MAP569 – Who are the high-frequency traders?

Ali Mammadov
ali.mammadov@ip-paris.fr

Ayham Olleik
ayham.olleik@polytechnique.edu

Khalig Aghakarimov
khalig.aghakarimov@polytechnique.edu

March 31, 2021

Abstract

This small paper report discusses the overall work done and possible solutions provided for the data challenge called "Who are the high-frequency traders?" organized by ENS for the academic and non-academic participants. The report starts with the introduction to the problem, then covers the acquaintance with the data, problems faced and overcome during the challenge, the models tried and finally ends with certain results achieved.

Keywords: Machine Learning, Data Challenge, Data pre-processing, Feature Engineering, Correlation, Model Training

1 Introduction

The financial stock market contains multiple types of traders based on the frequency and amount of trades they do. For example, High Frequency Traders (HFT) are complex algorithms that trade at a very high frequency compared to the normal traders. They also trade huge amount of stocks in almost impossible time for a normal human. HFT is then seen as a controversial topic as in some cases it allows the big companies to make profit at the expense of the small ones and additionally causing the term called ghost liquidity as explained in this article [3]. In this paper, we aim to classify these traders into three main categories: **HFT**, **Non HFT** and **MIX**.

2 Data pre-processing

Giving a general overview and understanding the nature of the data is significant before doing any learning or training procedure since it gives a good insight about what can be done to get a better working and state-of-the-art solution. This section describes several well-known pre-processing techniques applied to our data.

2.1 Scaling

Plotting helps to visually see how our data is distributed. After plotting the histograms for the features, we observed the concentration near 0 for the majority and decided to apply some techniques during the data cleaning process to scale it without any loss of information in the global ranges of values. We applied the following scaling methods:

- Standardization. It puts the whole data column in the range [0,1]:

$$Z = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

- Normalization. Scales the data column in such a way it ends up with the mean of 0 and standard deviation of 1:

$$Z = \frac{X - \mu}{\sigma} \tag{2}$$

where μ stands for the mean and σ stands for the standard deviation in the original case.

- Log-scaling. Scales the data column in a natural logarithmic scale and it is a way of displaying numerical data over a very wide range of values in a compact way — typically the largest numbers in the data are hundreds or even thousands of times larger than the smallest numbers:

$$Z = \log(X + 1) \tag{3}$$

The log-scaling gave the best results among all. One can see the effect of the technique looking at the following histograms:

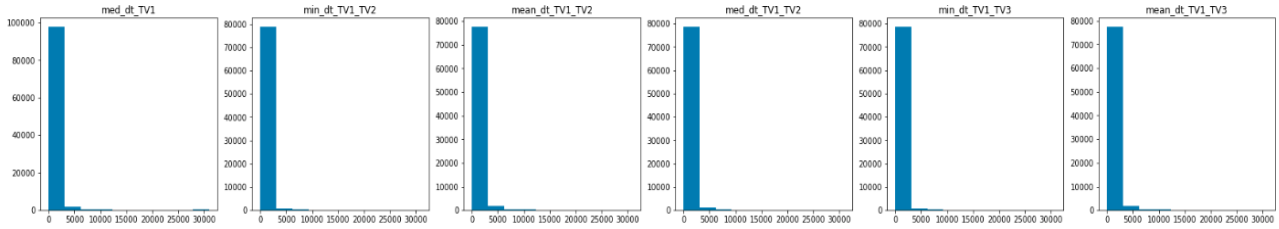

Figure 1: Histograms before scaling

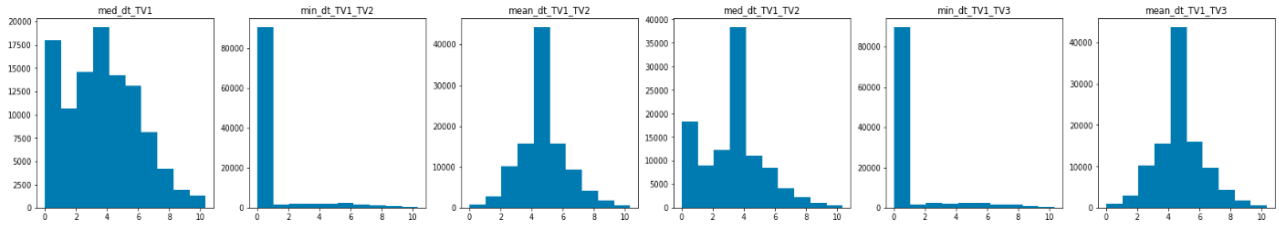

Figure 2: Histograms after scaling

2.2 Feature engineering

The Feature Engineering involves a process that deals with manipulating or dropping the features of the data. In terms of feature selection, we decided to throw the four following ones:

- Trader as it is the target that we want to predict.
- Index since it just represents the row number and has nothing to do with the training.
- Day due to its 0 correlation with the Trader.
- Share since it does not contain any useful information.

Besides, we are performing a one-hot encoding for the 3 types of traders that we want to predict, i.e. *HFT*, *NON HFT* and *MIX*. Since we are representing them as a 3 dimensional vector ([1, 0, 0], [0, 1, 0] and [0, 0, 1]), it will result in a manual expansion for the data columns during the pre-processing. This is exactly where we have referred to feature extraction.

2.3 Correlation Matrix

The correlation matrix is a good tool to see the dependency between the different features of our data and the one that we want to predict. In the figures below, you can see a part of the correlation matrix (see Appendix A for a full version) that we got for all our features in the data. We obtained this after the pre-processing stages that we described in the previous subsections.

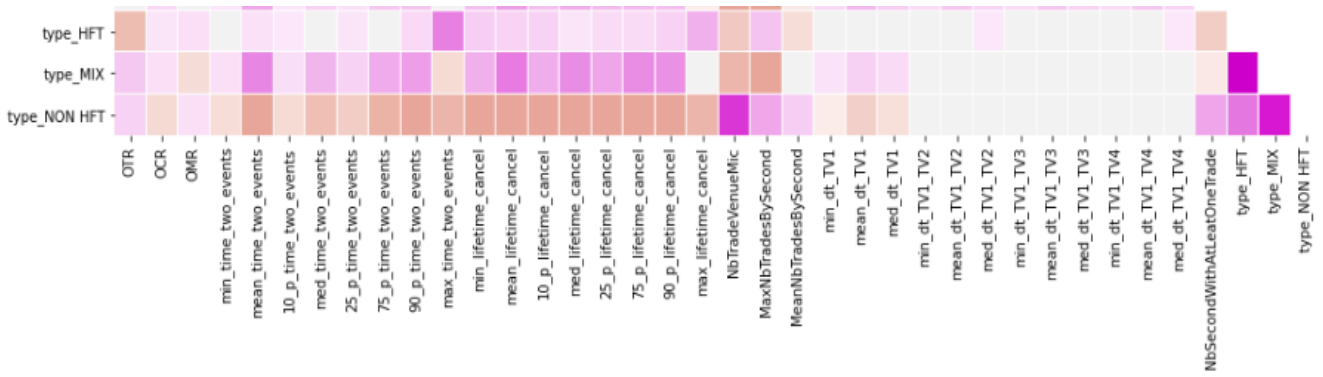

Figure 3: Correlation matrix without scaling

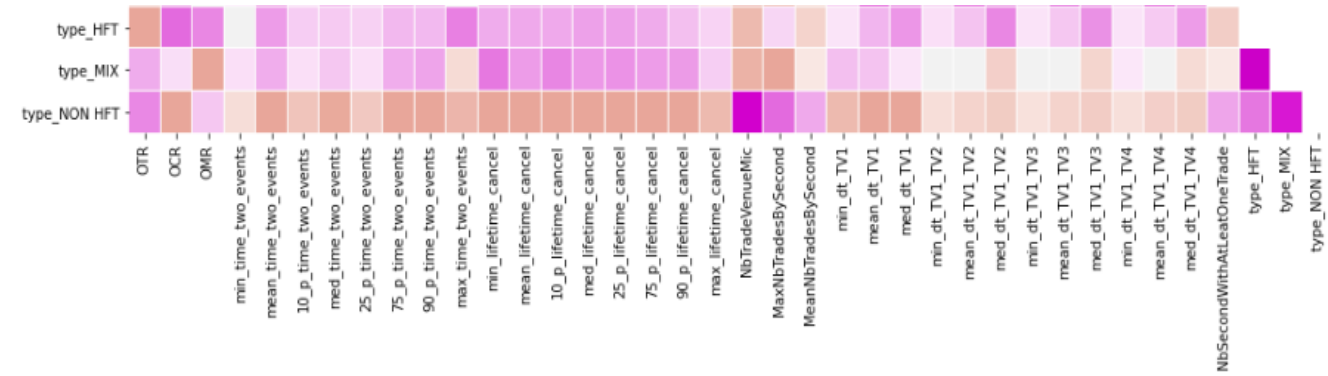

Figure 4: Correlation matrix after scaling

One can see a significant change in the colors between the 2 images that indicates a better correlation in *Figure 4*.

3 Models

3.1 Random forest (RF)

Random forests are a version of ensemble learning where many different decision trees are trained together so that each one of them gives its own prediction for the input. The random forest final prediction will usually be a function (average/mean ...) over these singular predictions to get the final predicted class. Random forests have an advantage of avoiding over-fitting compared to normal decision tree methods.

3.2 Gradient boosting (GB)

Gradient Boosting is also a version of ensemble learning that is very similar to Random forest as it usually uses the concept of building small learners based on decision trees. The main different aspects between Random forests and Gradient boosting are:

- Gradient boosting does not build independent decision trees, they are dependent upon each other and built one at a time.
- Gradient boosting combines the resulting predictions along the way when running the algorithm, unlike Random forest that finds the result as an average of all the single tree predictions at the last end [4].

3.3 Extreme Gradient Boosting (XGB)

XGB is an implementation of the Gradient boosting method which uses more accurate approximations to find the best tree model. It employs a number of nifty tricks that make it exceptionally successful, particularly with structured data. The most important are:

- Computing the second order gradient of the loss function to provide more information about the direction change of the gradient and to give a better approximation of the error.
- It uses advanced regularization (both L1 and L2) which prevents better overfitting [5].

3.4 Catboost (CatB)

Catboost is considered as the updated version of Gradient boosting since it involves some extra advantages in terms of performance. It is very convenient for solving problems related to regression, classification or multi-class classification. The good thing about Catboost is that, it implements a so-called ordered boosting that avoids overfitting using the permutation-driven alternative. This is the reason why Catboost performs better than the classical Gradient boosting. On the other hand, it deals with the categorical features based on its built-in approach, however we are not benefiting from this since our data totally consists of numerical features. [2]

3.5 Deep Learning Model

Another model which we used for this data challenge is the Deep Neural Network (DNN). For this method, we applied Automated Machine Learning (AutoML) approach from AutoKeras library [1]. AutoML is the process of automating the way for finding the number of layers and number of nodes for each layer by using random search (in our case) from given search space. Our search space consists of an input layer, several hidden Dense layers, then a Dropout layer and finally an output Dense layer of 3 nodes corresponding to (HFT, MIX, NON HFT) with **softmax** activation function which is the best for multi-class classification task. From this search space, AutoML will find the number of hidden layers, number of nodes for each of them as well as the dropout rate for creating the best architecture for our problem. Based on our experience, the best architecture is on *Figure 5*. For compiling this model, we used **categorical_crossentropy** loss function and Adam Optimizer. At the end, we applied the early stopping approach so that our model directly stops when the delta change over our error is below a certain tolerance $t = 10^{-4}$ for p consecutive epochs where $p = 5$ is our patience value. This approach prevents overfitting since it does not let the score to increase as a trade-off of regularisation.

4 Computational Results

We present the results of the different models we tried in the *Table 1*. As expected, the RF model did give intermediate results for both the training and the submission. Next, we moved to the GB model that improved our scoring result by almost 24% on the training score, but only 10% on the submission score which shows a bit of lack of generalization. Therefore, we tried a more regularized gradient boosting like XGB and CatBoost. Both of these models gave extremely accurate results on the training scores (almost 100%) and did generalize quite well for the submission score (92.8 % and 95.2% respectively). A final method we decided to try was to run a Deep Learning architecture with the willing of getting a more accurate result if possible. We tried different architectures and the best on is described in **Figure 5**. The highest submission score reached for these DL trials was around 90%.

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_78 (Dense)             (None, 80)                2880
_________________________________________________________________
dense_79 (Dense)             (None, 128)               10368
_________________________________________________________________
dense_80 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_81 (Dense)             (None, 48)                3120
_________________________________________________________________
dense_82 (Dense)             (None, 32)                1568
_________________________________________________________________
dense_83 (Dense)             (None, 16)                528
_________________________________________________________________
dense_84 (Dense)             (None, 3)                 51
_________________________________________________________________
activation_13 (Activation)   (None, 3)                 0
=================================================================
Total params: 26,771
Trainable params: 26,771
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: The best Deep Learning architecture

Model	Training Time(s)	Training Score	Submission Score
RF	10.2	0.717	0.761
GB	114	0.946	0.857
XGB	29.6	0.996	0.928
CatB	33.8	0.989	0.952
DL	144	0.905	0.904

Table 1: Model results

5 Conclusion

In this paper, we studied the problem of classifying the traders in the market into 3 main categories: HFT, NON HFT and MIX. The data provided about the traders consists of a bunch of numerical columns that give statistics (mean, media, percentile ...) of the frequency and number of trades that the trader is performing. The data also contains the amount of modifications and cancellations made by each trader. After exploring multiple classification models, we ended up with the highest result from the CatBoost classifier with a result of 95.2% on the submission score *Table 1*.

6 Code

Our code is publicly available in the following GitHub link.

References

[1] *AutoKeras*. URL: https://autokeras.com/.

[2] *Catboost*. URL: https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14.

[3] *HFT*. URL: https://www.investopedia.com/terms/h/high-frequency-trading.asp.

[4] *Random forest Vs Gradient Boosting*. URL: https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained#:~:text=Like%5C%20random%5C%20forests%5C%2C%5C%20gradient%5C%20boosting,one%5C%20tree%5C%20at%5C%20a%5C%20time..

[5] *XGBboost*. URL: https://www.shirin-glander.de/2018/11/ml_basics_gbm/#:~:text=While%20regular%20gradient%20boosting%20uses,)%2C%20which%20improves%20model%20generalization..

7 Appendices

Appendix A.1: Correlation matrix before pre-processing

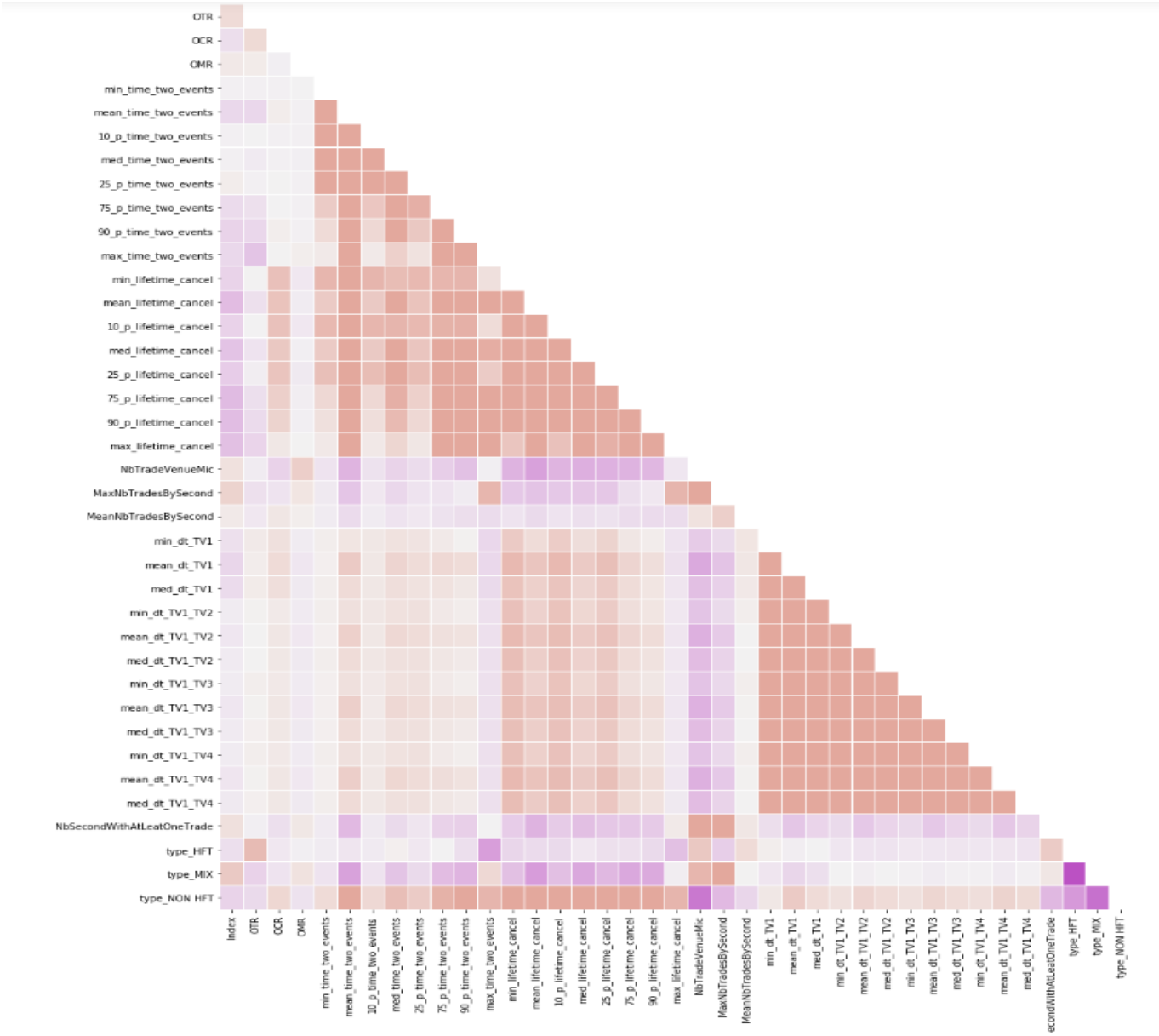

Figure 6: Correlation matrix before

Appendix A.2: Correlation matrix after pre-processing

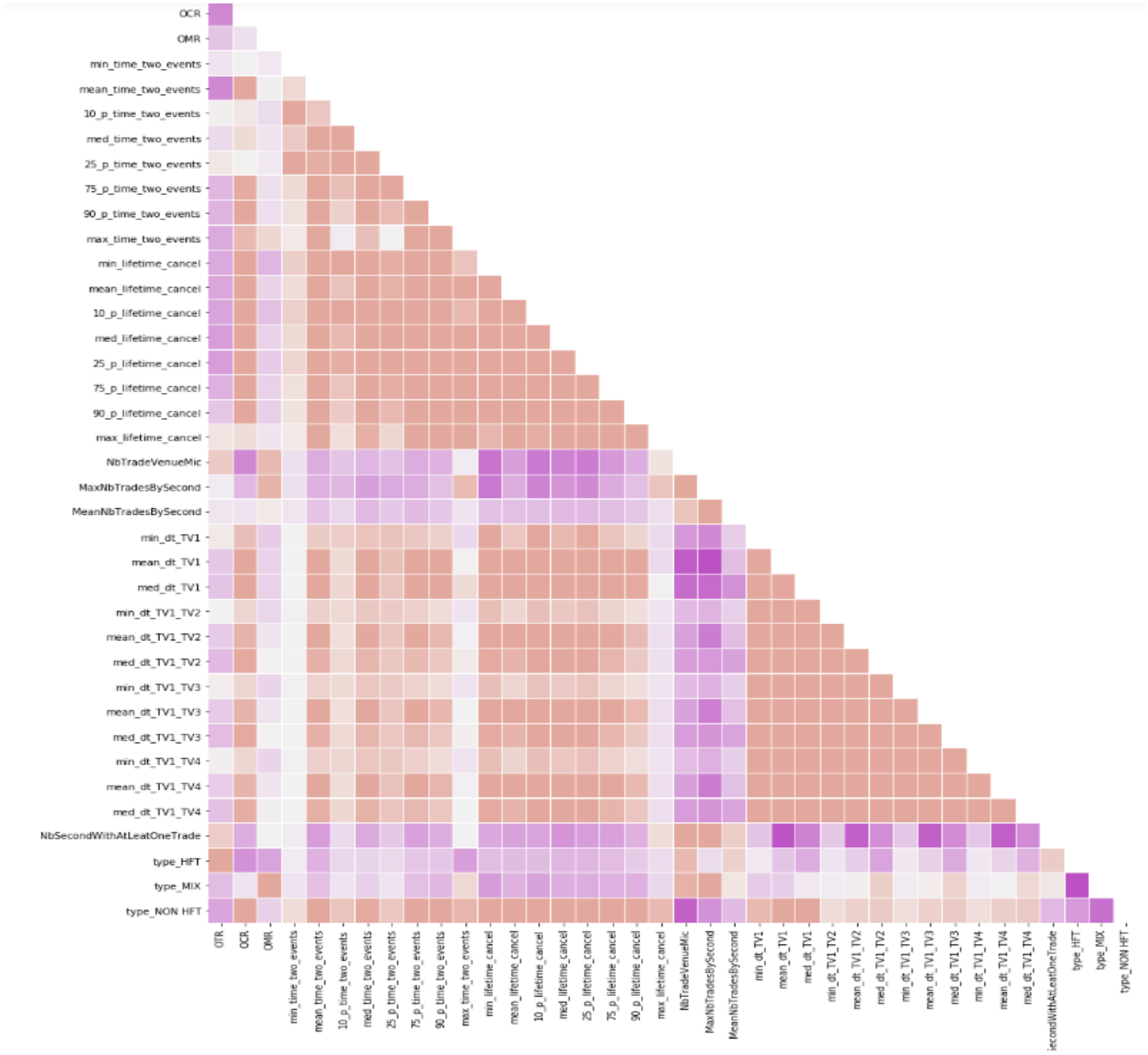

Figure 7: Correlation matrix after

Appendix B.1: Histograms before pre-processing

Appendix B.2: Histograms after pre-processing