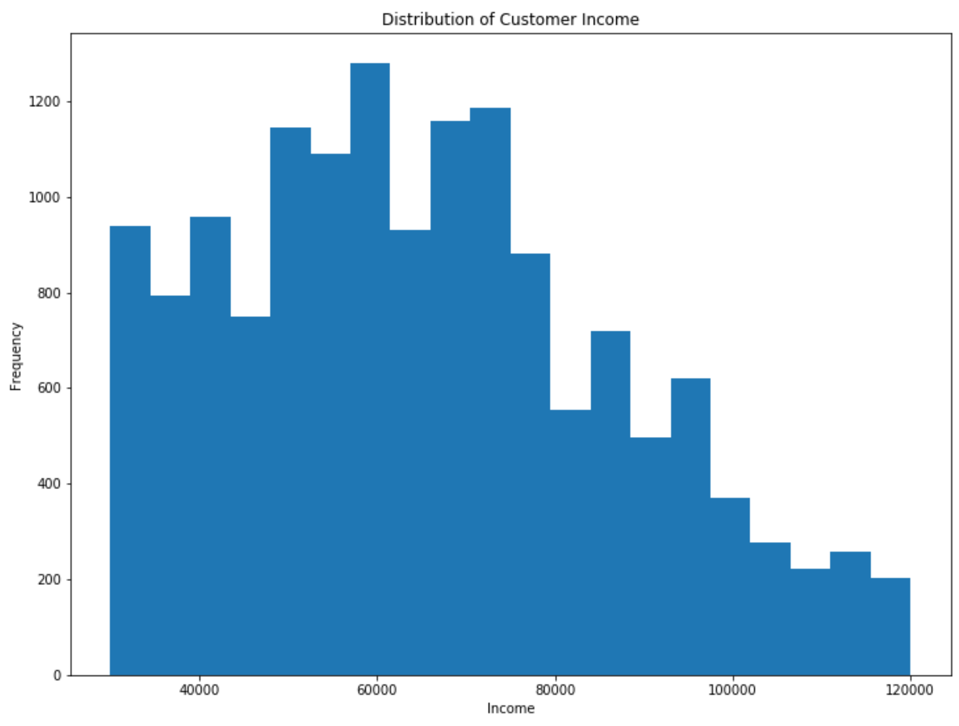
Below is the distribution of users in days since they sign-up (it is assumed to do the calculus that the last day is the last day of the dataset).



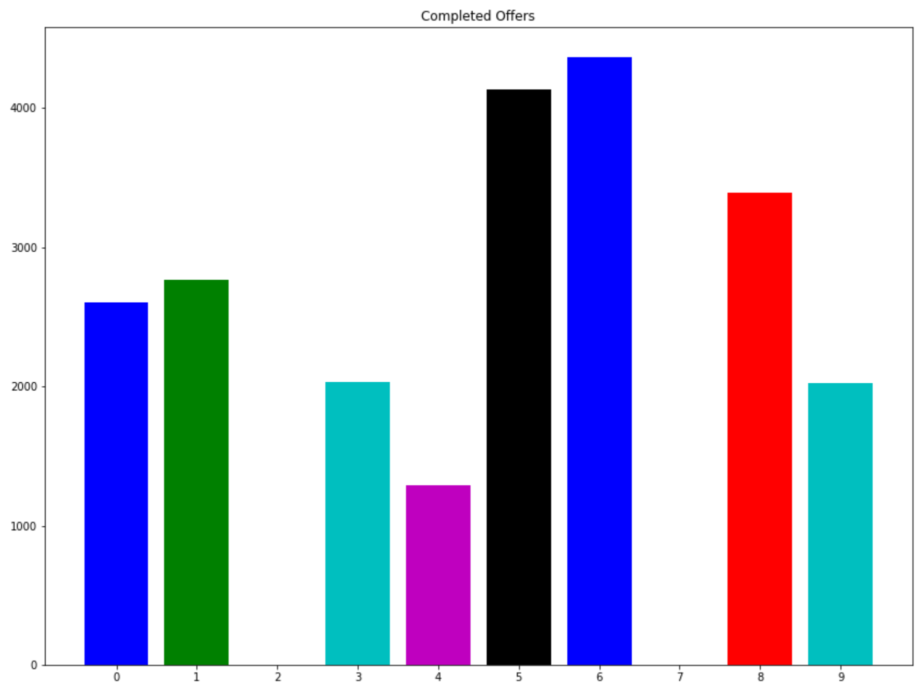
As we can see most of the users are the ones who have ben user for **250-400** days,

<u>The distribution of user's income</u>

The usual user income is from **30k to 65k.**



Completed Offers

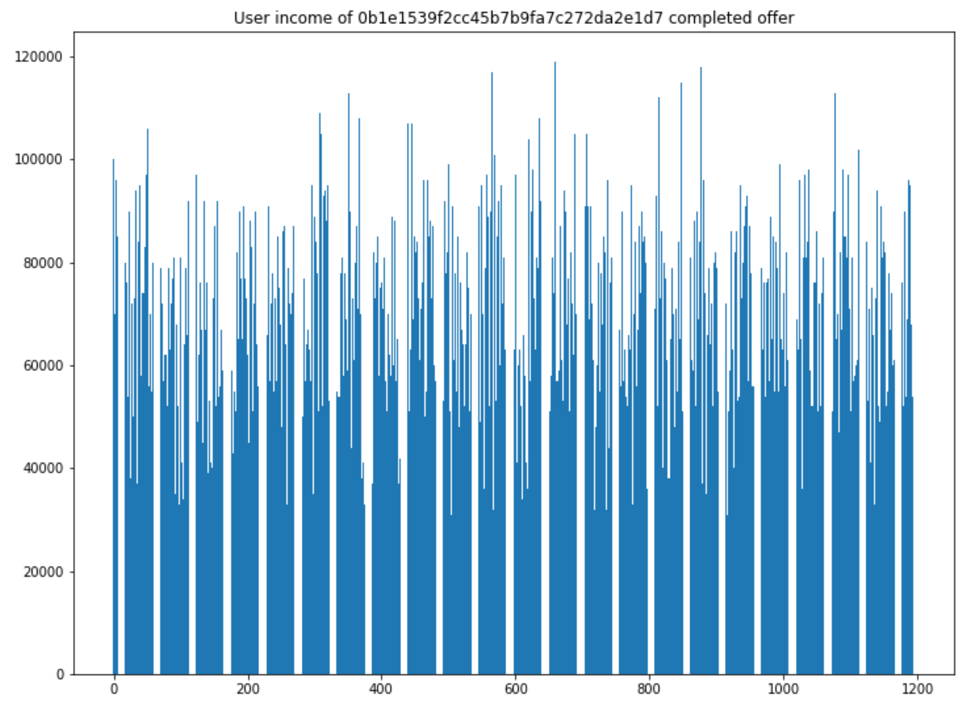


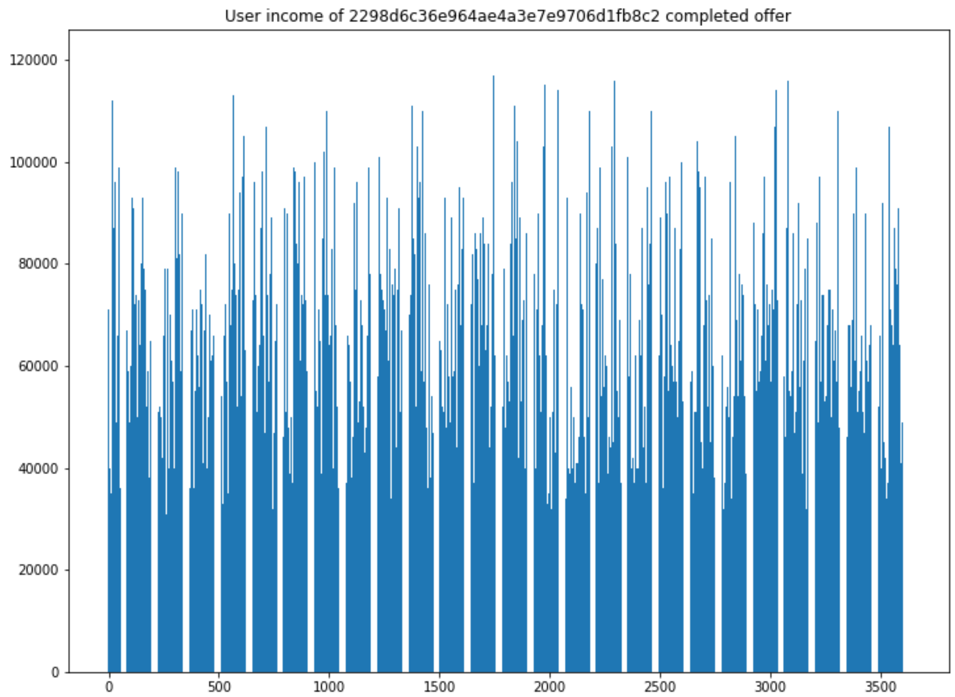
As you can see the more completed offers are:

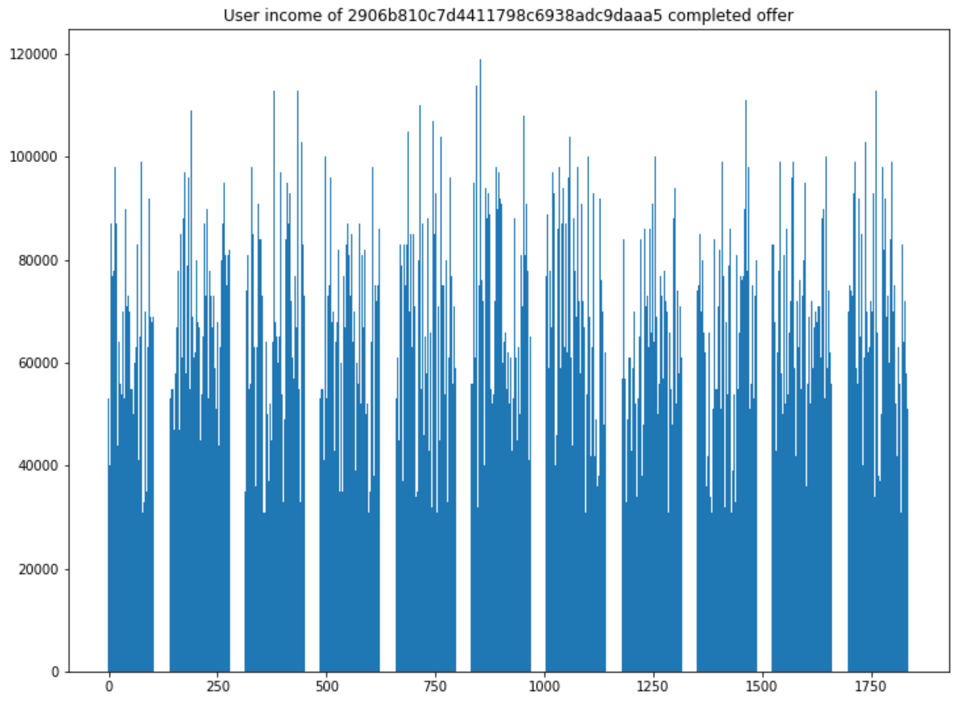
- Discount with rewad 3
- Discount with reward 2

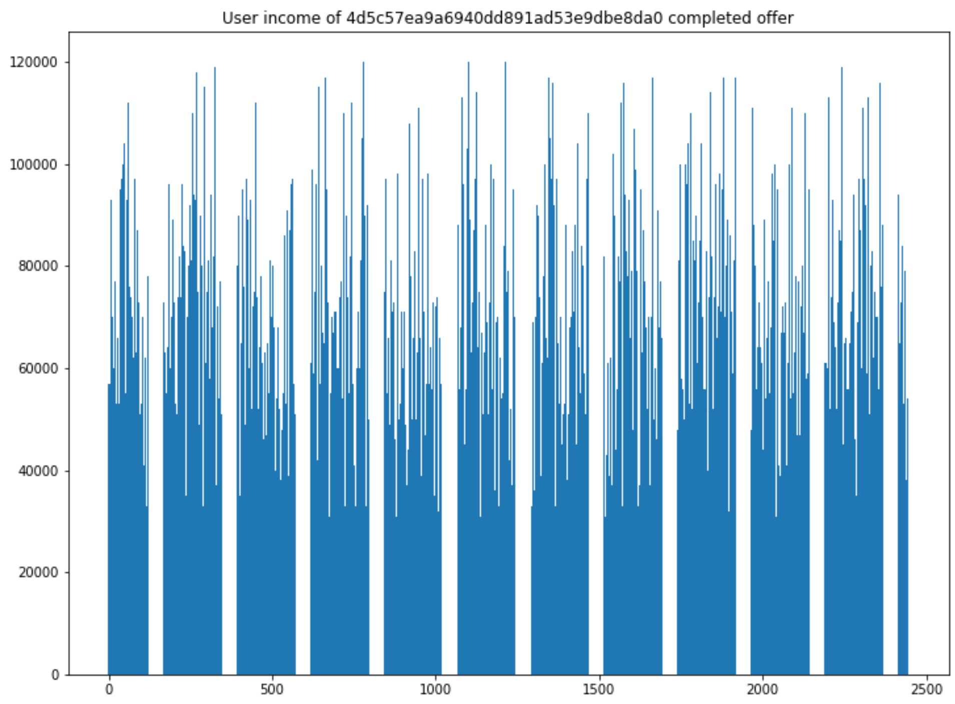
Both are on web, email, mobile and social channels (All the available channels).
The bogo type is the second top, with the half completed offers more or less.

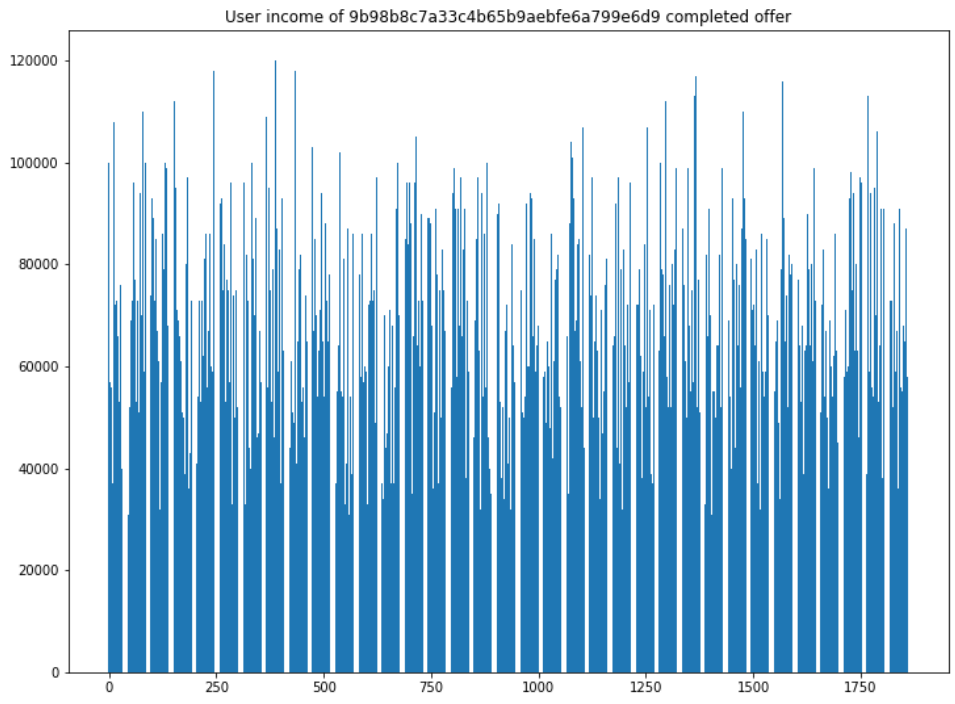
<u>Income of completed offers</u>

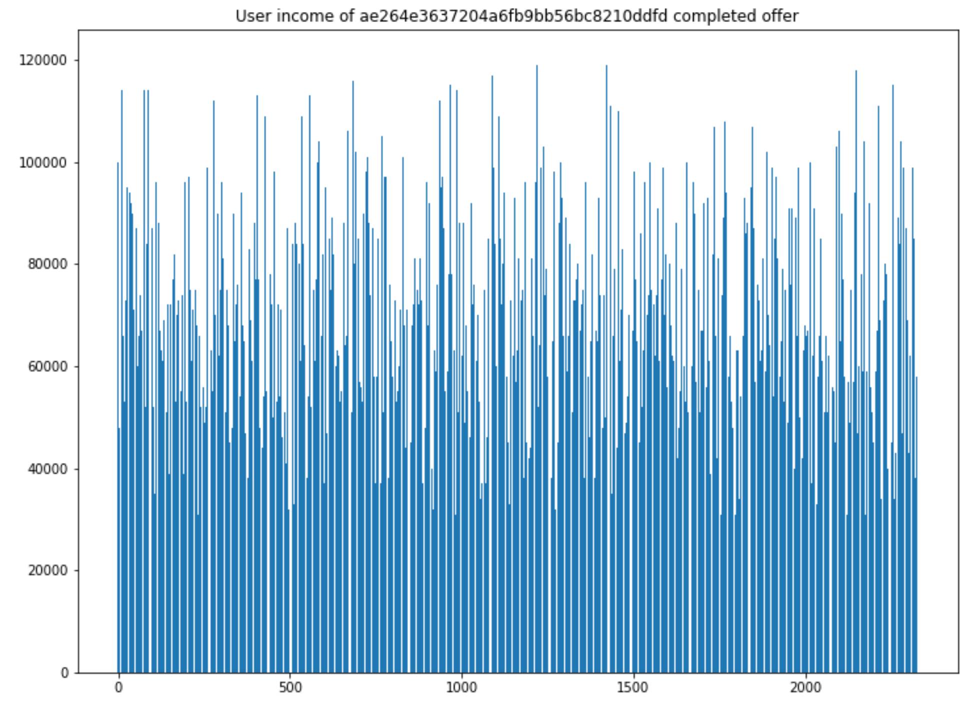


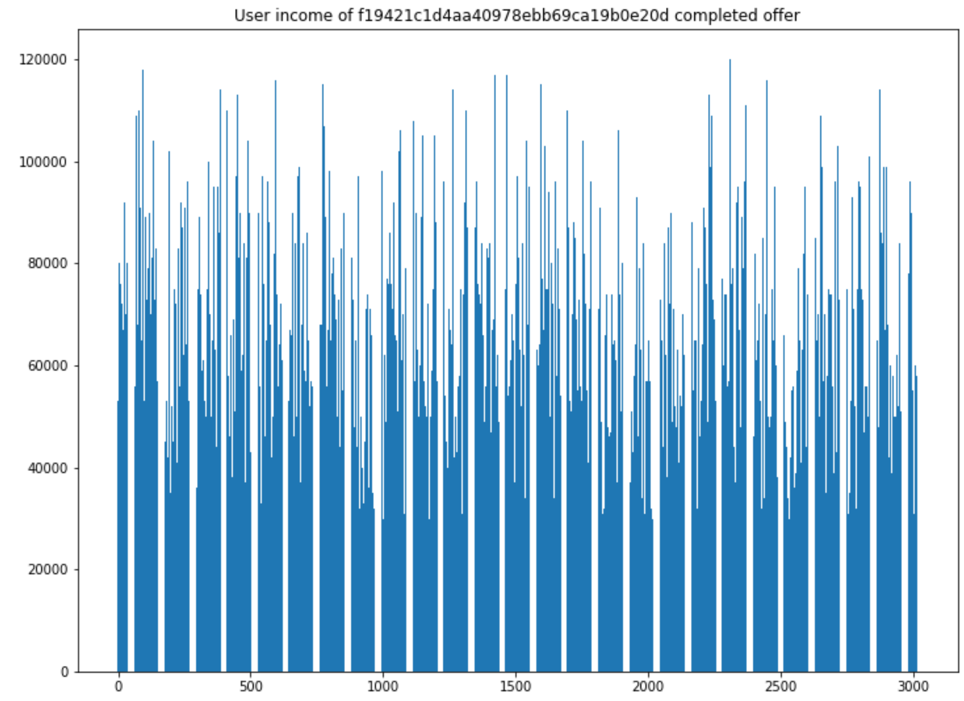

User income of 2298d6c36e964ae4a3e7e9706d1fb8c2 completed offer
120000
100000
80000
60000
40000
20000
0
0
500
1000
1500
2000
2500
3000
3500

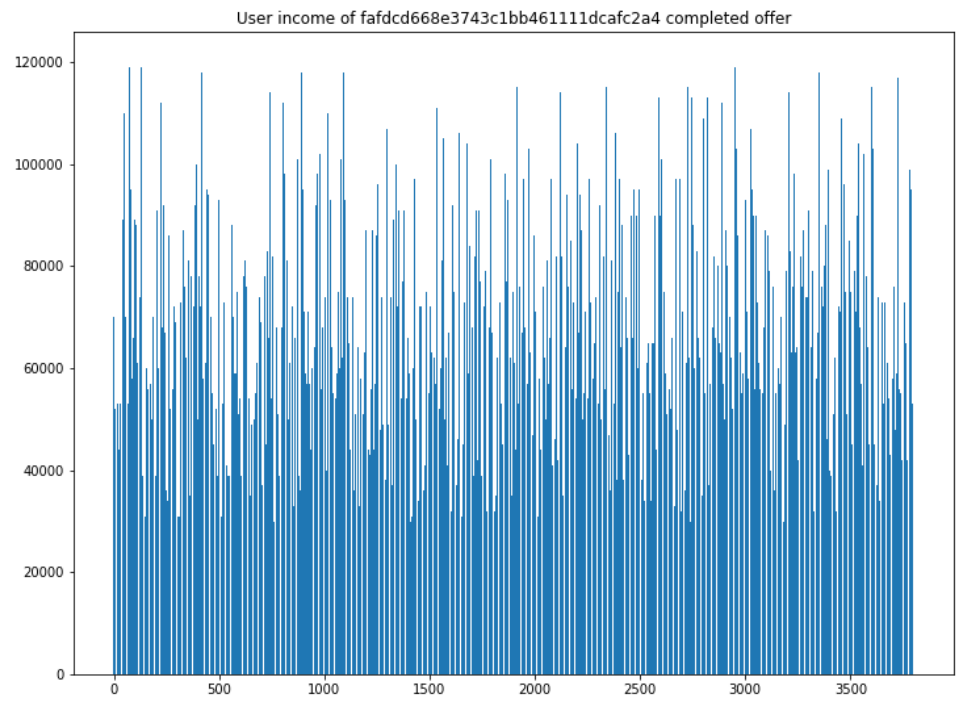

User income of 2906b810c7d4411798c6938adc9daaa5 completed offer
120000
100000
80000
60000
40000
20000
0
0
250
500
750
1000
1250
1500
1750


User income of 4d5c57ea9a6940dd891ad53e9dbe8da0 completed offer
0
20000
40000
60000
80000
100000
120000
0
500
1000
1500
2000
2500


User income of 9b98b8c7a33c4b65b9aebfe6a799e6d9 completed offer
0
20000
40000
60000
80000
100000
120000
0
250
500
750
1000
1250
1500
1750

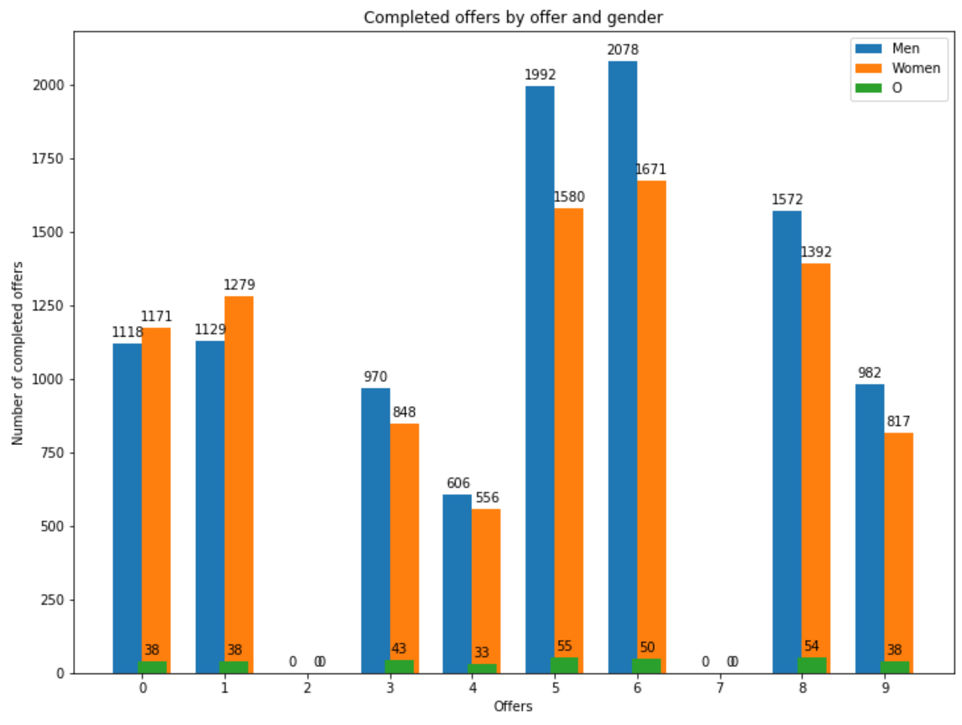

User income of ae264e3637204a6fb9bb56bc8210ddfd completed offer
120000
100000
80000
60000
40000
20000
0
0
500
1000
1500
2000


User income of f19421c1d4aa40978ebb69ca19b0e20d completed offer
120000
100000
80000
60000
40000
20000
0
0
500
1000
1500
2000
2500
3000



There isn’t any remarkable value about the income of the completed offers.

Completed offers by gender



On the 3-top completed offers there are more men than women like almost all the offers except on the BOGO ones who there are slightly more women than men.

The *Others* genre are underrepresented.

Algorithms and Techniques

Custom ANN

Artificial neural networks are built of simple elements called neurons, which take in a real value, multiply it by a weight, and run it through a non-linear activation function. By constructing multiple layers of neurons, each of which receives part of the input variables, and then passes on its results to the next layers, the network can learn very complex functions.

Its strength is its ability to dynamically create complex prediction functions and **emulate human thinking** (as this problem is about how humans complete offers), in a way that no other algorithm can. There are many classification problems for which neural networks have yielded the best results.

Naive Bayes Classifier

It calculates the probability that each of the features of a data point (the input variables) exists in each of the target classes. It then selects the category for which the probabilities are maximal. The model is based on an assumption (which is often not true) that the features are conditionally independent.

Naive Bayes is **surprisingly accurate for a large set of problem**s, scalable to very large data sets. But it **has problems where categories may be overlapping** or there are unknown categories so the set of categories selected must be exhaustive.

The naïve Bayes classifier has been implemented with:

- BinaryRelevance: Performs classification per label.
- ClassifierChain: A multi-label model that arranges binary classifiers into a chain.
- LabelPowerset: Transform multi-label problem to a multi-class problem.
- MLkNN: kNN classification method adapted for multi-label classification.

Benchmark

BCELoss (For the ANN)

Creates a criterion that measures the Binary Cross Entropy between the target and the output:

The unreduced loss can be described as:

$$\ell(x,y) = L = l1, \ldots, lN\top, ln = -wn[yn \cdot logxn + (1 - yn) \cdot log(1 - xn)],$$

where N is the batch size. If reduction is not none, then this is used for measuring the error of a reconstruction in for example an auto-encoder. Note that the targets yy should be numbers between 0 and 1.

$$\ell(x,y) = \begin{cases} \operatorname{mean}(L), & \text{if reduction} = \text{'mean';} \\ \operatorname{sum}(L), & \text{if reduction} = \text{'sum'.} \end{cases}$$

Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Recall

Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

F1-score

F1 Score is the weighted average of Precision and Recall.

III. Methodology

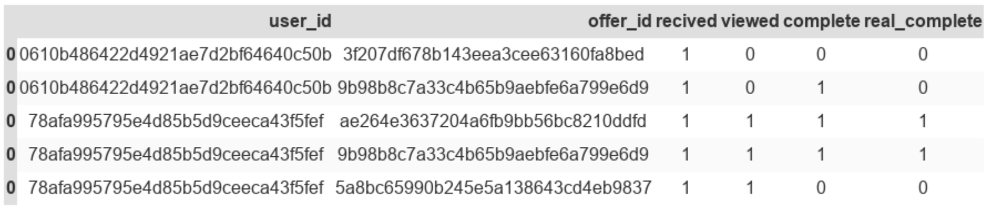
Data Preprocessing

To process all the information of the transactions a user-offer matrix has been made.

This matrix gets the information of the transcript and encode how much times a user has received, view or completed an offer

A real_complete column is added because a user can complete an offer without seeing it, so in order to get a value in the real_complete column the user had to had received, viewed and complete an offer in that order.

It looks like this:



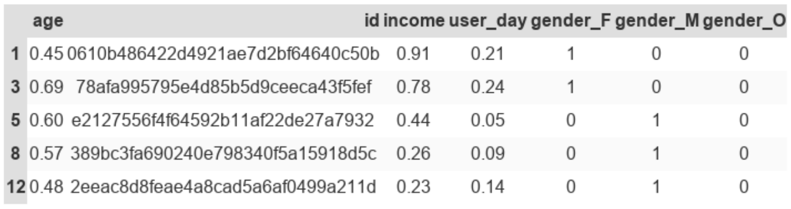
	user_id	offer_id	recived	viewed	complete	real_complete
0	0610b486422d4921ae7d2bf64640c50b	3f207df678b143eea3cee63160fa8bed	1	0	0	0
0	0610b486422d4921ae7d2bf64640c50b	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	0	1	0
0	78afa995795e4d85b5d9ceeca43f5fef	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1	1
0	78afa995795e4d85b5d9ceeca43f5fef	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	1	1
0	78afa995795e4d85b5d9ceeca43f5fef	5a8bc65990b245e5a138643cd4eb9837	1	1	0	0

Once there is a matrix correlating the users and the offers, we can model the data to the training

The data that is used for training will be:

- age
- gender
- income
- days a user has been a user

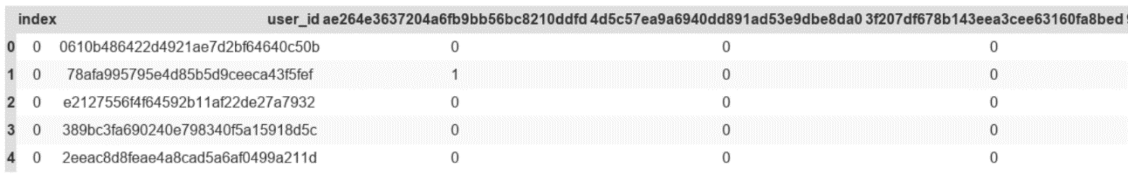
The age, income and user_day will be normalized, and the gender will be one-hot-encoded



	age	id	income	user_day	gender_F	gender_M	gender_O
1	0.45	0610b486422d4921ae7d2bf64640c50b	0.91	0.21	1	0	0
3	0.69	78afa995795e4d85b5d9ceeca43f5fef	0.78	0.24	1	0	0
5	0.60	e2127556f4f64592b11af22de27a7932	0.44	0.05	0	1	0
8	0.57	389bc3fa690240e798340f5a15918d5c	0.26	0.09	0	1	0
12	0.48	2eeac8d8feae4a8cad5a6af0499a211d	0.23	0.14	0	1	0

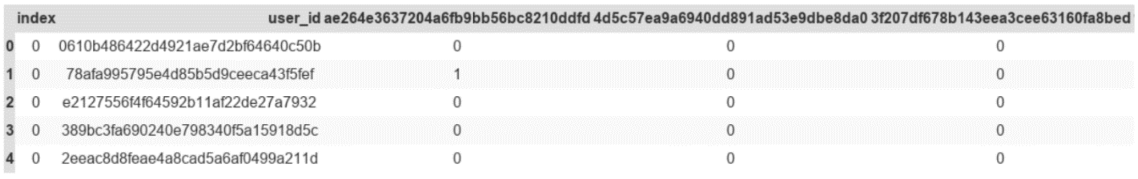
For the labels 3 encodeds are going to be made:

1. A one hot encoded user with the completed offers



	index	user_id	ae264e3637204a6fb9bb56bc8210ddfd	4d5c57ea9a6940dd891ad53e9dbe8da0	3f207df678b143eea3cee63160fa8bed
0	0	0610b486422d4921ae7d2bf64640c50b	0	0	0
1	0	78afa995795e4d85b5d9ceeca43f5fef	1	0	0
2	0	e2127556f4f64592b11af22de27a7932	0	0	0
3	0	389bc3fa690240e798340f5a15918d5c	0	0	0
4	0	2eeac8d8feae4a8cad5a6af0499a211d	0	0	0

1.1 The one hot encoded has also the number of completed offers (the value can be higher than 1)

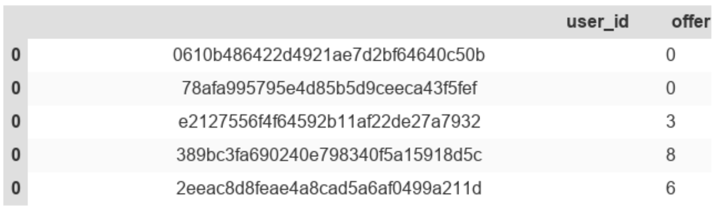


	index	user_id	ae264e3637204a6fb9bb56bc8210ddfd	4d5c57ea9a6940dd891ad53e9dbe8da0	3f207df678b143eea3cee63160fa8bed
0	0	0610b486422d4921ae7d2bf64640c50b	0	0	0
1	0	78afa995795e4d85b5d9ceeca43f5fef	1	0	0
2	0	e2127556f4f64592b11af22de27a7932	0	0	0
3	0	389bc3fa690240e798340f5a15918d5c	0	0	0
4	0	2eeac8d8feae4a8cad5a6af0499a211d	0	0	0

2. The label with the maximum reward

For each user you get the completed offers for those offers you get the rewards the label is choose with the max of all completed rewards

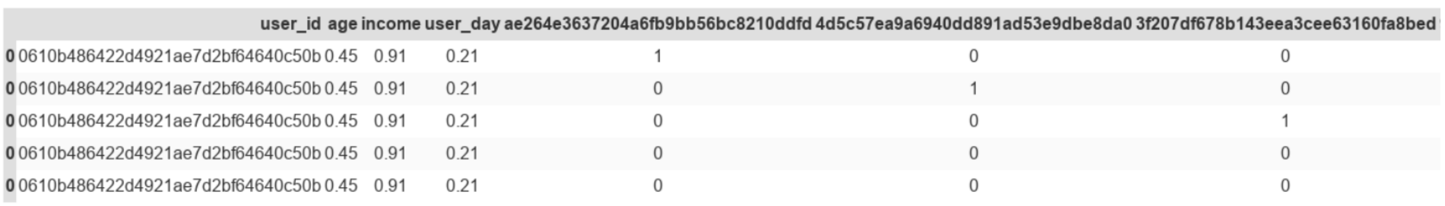
For example, if an offer with a reward of 5 is completed 2 times it will be choose before a completed offer with a reward of 5 completed 1 time



	user_id	offer
0	0610b486422d4921ae7d2bf64640c50b	0
0	78afa995795e4d85b5d9ceeca43f5fef	0
0	e2127556f4f64592b11af22de27a7932	3
0	389bc3fa690240e798340f5a15918d5c	8
0	2eeac8d8feae4a8cad5a6af0499a211d	6

As the previous datasets were not giving good results, the problem and the input were re-engineered. In this case, merging the offers data in the independent variables along with the demographic features is chosen so a binary classifier is made, this classifier specifies either the offer will be successful (viewed and completed) or not successful (ignored).

3. Demographic features + hot-encoded completed offers



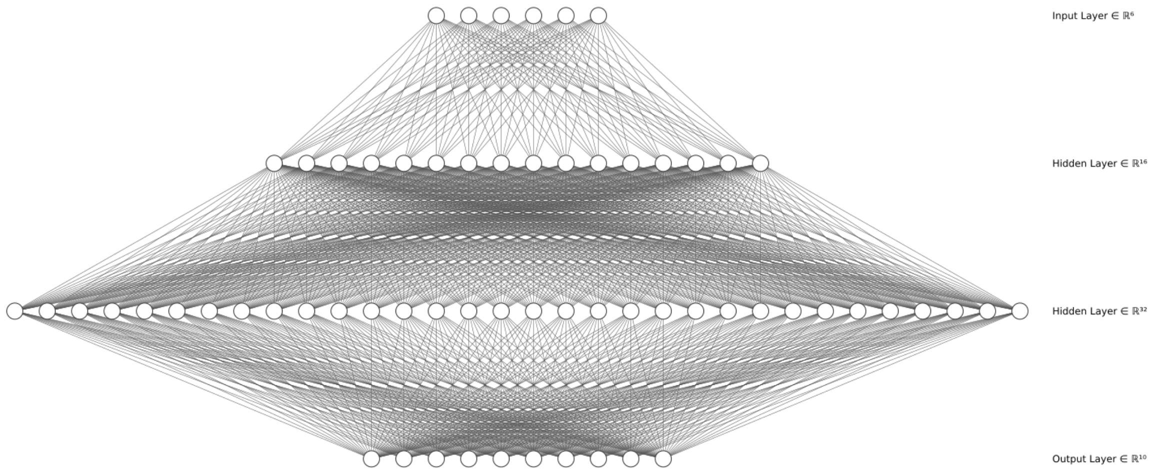
	user_id	age	income	user_day	ae264e3637204a6fb9bb56bc8210ddfd	4d5c57ea9a6940dd891ad53e9dbe8da0	3f207df678b143eea3cee63160fa8bed
0	0610b486422d4921ae7d2bf64640c50b	0.45	0.91	0.21	1	0	0
0	0610b486422d4921ae7d2bf64640c50b	0.45	0.91	0.21	0	1	0
0	0610b486422d4921ae7d2bf64640c50b	0.45	0.91	0.21	0	0	1
0	0610b486422d4921ae7d2bf64640c50b	0.45	0.91	0.21	0	0	0
0	0610b486422d4921ae7d2bf64640c50b	0.45	0.91	0.21	0	0	0

Implementation

The architectures are split into the different classification problems.

<u>Multi-label</u>
Custom ANN (pytorch implementation):

6 inputs 2 hidden layers (16,32) and 10 oputputs as there is 10 different offers

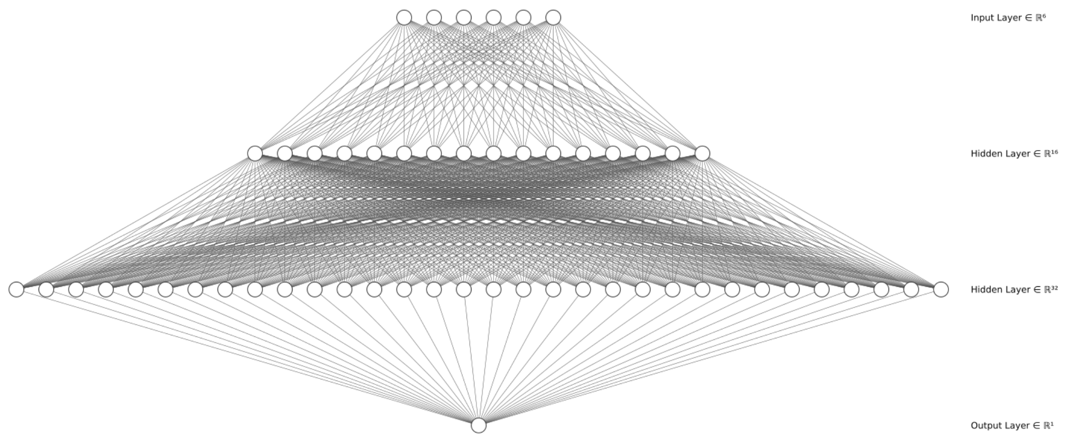


<u>Scikit</u>

Naive Bayes + BinaryRelevance
Naive Bayes + LabelPowerset
Naive Bayes + MLkNN
Naive Bayes + ClassifierChain

<u>Multiclass</u>
Custom ANN (pytorch implementation)

6 inputs 2 hidden layers (16,32) and 1 outputs the top rewarded offer by user



Xgboost Classifier

- Objective → multi:softmax

<u>Binary Clasifier</u>

Xgboost Classifier

- Objective → binary:logistic

Refinement

At first the goal was to make a multi-label classification, so a deep learning model was built. The model didn't perform well so a fine-tuning mas made with no results.

As there weren't any improvements other algorithms like naïve bayes with some implementations in scikit were tried with no results, so the problem was rethink as a multiclass problem were neither the deep learning approach neither the Xgboost classifier did well.

So, the problem was re-engineering by merging the offers data in the independent variables along with the demographic features and make it a binary classification that will solve if the user will complete the offer or not. This problem was solved using Xgboost Classifier.

IV. Results

Model Evaluation and Validation

Multi-label

Custom ANN (pytorch implementation): BCE Test loss: 0.29830405286008266

Naive Bayes

BinaryRelevance

	precision	recall	f1-score	support
avg / total	0.12	0.01	0.02	3585

LabelPowerset

	precision	recall	f1-score	support
avg / total	0.17	0.51	0.25	3585

MLkNN

	precision	recall	f1-score	support
avg / total	0.14	0.00	0.00	3585

ClassifierChain

	precision	recall	f1-score	support
avg / total	0.16	0.01	0.03	3585

Multiclass
Custom ANN (pytorch implementation): BCE Test loss: 0.45583873448677675

Xgboost Classifier

	precision	recall	f1-score	support
avg / total	0.29	0.40	0.25	2965

Binary Clasifier

Xgboost Classifier

	precision	recall	f1-score	support
avg / total	0.77	0.88	0.82	29650

V. Conclusion

The models do not perform well in this classification task using chosen input, although all classifiers give the same score

For the ANN an exploratory could been made with the **loss function and the activation function used in the output layer**.

For this kind of problems **traditional machine learning algorithms such as gradient boosting techniques (XGBoost, LightGBM..etc) will get better results and performance.**

If you have issues related to the solution to the problem a **re-engineer should be considered** in this case the problem was changed to a **binary classifier** that tells if a certain user (**only using its demographic values**) will complete the offer or not.

Bibliography

[1]"torch.nn — PyTorch master documentation", *Pytorch.org*, 2020. [Online]. Available: https://pytorch.org/docs/stable/nn.html. [Accessed: 02- Jan- 2020]

[2]"3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.1 documentation", *Scikit-learn.org*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#log-loss. [Accessed: 02- Jan- 2020]

[3]"sklearn.metrics.accuracy_score — scikit-learn 0.22.1 documentation", *Scikit-learn.org*, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html. [Accessed: 02- Jan- 2020]

[4]"scikit-multilearn | Multi-label classification package for python", *Scikit.ml*, 2020. [Online]. Available: http://scikit.ml/api/skmultilearn.problem_transform.br.html. [Accessed: 02- Jan- 2020]

[5]"Classification with Neural Networks: Is it the Right Choice? - MissingLink.ai", *MissingLink.ai*, 2020. [Online]. Available: https://missinglink.ai/guides/neural-network-concepts/classification-neural-networks-neural-network-right-choice/. [Accessed: 02- Jan- 2020]