Team Jago
Data Mining 2017 project

Sklearn, KNIME, H2O

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

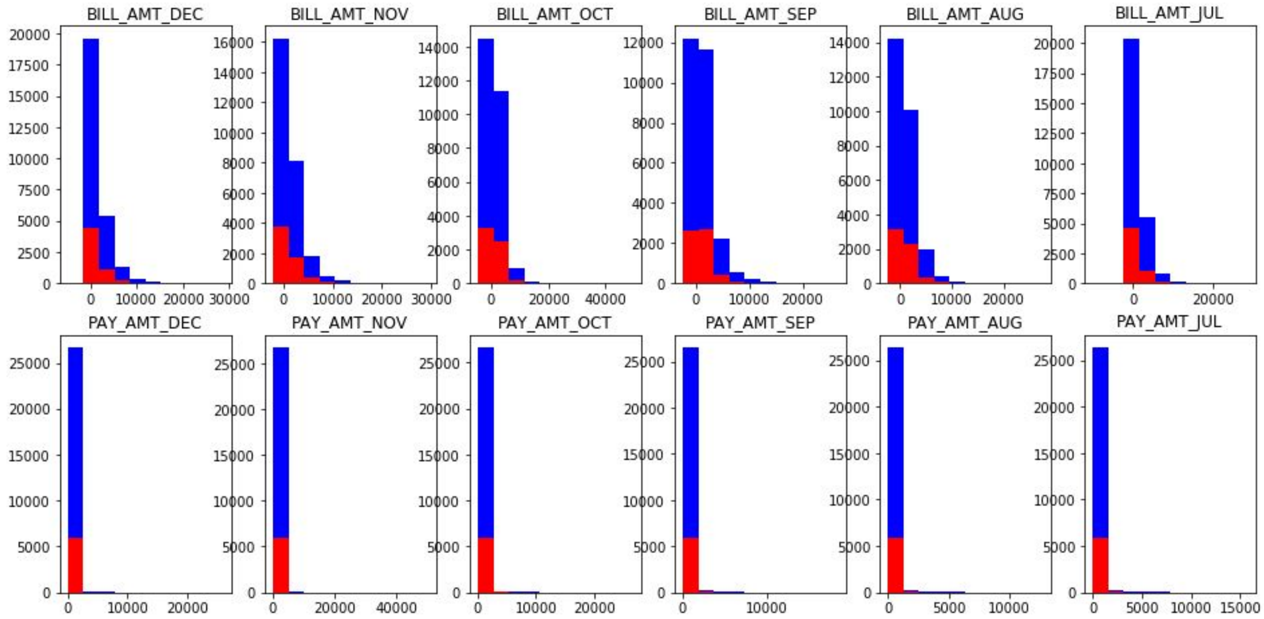
Sklearn approach

Sklearn was used along with:

- pandas
- numpy
- matplotlib
- xgboost

Sklearn approach: Data analysis and exploration

- Variables distribution w.r.t. target variable



- High skew
- Non discriminative

Sklearn approach: Train test split

Stratified train test split:

- 67% train set
- 33% test set

From now on, test set is not used to make decisions regarding both data processing and algorithms tuning.

Sklearn approach: Fixing variables

Problematic variables:

- PAY_AMT_*
- BILL_AMT_*

Problems:

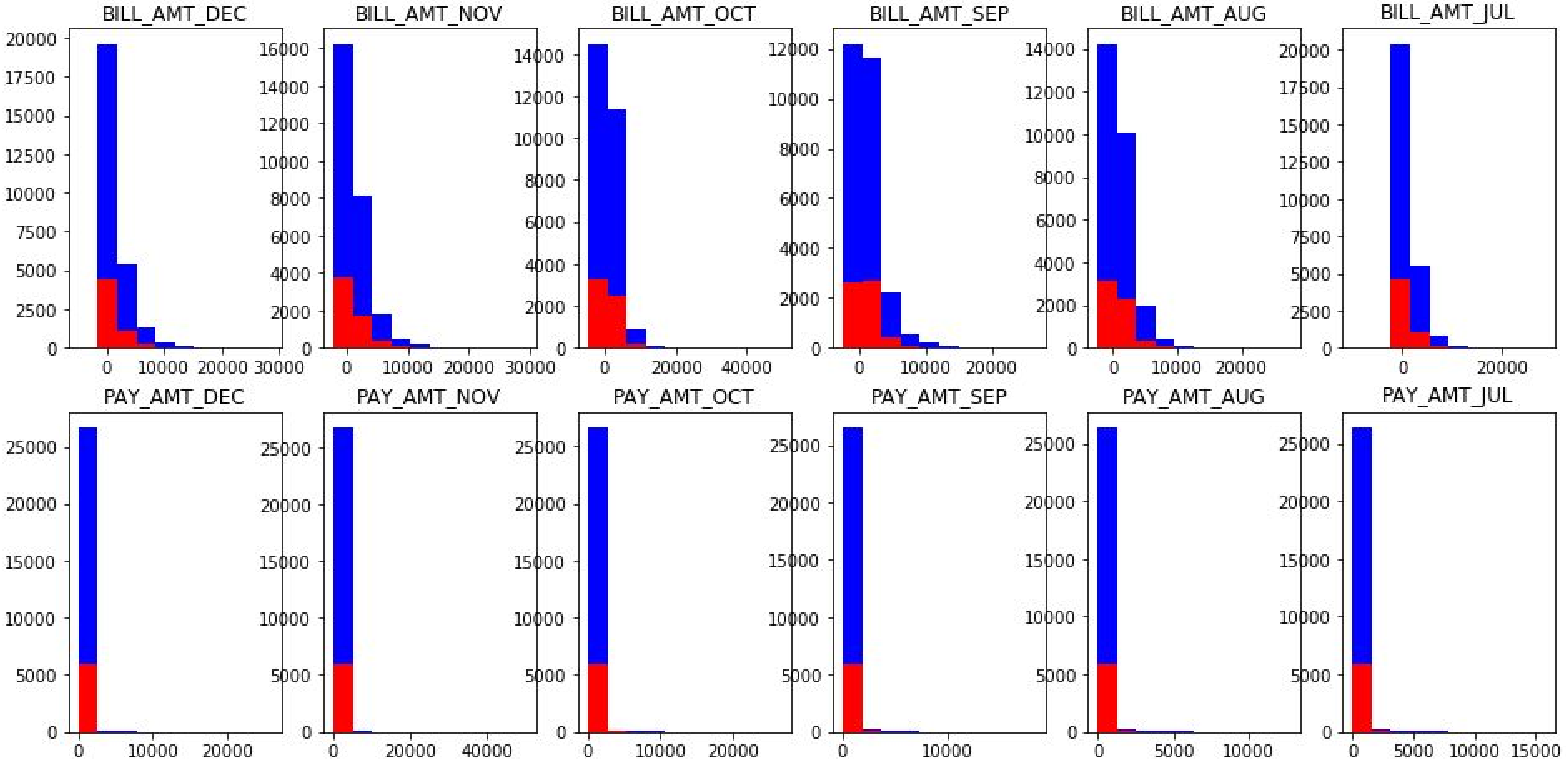
- High skew
- Both positive and negative values (no square root nor logarithm trick)
- Non discriminative (no fix possible at this stage)

Sklearn approach: Fixing variables cont’d

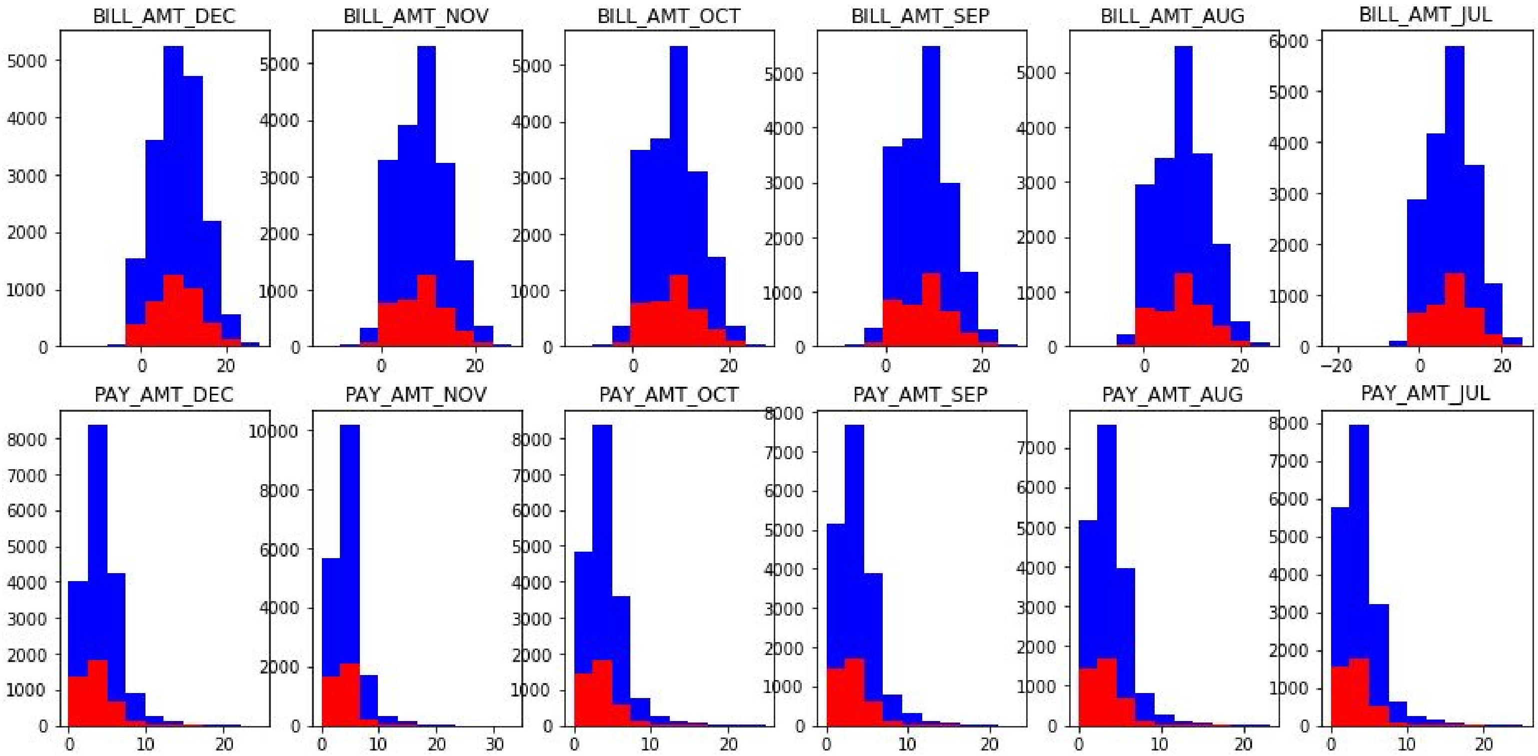
To fix skewness with negative and positive values we used:

- Cubic root

Sklearn approach: before cubic root



Sklearn approach: after cubic root



Sklearn approach: labels

Labeled variables:

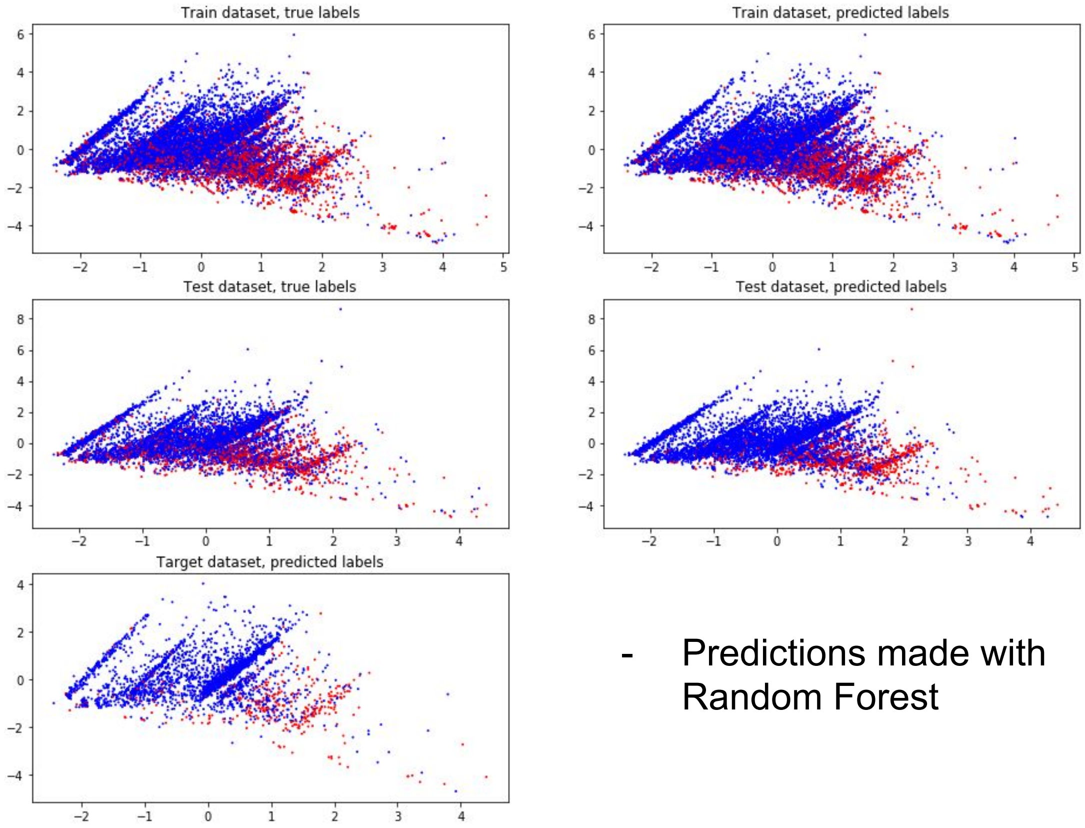
- SEX → One hot encoded
- EDUCATION → Converted to numeric variable
- MARRIAGE → Converted to numeric variable
- BIRTH_DATE → Converted to AGE variable and filled missing values with median

Sklearn approach: RobustScaler

RobustScaler used to normalize data:

- Statistics robust to outliers

Sklearn approach: visualize data using PCA



- Predictions made with Random Forest

Sklearn approach: KMeans clustering

KMeans used to find 4 clusters:

- Cluster then one hot encoded the into CLUSTER_* variables and added to the dataset
- Unsupervised approach

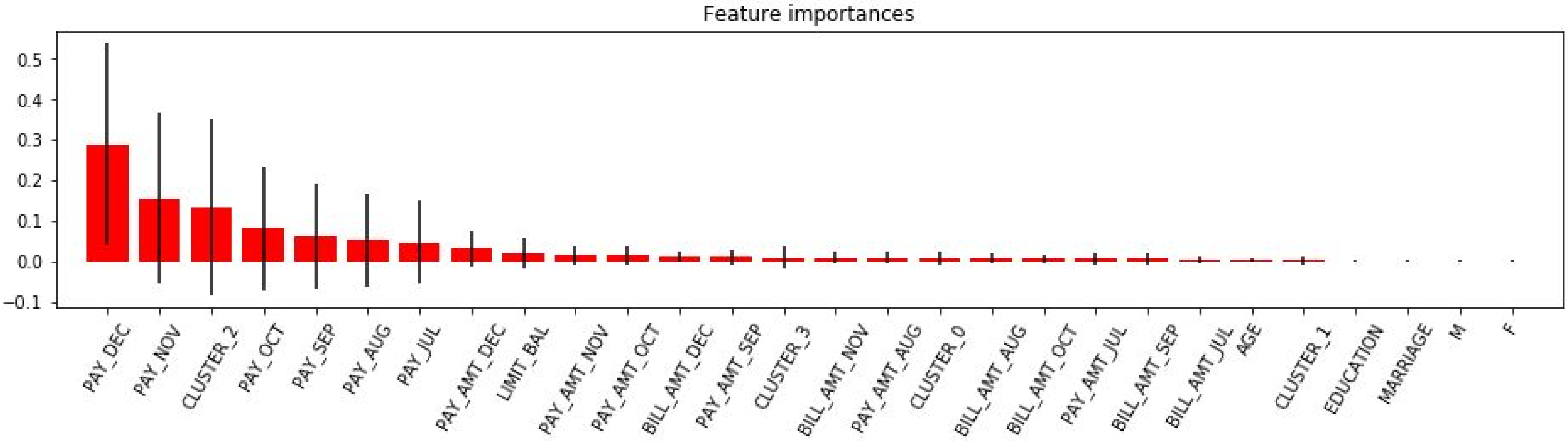
Sklearn approach: Stratified 10 fold CV

To evaluate algorithms we used stratified 10 fold cross validation

- Stratification necessary to face unbalanced classes
- 10 fold gives stability to predicted scores

Sklearn approach: variables importance

Random Forest results:



Sklearn approach: Models and Scores

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
<u>*Random Forest*</u>	<u>0.24</u>	<u>0.545 ± 0.022</u>	<u>0.547</u>
K Neighbors Classifier	0.24	0.531 ± 0.021	0.526
<u>*Multi Layer Perceptron*</u>	<u>0.29</u>	<u>0.541 ± 0.022</u>	<u>0.542</u>
Logistic Regression	0.25	0.525 ± 0.025	0.519
<u>*XGBoost*</u>	<u>0.27</u>	<u>0.547 ± 0.023</u>	<u>0.547</u>
Linear Discriminant Analysis	0.21	0.524 ± 0.026	0.516
Quadratic Discriminant Analysis	0.30	0.530 ± 0.022	0.524
<u>*Soft Voting Ensemble (RF, XGB, MLP)*</u>	<u>0.24</u>	<u>0.548 ± 0.022</u>	<u>0.549</u>

Sklearn approach: algorithm of choice

Soft Voting Classifier of:

- Random Forest (w = 0.4)
  - criterion = ‘gini’, max_depth = 4, n_estimators = 1000, max_features = 4, oob_score = True, min_samples_leaf = 20, max_leaf_nodes = 20
- Multi Layer Perceptron (w = 0.2)
  - hidden_layer_size = (14), activation = ‘logistic’, alpha = 0.001
- XGBoost (w = 0.4)
  - reg_lambda = 0.01

Sklearn approach: final scores

Threshold = 0.24

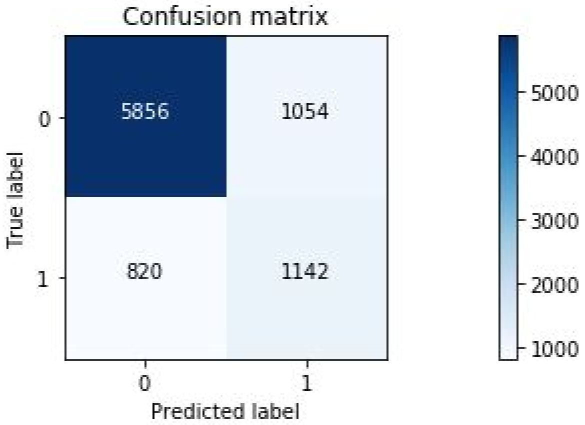
Cross validation

- F1 = 0.548 ± 0.022

Test

- F1 = 0.549

```
==== Cross validation report ====
Threshold = 0.24
f1_cv = 0.548 ± 0.022
acc_cv = 0.784 ± 0.012
prec_cv = 0.511 ± 0.024
rec_cv = 0.592 ± 0.027
===================================
======= Test report =============
Threshold = 0.24
```




```
f1 = 0.549
acc = 0.789
prec = 0.520
rec = 0.582
===================================
```

KNIME approach: preprocessing

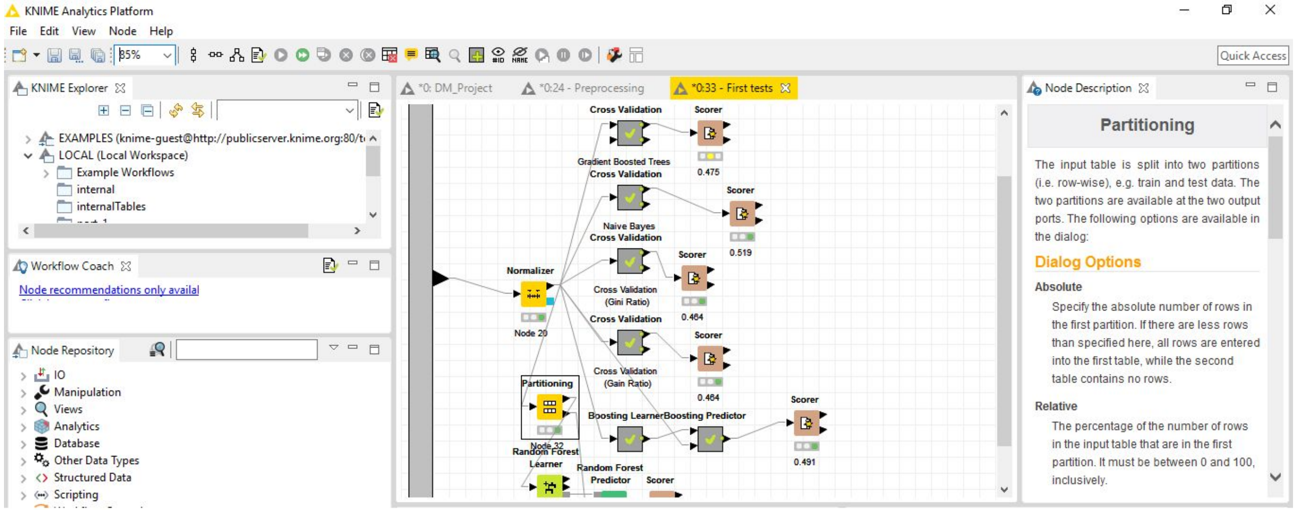
Initial preprocessing of the available data:

- Convert 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 a PCA evaluation on the data

KNIME approach: models

The models we tried were:

- 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



Screenshot of the KNIME workflow

Deep learning approach

Anomaly detection can be addressed using deep learning solutions:

- RNNs
- deep autoencoders

We chose to study and experiment the autoencoders based approach.

Deep learning approach: Training

- The training set is made of the training samples belonging to the majority class.
- The training phase is done in an unsupervised way with the objective of minimizing the reconstruction MSE
- The network learns an encoded representation of the majority class

Deep learning approach: Inference

- At inference phase we ask the autoencoder to process a stratified test set
- We check the reconstruction MSE for each test sample

We expect:

- reconstructionMSE(test_sample) = 0   if "test_sample" belongs to majority class
- reconstructionMSE(test_sample) > 0   if "test_sample" belongs to minority class

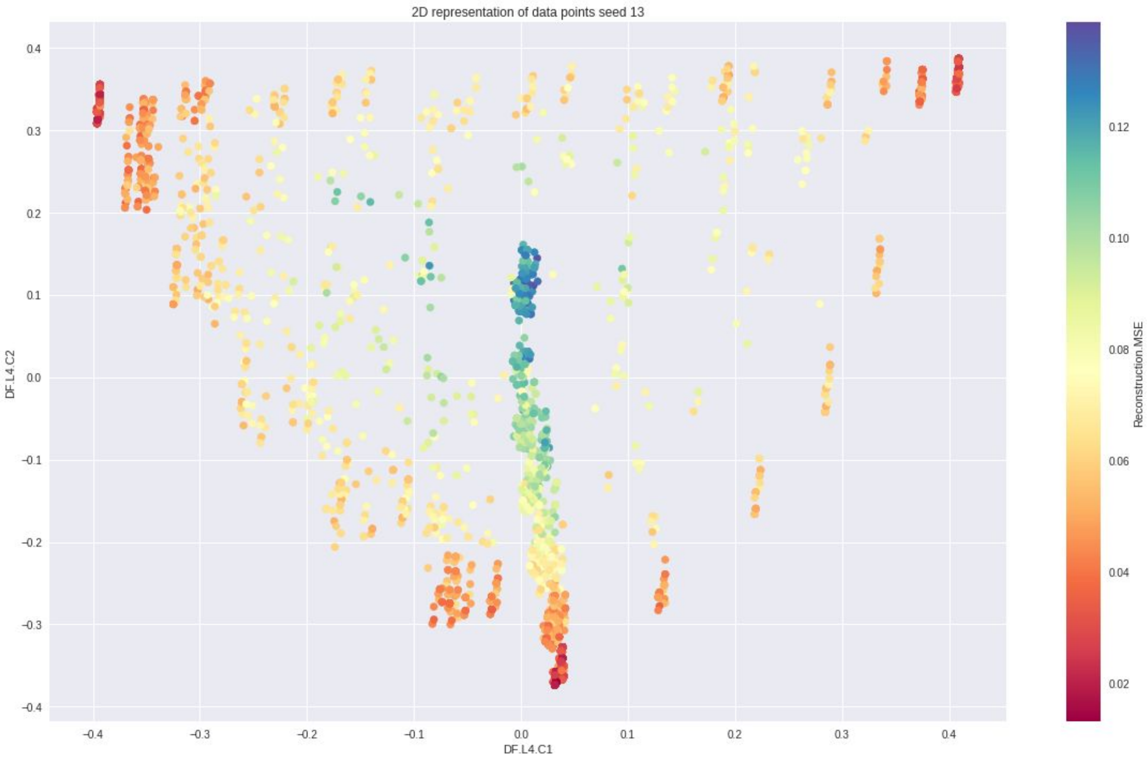
Practically we have to choose a value of reconstructionMSE as the threshold for deciding how to classify data points

Deep learning approach: Best setting

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
- 500 epochs

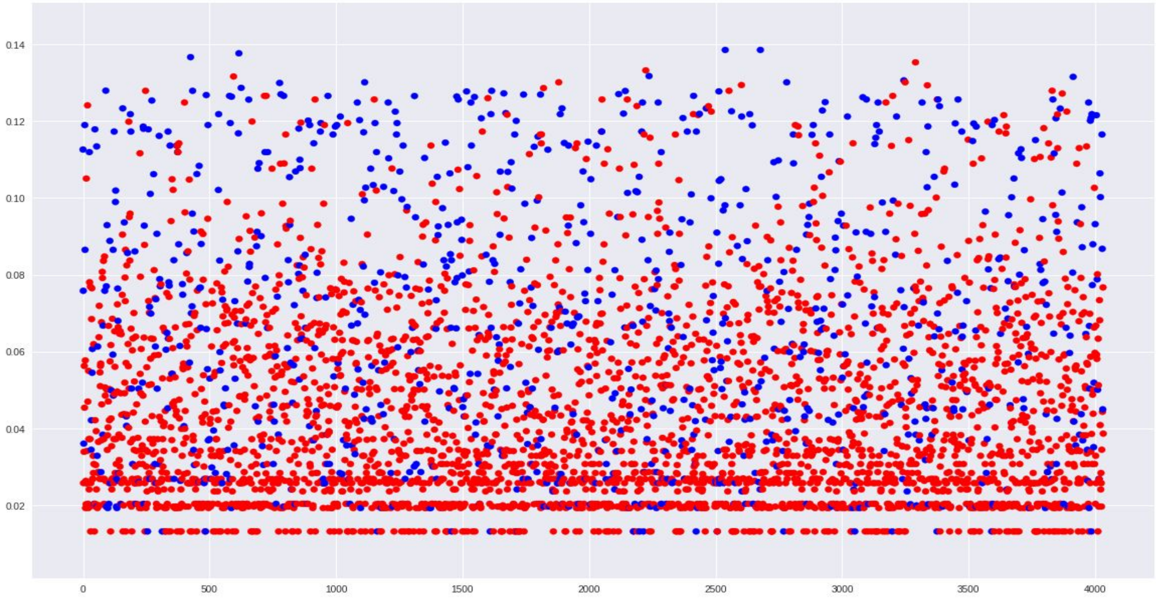
Deep learning approach: 2D hidden representation



We can see the test data points plotted in the 2 hidden dimensions of the 3rd hidden layer.

The color assigned to each point allow us to visualize the reconstruction MSE for each test point.

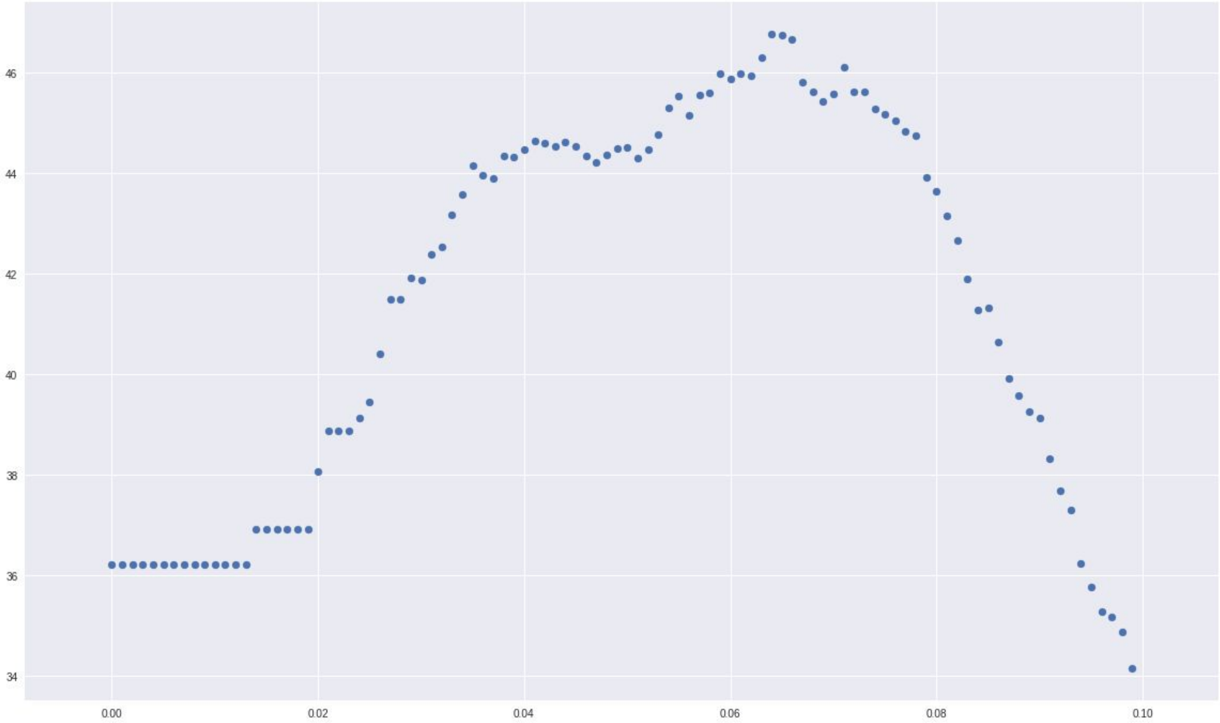
Deep learning approach: MSE



This plot shows 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.

Deep learning approach: Threshold



The search for the threshold has been performed by testing each MSE inside the range between 0.00 and 0.1.

Deep learning approach: Performances

The best performance achieved is f1 = 0.47 with threshold = 0.065.

The only columns used were “PAY_MONTH”, the others were not useful for predicting the target variable.

We believe that the model can be greatly improved by:

- training the model with a bigger training set
- reshaping the dataset in order to make the remaining columns meaningful
- grid tuning the hyper parameters

Deep learning approach: Future work

Another approach is exploiting the 2 dimensional representation generated by the autoencoder and:

- add high order features
- use for example logistic regression