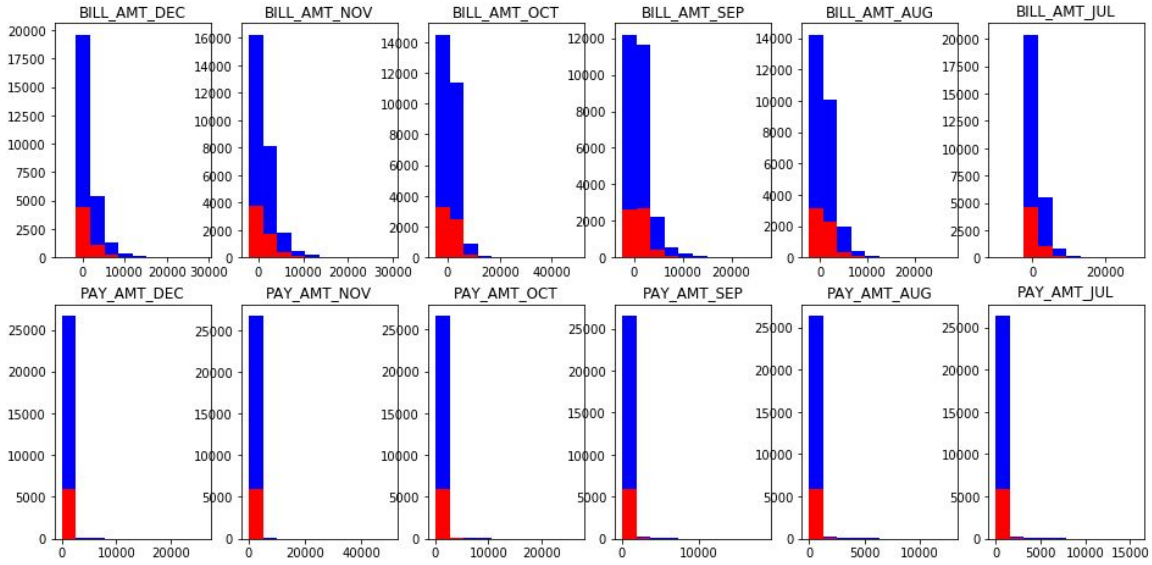
Team Jago - Data Mining 2017 project

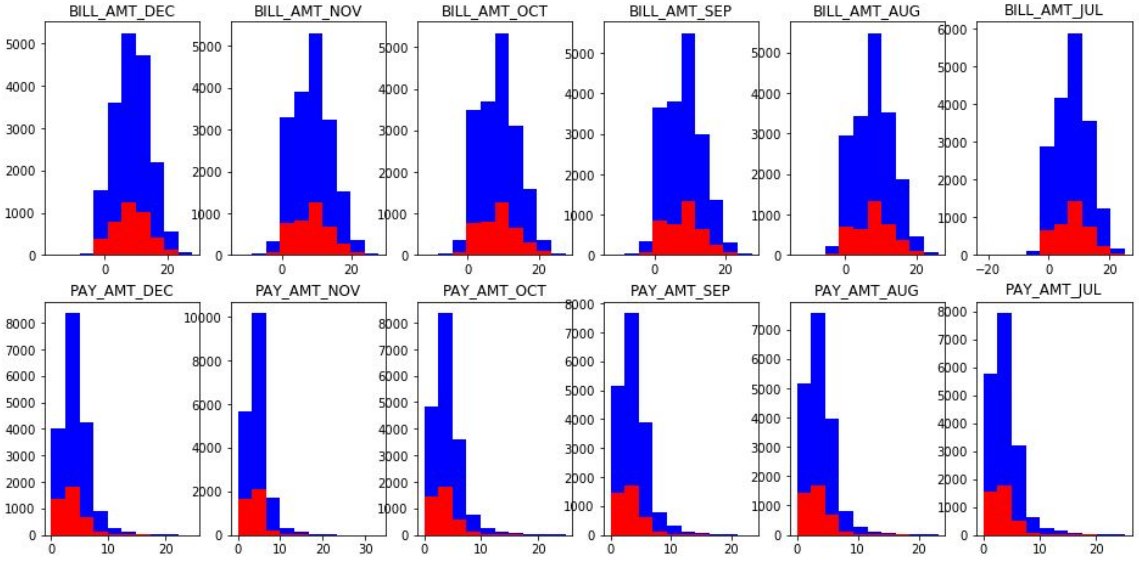
Team members:
Crippa Mattia -- 10397252
Tran Khanh Huy Paolo -- 10401830
Pirovano Alberto Mario -- 10396610
Vetere Alessandro -- 10425802

Sklearn approach

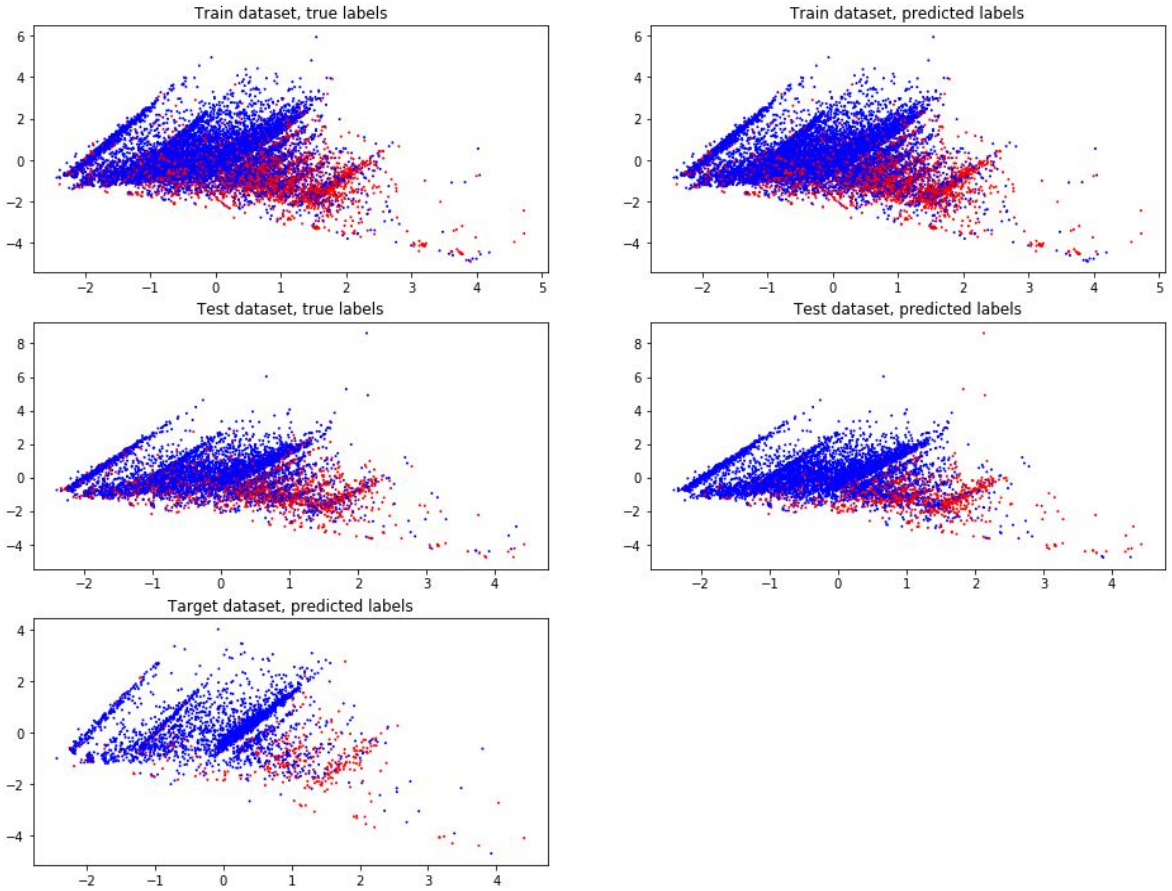
We used sklearn with pandas, numpy, matplotlib and xgboost to develop the solution for the given problem.
First of all we analyzed the distribution of the data with respect to the target and we found that many of the variables by themselves were non discriminative and highly skewed.



Then we proceeded to split the dataset 67% train, 33% test using a stratified approach with respect to the target variable.
Test dataset from now on were not used anymore to take decisions regarding what to do with the data.
We then tried to fix the skewed distribution of the above mentioned variables LIMIT_BAL, BILL_AMT_*, PAY_AMT_*: usual tricks like doing square root or logarithm wouldn't work in this case given that we had negative values, so we used the cubic root instead, and the results were satisfying enough.

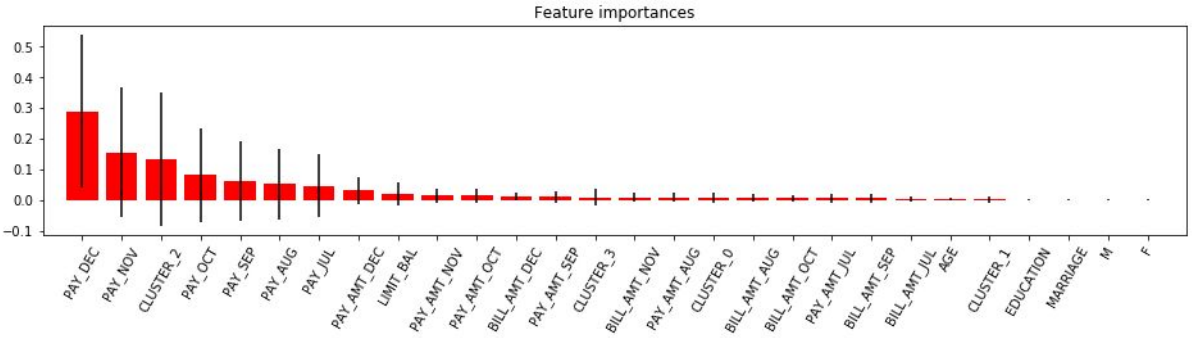


We transformed BIRTH_DATE to an AGE variable, filling with the median AGE the missing values.
We transformed EDUCATION into a numeric variable, as well as MARRIAGE.
We one hot encoded SEX into M and F.
We didn't one hot encode the other variables to preserve the ordering present in such variables.
Then, we scaled the data using RobustScaler from sklearn which scales the data using statistics robust to outliers.
Furthermore we did PCA into 2 components for visualizing purpose, using a RandomForest to do preliminary predictions and see how the performance were:



Here it is clear why RandomForest is one of the best algorithm for this problem, given the high non-linearity of class boundary.
Also, it's clear that target dataset resembles the structure of the one we are using for train and test: this is good as we don't know anything about the value of the target variable for target dataset, but from this preliminary analysis it is clear that the distribution of the data is similar to the one we will train algorithms to.
But it is also clear that PCA doesn't help in making more clear the boundary between classes: so we didn't use PCA to train algorithms.
The last thing we did to data was doing unsupervised clustering to data: we one hot encoded the cluster found by KMeans and added these columns to the dataset.
To evaluate the algorithms we used stratified 10 folds cross validation.
We used only probabilistic classifiers to enable finding the threshold which maximized the mean F1 score in cross validation.
The algorithms we used so far are:

Model	Threshold	F1 Cross Validation	F1 Test
Decision Tree	0.25	0.524 ± 0.024	0.525
Gaussian Naive Bayes	0.48	0.522 ± 0.016	0.528
Random Forest	0.24	0.545 ± 0.022	0.547
K Neighbors Classifier	0.24	0.531 ± 0.021	0.526
Multi Layer Perceptron	0.29	0.541 ± 0.022	0.542
Logistic Regression	0.25	0.525 ± 0.025	0.519
XGBoost	0.27	0.547 ± 0.023	0.547
Linear Discriminant Analysis	0.21	0.524 ± 0.026	0.516
Quadratic Discriminant Analysis	0.30	0.530 ± 0.022	0.524
Soft Voting Ensemble (RF, XGB, MLP)	0.24	0.548 ± 0.022	0.549



This plot show the feature importances for the Random Forest Classifier and it shows that the most important features are the same already identified by the bank, PAY_*. We didn't find new important variables, except for the unsupervised CLUSTER_2.
For the final delivery we used the Soft Voting Ensemble of Random Forest, XGBoost and Multi Layer Perceptron with custom weights.
This is mainly not because of the F1 score of such algorithm, but hoping to give further stability to the predictions made on target dataset.

KNIME approach

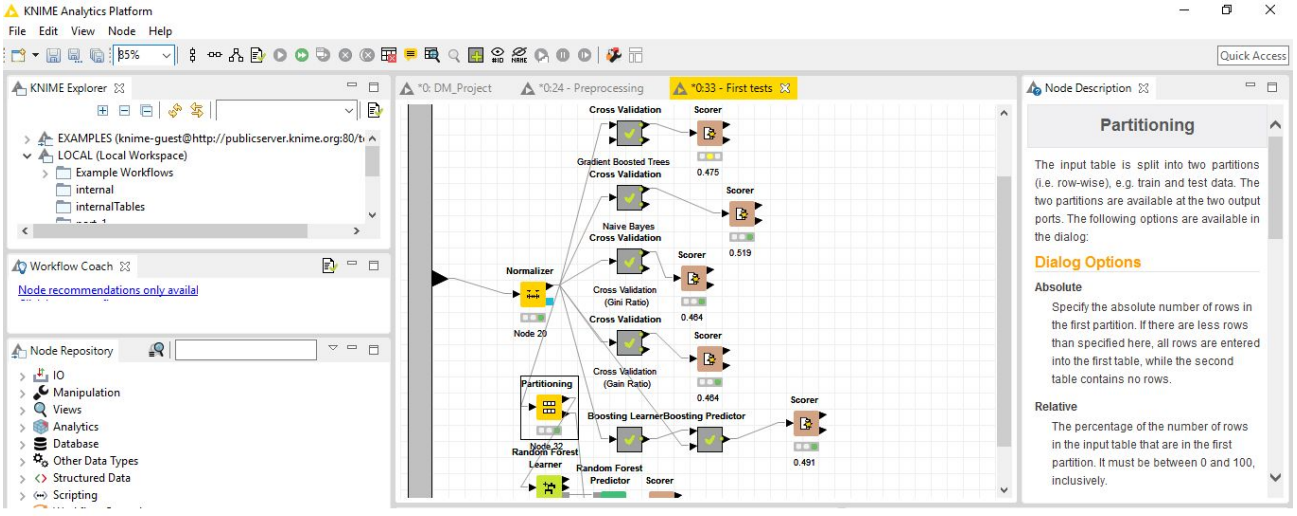
We tried to use KNIME in order to see if there was any difference between its algorithms results and the ones made by the Python libraries.
First of all we made some preprocessing on the available data, in particular:

- Converting the birth date into the age
- Fill the missing value with the average/median/linear regression value
- Round the values
- Filter out some columns after doing a PCA evaluation on the data

Then we proceed to normalize the remaining data in order to reduce the variance of the values. (Normalization was made with max-min algorithm and with the gaussian function as well).
After the preprocessing part we proceed to use the various algorithm taught in class, in particular:

- Gradient Boosted Trees
- Naive Bayes
- Decision Trees with Gini Index
- Decision Trees with Gain Ratio
- Boosting learner with Naive Bayes and with Decision Trees
- Random Forest

We use the various learner with stratified cross validation. Since the KNIME cross validation node was keeping a very small subset as a test set, we proceed to use a partitioner in order to increase the dimension of the test set up to 20% of the entire dataset.



Screenshot of the KNIME workflow

Unfortunately we found similar results between preprocessed data and not preprocessed data. The most important features were the PAY_MONTH features generally speaking.

Deep learning approach

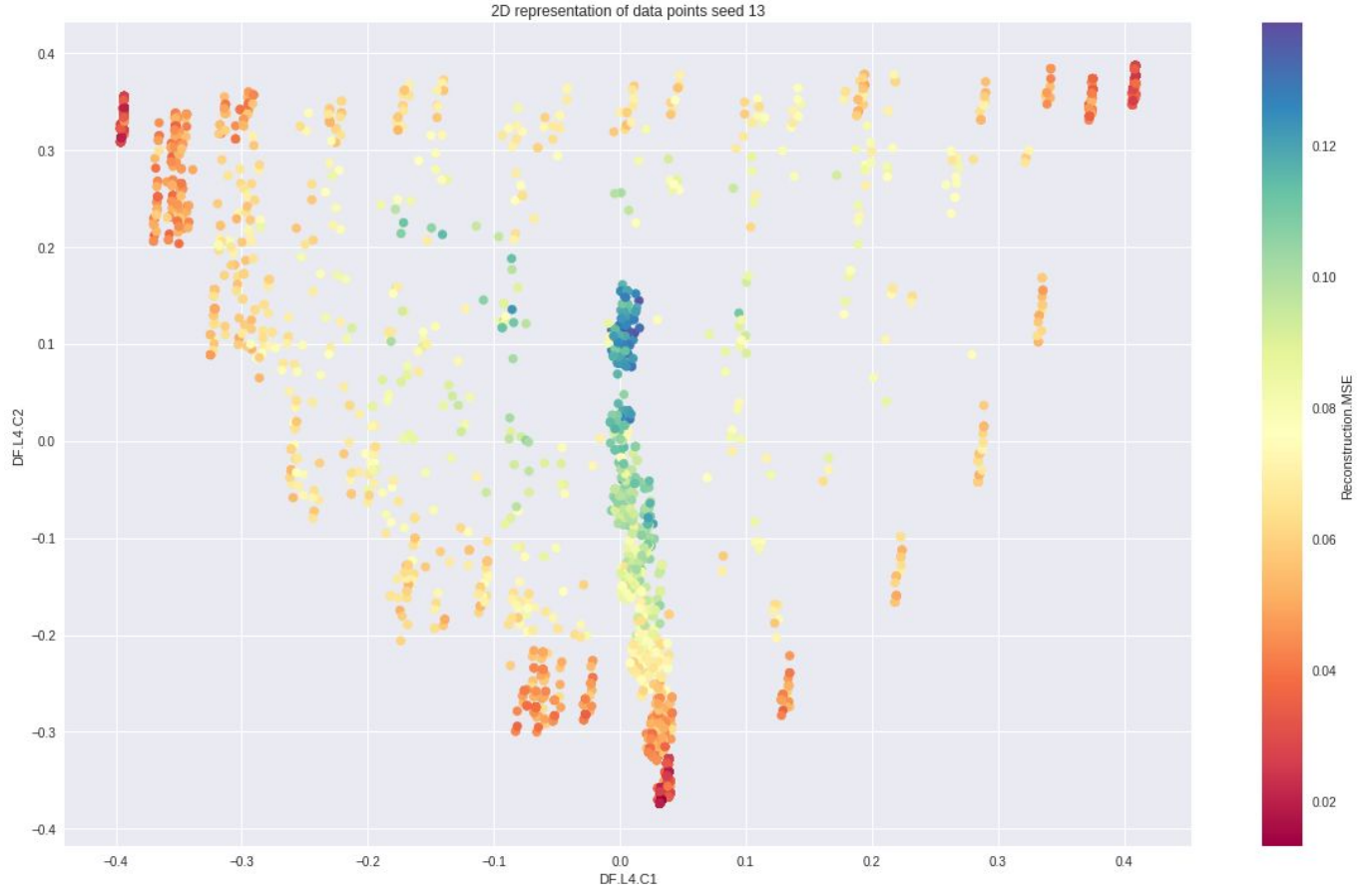
We also tried another edge approach for addressing this kind of outliers detection problem.
In literature a great number of papers talk about deep learning based solutions to classify data points in an anomaly detection environment, some of them using RNNs, others using deep autoencoders.
For the strict time constraint, we experimented only the second approach.

The main idea behind it is to design an auto encoder that is trained in an unsupervised way with samples from the training set belonging to the majority class, in our case the customers being labeled with "0".
This way the network is trained to minimize the reconstruction error of the samples that sees during the training phase and so it is forced to learn how to internally encode their representation.
During the inference stage we ask the network to encode a stratified test set.
For each sample we check the reconstruction MSE and, if the encoder has really catched how to encode the samples, we expect that samples with reconstruction MSE equal to 0 belong to the majority class and the samples with MSE different from 0 come from the minority class.
By inspecting the magnitude of the reconstruction MSE we can also see if there are samples belonging to the same anomaly class, for example samples with reconstruction MSE near to 4 can belong to anomaly_1 and the ones that have reconstruction error near to 10 can be considered to belong to a different anomaly_2.
Practically we finally have to tune a threshold in order to define a rule to classify samples.

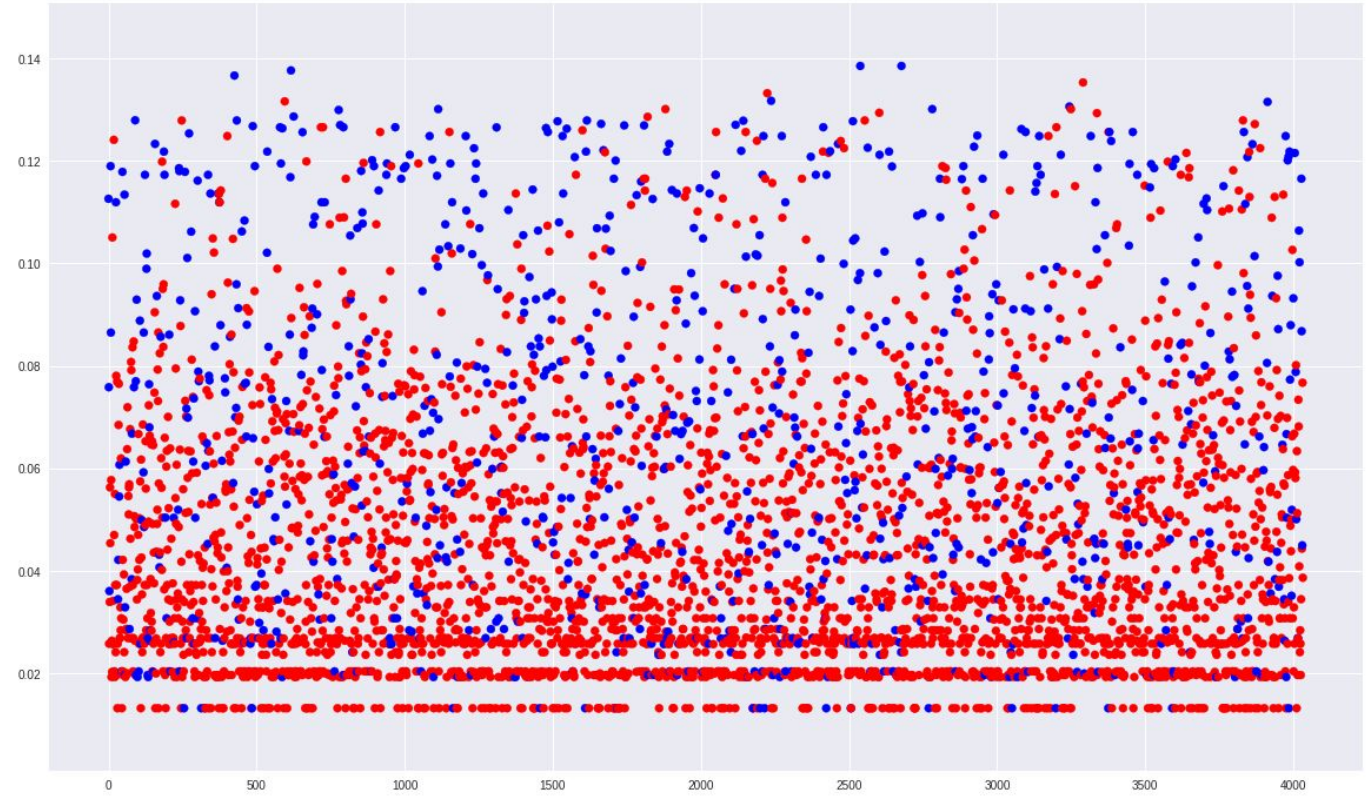
The configuration that has given the best performances in terms of f1 measure is the following:

- 7 hidden layers of 16,8,4,2,4,8,16 hidden units respectively
- L1 regularization = 1e-1
- L2 regularization = 1e-1
- tanh activation function
- 300 epochs

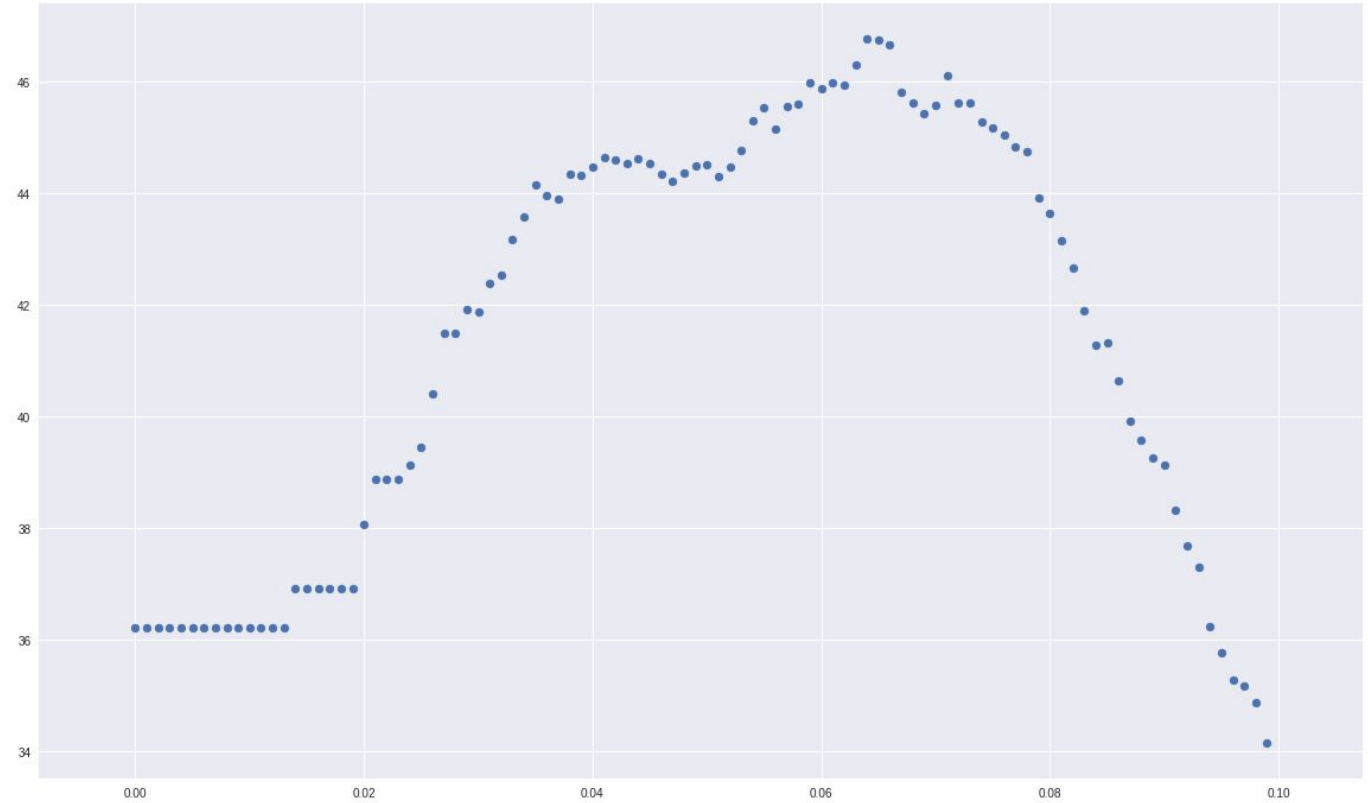
We can inspect the test set plotted in the 2 hidden dimensions of the 2 layer (starting from layer_0). The color map on the right assigns a color to each point related to its MSE, high MSE means blue point, low MSE means red point:



We can now plot the MSE for each sample assigning blue color to samples belonging to the minority class and red to samples coming from the majority class:



As we can see the bottom of the plot has mainly samples from the "0" class and the top instead has mainly data points from the minority class. The middle part of it is mashed and we believe that having a better representation of data and more data we can separate the dataset even better.
The search for the threshold has been performed by testing each MSE inside the range between 0.00 and 0.1:



As we can see the best threshold gives an f1 = 0.47.

The obtained performances are neither completely satisfactory nor so bad to suggest that the approach is not suitable for this problem.
In fact the quality of a data science problem is strongly dependent on the data cleaning and data reshaping and we believe that a neural network approach is even more dependent from this stage.
We claim also that the main reasons for having that value of the performance is the shape of data that we used and the little amount of data that was available.
For this reason we tried some data transformation such as temporal aggregation from several points of views and we generated six variations of the dataset.
Even with these transformations the performances remained stable on that value and so we claimed that we didn't find the right data transformation for make data really meaningful.
In order to eliminate the noise we decided to train the encoder retaining only the One Hot Encoded version of the columns 'PAY_MONTH".
We decided this way because we saw that the other columns, also combined in some fancy ways, are not useful to predict the target.
The parameters tuning phase was done by try and retry given that it was computationally too expensive to grid tune them.