School of Finance Hackathon

Binary classification.
Anomaly detection in financial data.

Team - “Busters”

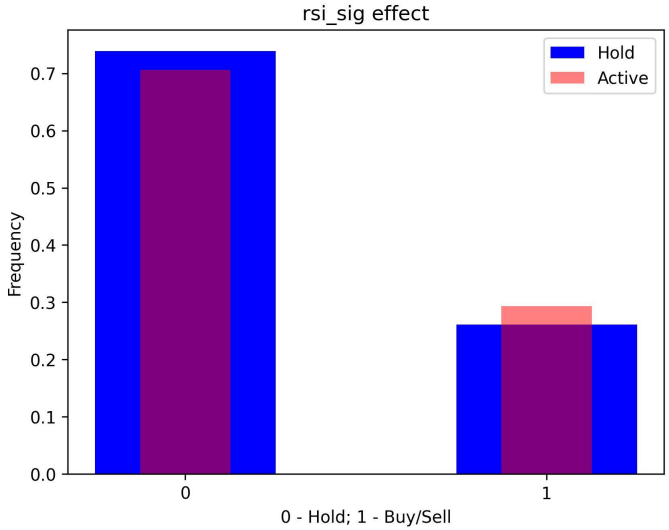
Mikhail Mironov

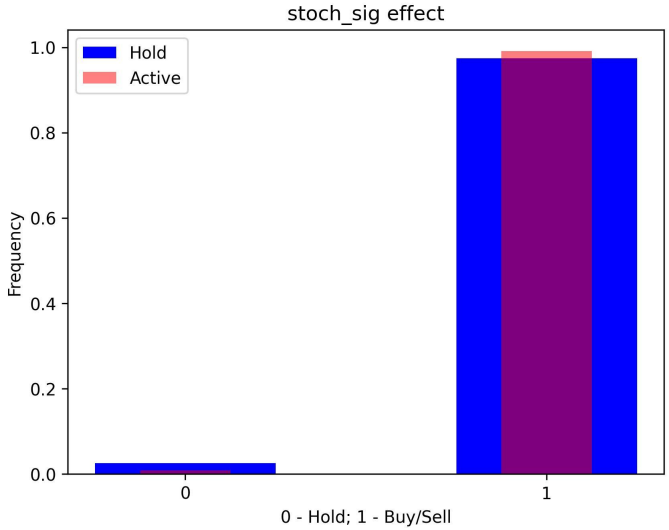
Alexander Illyuk

Github repo with notebooks

Introduction. Data analysis

- The task is to identify to distinguish active trades from hold positions
- The explanatory variables are different technical analysis tools
- The data set organized as the technical indicators and their decision: Hold or Buy/Sell
- The decisions by technical instruments do not explain difference between types of position

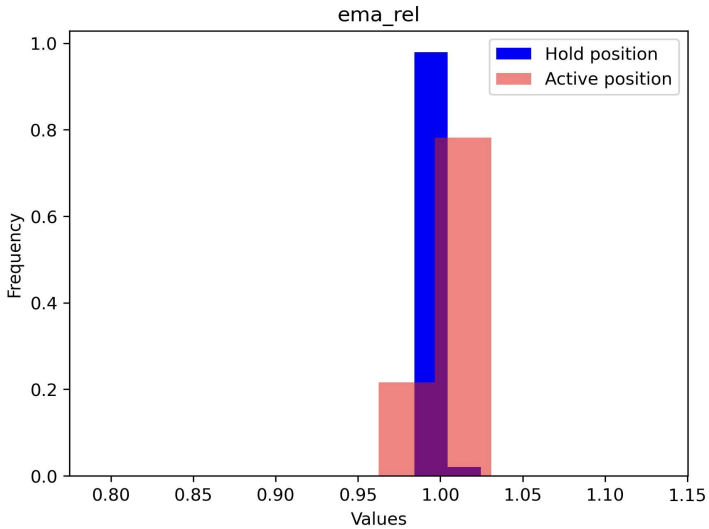


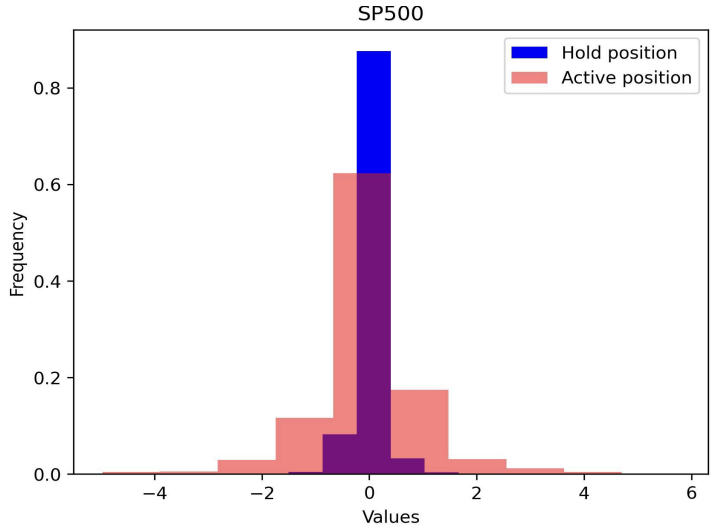


Source:

Introduction. Data analysis

- The numerical explanatory variables have more distinctive features
- Especially, features directly connected with S&P500

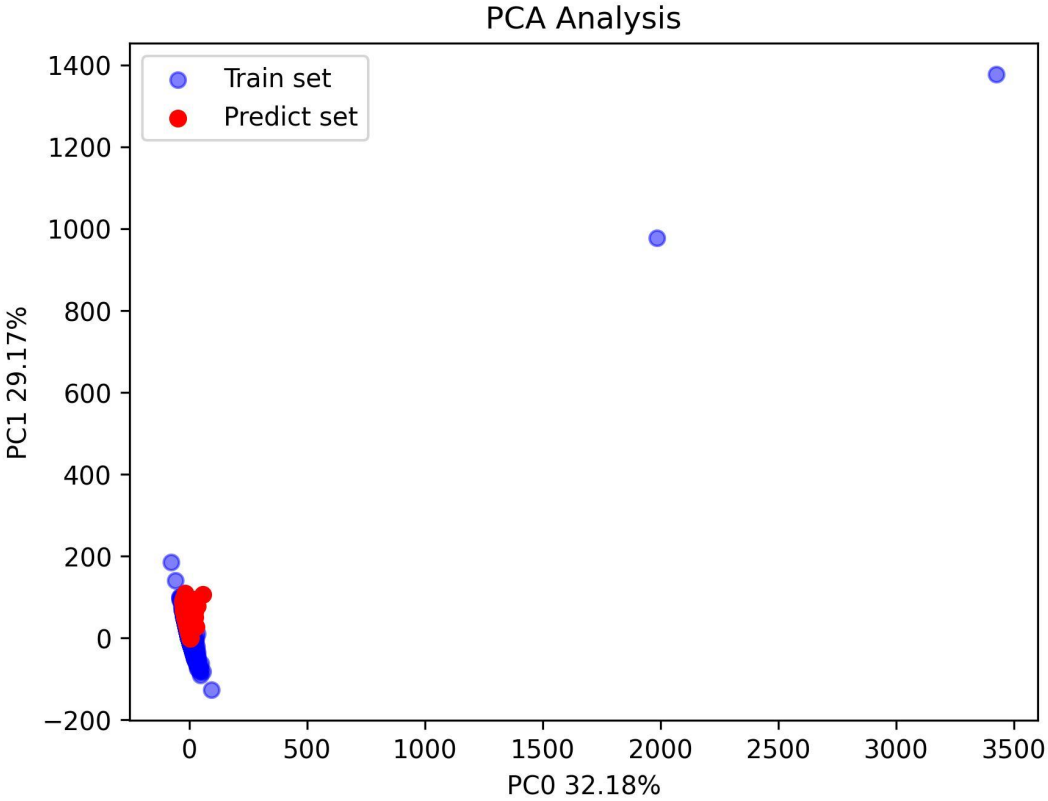




- However, almost all of them suffers from the **outliers** problem
- All Outliers could not be removed from the data
- The submission data has outliers either
- To generally address this problem, PCA analysis is performed

Source:

Principal Components. Outliers

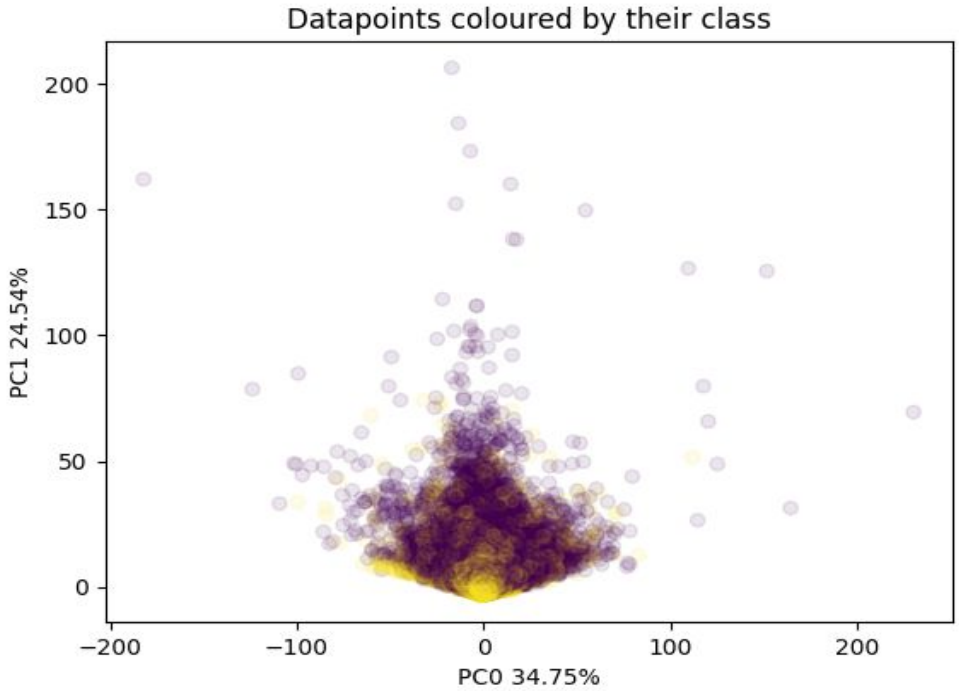


- We have performed PCA (Principal Components) with multiple dimensions. First two components accounted for almost 60% variance of the initial data

- As we see there are two obvious outliers, they offset the whole PCA projection. They will also affect the standardisation, therefore, we should remove them

Source:

Principal Components. Clean projection



- We have removed those two outliers and performed PCA again. This time those two obvious outliers do not offset the projection

- Data is coloured using class labels, we see that **minority class (yellow)** is more inlier class.

Source:

Scaling the data. Outliers removal impact

	feature	corrected_mean	noised_mean	diff_to_scale
25	ichi_b	-0.019874	-0.014857	0.252442
11	pivot_pt	-0.011122	-0.009556	0.140745
27	support1	-1.196541	-1.213082	0.013824
13	resist2	2.364921	2.386163	0.008982
29	resist3	4.740965	4.781883	0.008631

After removing the outliers, we saw that the sample mean as well as the standard deviation changed drastically. By removing the outliers we used better estimates for standardizing the data

Source:

Splitting data. Training a baseline model

- We went for a simple train, validation, test splits (0.7, 0.1, 0.2). Due to imbalanced nature of the data, we should split the data ensuring that the imbalance ratio is preserved among sets.
- As a baseline we went for a vanilla Catboost Classifier with loss function of logloss. We trained the model on the train set and validated using the left out validation set. Below is the result of model on validation set

	precision	recall	f1-score	support
0.0	0.76	0.55	0.64	14764
1.0	0.95	0.98	0.96	121812
accuracy			0.93	136576
macro avg	0.85	0.77	0.80	136576
weighted avg	0.93	0.93	0.93	136576

Source:

Tuning Catboost model with Optuna library

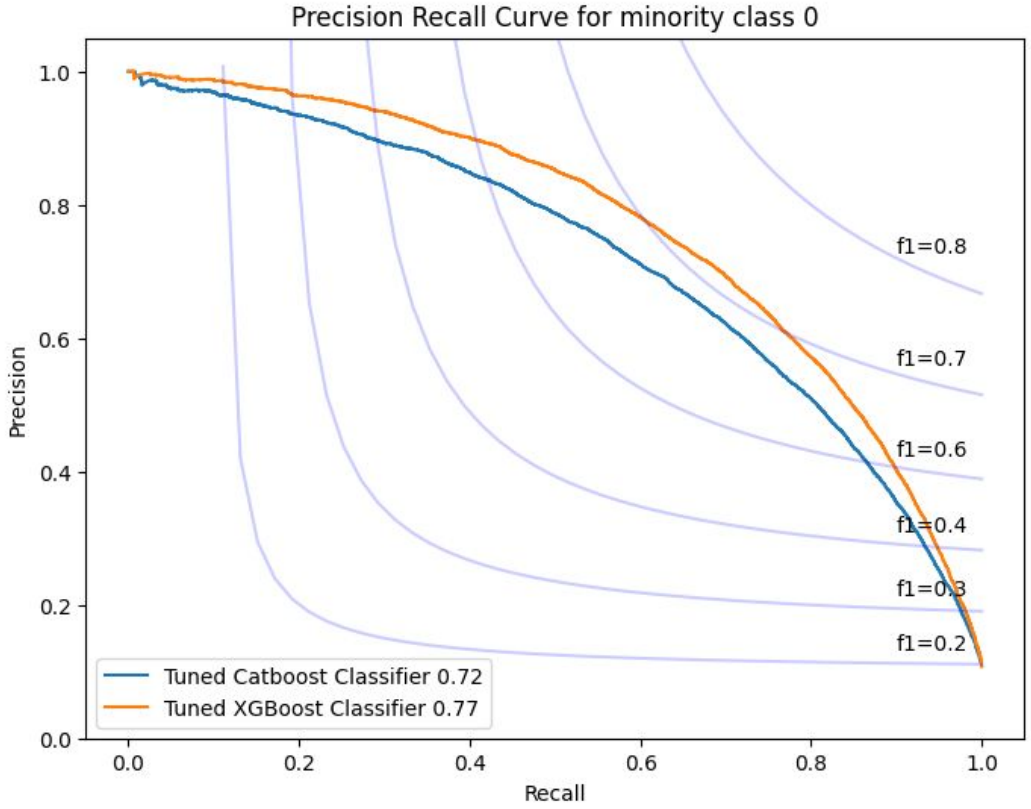
- We optimized hyperparameters of the model using optuna (implementation of the Bayesian optimisation). It uses validation set for evaluation of the model and chooses the best set of parameters maximizing the F1 micro metric.

- We encoded categorical features using Catboost target encoding.

- Catboost was very slow to train and validate, therefore it was difficult to optimize. Because of this we decided to try XGBoost and LightGBM boosting libraries.

Source: https://optuna.org/

XGBoost vs Catboost

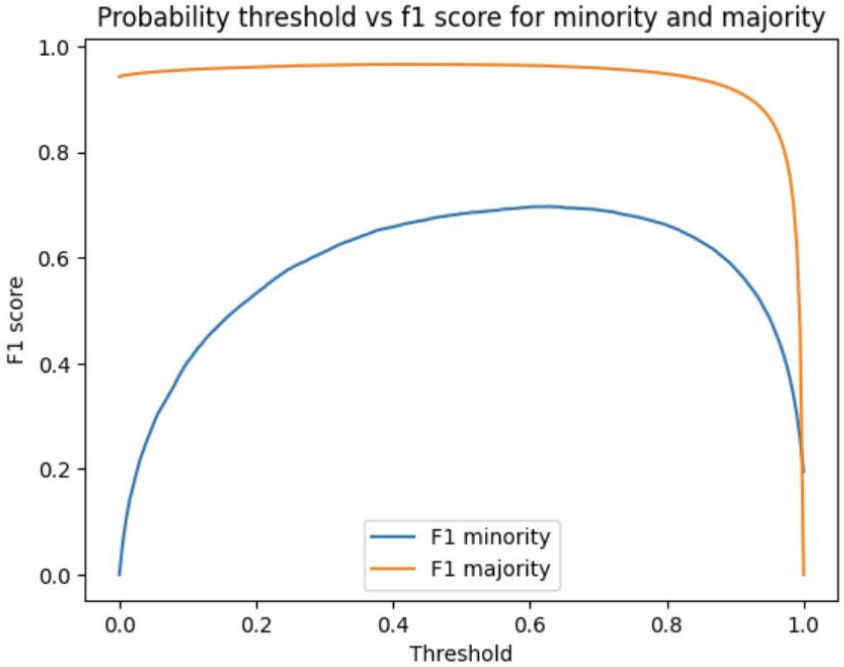


- We trained XGBoost model using the same pipeline. In the same way we optimized its hyperparameters using Optuna and the same validation set. The result are following:
- As a result **XGBoost was faster** to train and **overall better** (maybe due to higher number of boosting rounds we were able to run)
- To verify this we created Precision-Recall curve. The higher area under the graph = better. Overall tuned XGBoost model performs better, this corresponds with higher target f1 micro score

Source:

XGBoost tuning + Threshold optimisation

- After seeing that XGBoost is more preferable in our case. We put more effort and GPU resources into finding optimal hyperparameters.
- On top of that, we optimized for threshold. Threshold is used as cut off point where classifier chooses weather it is class 0 or 1. This way we were able to control how confident the model should be when labeling observations.
- By default, 0.5 threshold is used, meaning that if predicted probability is 0.51, classifier is really not sure if this is 0 or 1



Source:

Training the final model

- To ensure that the model is trained without under or overfitting we went for training with early stopping. This means, that once there is no improvement in loss function (Logloss in our case) on the validation set, the model stops boosting rounds.

- We used 5% of the whole data as a validation set for early stopping. This way we were able to see the rough performance of the model before submitting it.

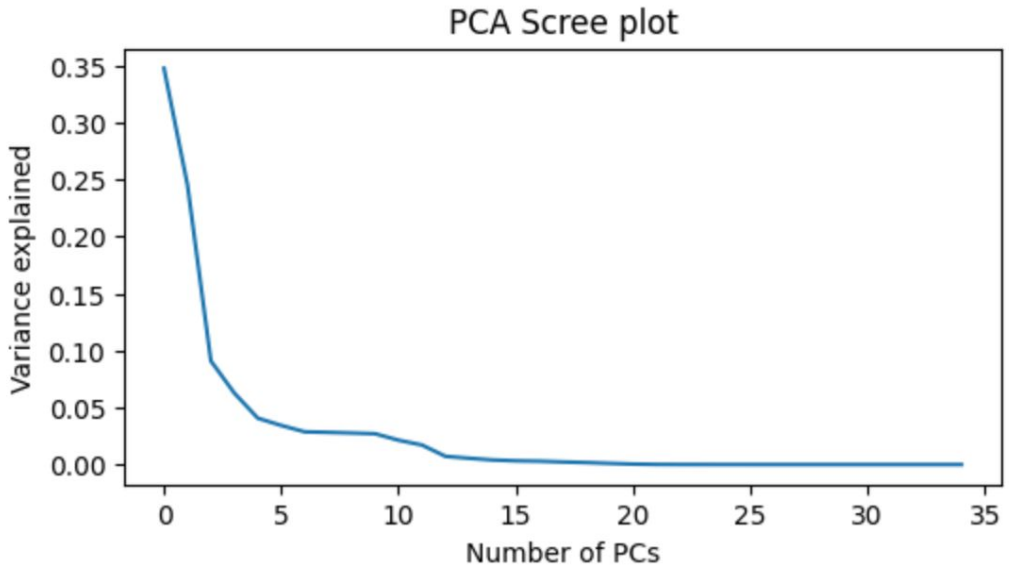
	precision	recall	f1-score	support
0.0	0.81	0.59	0.68	9360
1.0	0.95	0.98	0.97	76000
accuracy			0.94	85360
macro avg	0.88	0.78	0.82	85360
weighted avg	0.93	0.94	0.93	85360

Source:

Appendices

Appendix 1. PCA Scree plot

- First 15 Principal Components describe more than 99.95% of the variance in the data.
- We tried to use PCs as features instead of the initial features (mainly to reduce number of features to be able to train faster)
- But also with this approach we were able to spot two heavy outliers and get overall feel of the data

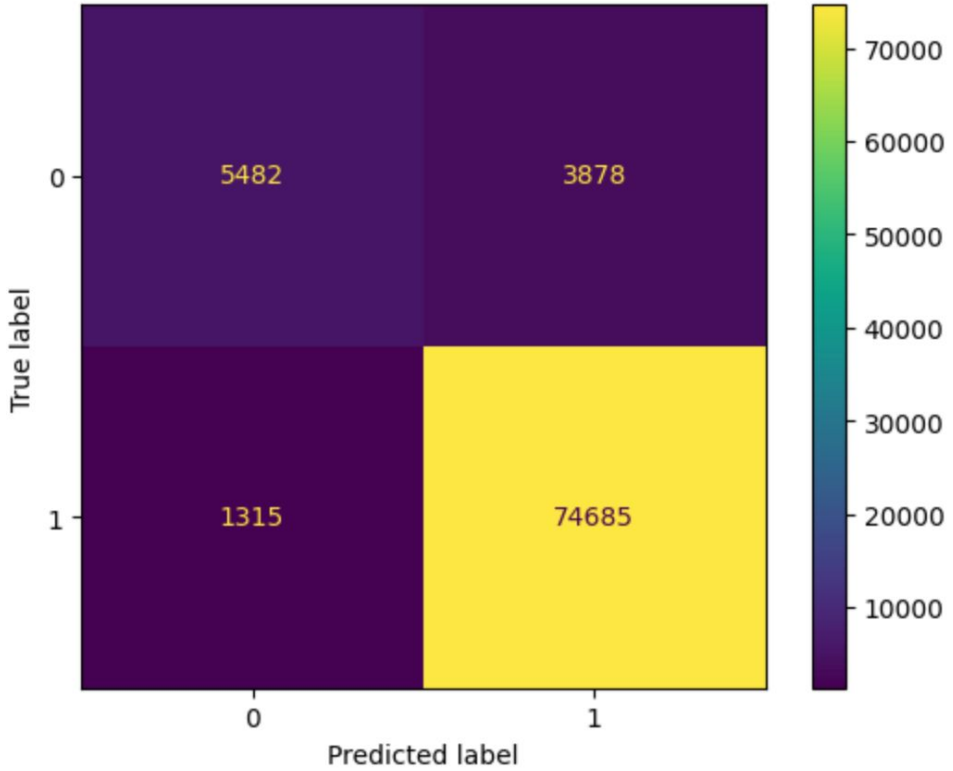


Source:

Appendix 2. Confusion Matrix. Final model

- We can see that the model is well-behaved. It doesn't produce 0 prediction just to label minority classes.

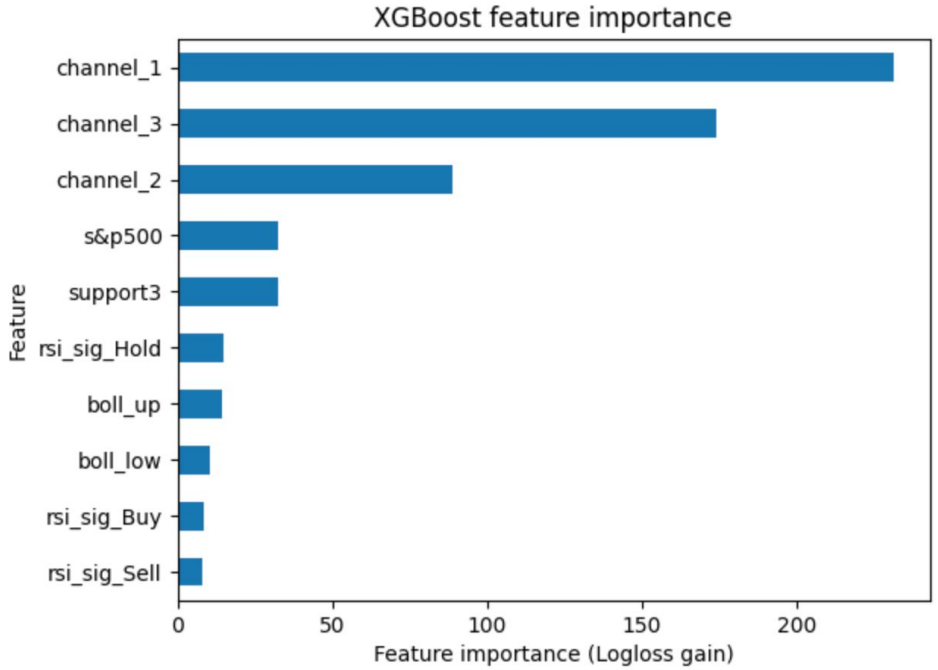
- Configuration of the model whether it is able to detect all observations of the minority depends on the task. Here we were aiming to optimize for an arbitrary f1 micro score. In real life, we might be obliged to detect all of the zeros, therefore this wouldn't be a great result, since we still misclassify quite a few zeros.



Source:

Appendix 3. Feature importance. Informative variables

- As we can see from feature importances (defined as a contribution to the final logloss decrease). We could use very few variables, as others are not informative.
- But still we went for all of the variables since they still contain some variance (it might end up being just random totally unrelated noise) but this still helped a bit with the target f1-score. But if we were to train the real production model, we would have taken only significant variables to make sure we do not pick up any noise.



Source: