- Machine Learning II -
Who are the high-frequency traders ?
AMF - Final Report

Hugo Ehlinger, Nicolas Maisonneuve-Bonteil, Maxime Redstone Leclerc

March 2021

Introduction

The Autorité des Marchés Financiers (AMF) seeks to better understand the different types of players that make up the financial markets. Each player has its own objectives and strategies. The aim of this project is to design an algorithm able to predict the type of player using data from trades executed on the market.

Three categories of traders have been identified by the AMF: High Frequency Traders (*HFT*), *Non-HFT* and *MIX* (traders executing algorithmic and normal trades).

This report outlines our approach to this problem using Machine Learning and the obtained results. Section 1 describes the data provided by the AMF. Section 2 outlines step by step the pipeline we set up to predict the type of traders, from data preprocessing to the algorithm we used. Section 3 evaluates the performance of our approach while Section 4 describes how to reproduce our results.

This project is supported by a public Github repository that contains the entire code we will refer to in this report. The repository can be found at `https://github.com/hehlinge42/AMF_challenge`.

1 Exploratory Data Analysis

The AMF provided several features regarding the trades executed by different traders. These include the number of trades executed, cancelled or modified, the number of venues the trader traded on and the time delta between different trades in different trading venues. An exhaustive description of features can be found in Appendix A.

Figure 1 shows the overall correlation between all the features while Figure 2 only outputs the correlations greater than 90% and highlights a couple of redundant features.

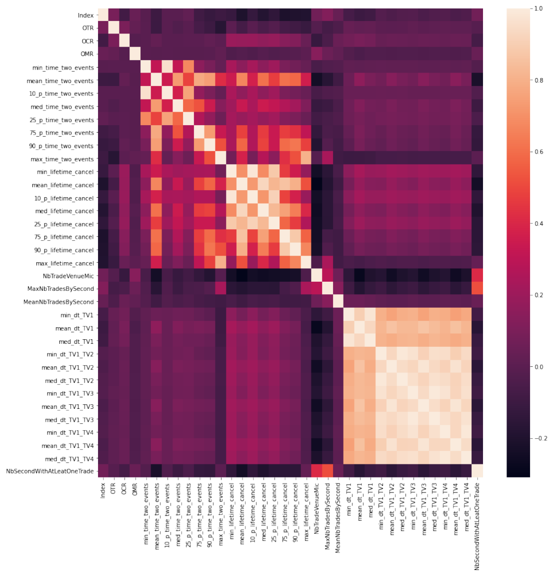

Figure 1: HeatMap of all trading features.

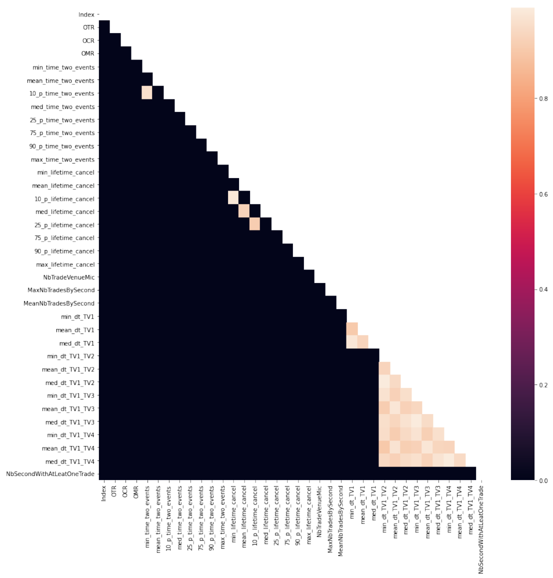

Figure 2: HeatMap of trading features with above 90% correlation.

2 Pipeline

This section describes step by step the pipeline set up to reach the objective. It contains several steps of pre-processing and modelling.

2.1 Preprocessing

2.1.1 Dealing with missing values

The training dataset contained several features with missing values, which needed to be handled before training a machine learning model. These features are *OTR*, *OMR*, *min_dt_TV1*, *mean_dt_TV1*, *med_dt_TV1*, *min_dt_TV1_TV2*, *mean_dt_TV1_TV2*, *med_dt_TV1_TV2*, *min_dt_TV1_TV3*, *mean_dt_TV1_TV3*, *med_dt_TV1_TV3*, *min_dt_TV1_TV4*, *mean_dt_TV1_TV4* and *med_dt_TV1_TV4*.

The *OTR* and *OMR* features are defined as ratios between the number of all types of events sent to the Limit Order Book (LOB) and the number of trades and the number of modification-type event respectively. Therefore, NaNs can occur when there is no trades or modification-type events as 0 would be at the denominator of the ratio. We inverted this ratio, so that the case when there is no trade or modification-type event resulted in 0 being in the numerator, which in turn outputted 0 instead of NaN. We also normalized these ratios to make them add up to 1. This ensured that the scale of these features remained constant over the records. Below is the pseudo-code for these transformations.

Algorithm 1: Normalizing NaNs

```
   Input: dataset
1  for record in dataset do
2      for feature in ['OTR', 'OCR', 'OMR'] do
3          if record[feature] == NaN then
4              record[feature] = 0
           end
5          else
6              record[feature] = 1 / record[feature]
           end
       end
7      record['total'] = record['OTR'] + record['OCR'] + record['OMR']
8      for feature in ['OTR', 'OCR', 'OMR'] do
9          record[feature] = record[feature] / record['total']
       end
   end
   Output: transformed dataset
```

4% of the records had missing values in the fields *min_dt_TV1*, *mean_dt_TV1* and *med_dt_TV1*. These missing values meant that only one trade occurred for this particular trader and share at this date on a single trading venue. As the time delta between trades is the main distinctive feature between High Frequency Traders and Non High Frequency Traders, these records were deleted as they are unlikely to bring relevant information to the algorithm.

24% of the records have missing values in the fields *min_dt_TV1_TV2*, *mean_dt_TV1_TV2*, *med_dt_TV1_TV2*, *min_dt_TV1_TV3*, *mean_dt_TV1_TV3*, *med_dt_TV1_TV3*, *min_dt_TV1_TV4*, *mean_dt_TV1_TV4* and *med_dt_TV1_TV4*. These missing values meant that only one trade occurred for this particular trader and share at this date on different trading venues. The share of missing values is high and these features are very correlated with *min_dt_TV1*, *mean_dt_TV1* and *med_dt_TV1* as shown in section 1. Therefore, these features were deleted. This fulfills the objective of the AMF to reduce the volume and sources of data required to detect High Frequency Traders.

2.1.2 Feature Engineering

Some feature engineering was performed in order to create additional meaningful features from the existing data.

- The number of daily trade was computed by multiplying the features *NbSecondWithAtLeatOneTrade* and *MeanNbTradesBySecond*.
- The gap between some features was also computed. For instance, the ratio between the features *max_time_two_events* and *min_time_two_events*. This resulted from the fact that the conditional means of the three classes of traders were significantly different as shown in appendix B. The *MIX* traders have a wider gap than the other two classes as they behave both as a robot, having very small values for *min_time_two_events* and as a human, having large values for *max_time_two_events*. Therefore, this feature is expected to ease the identification of *MIX* traders. The entire list of features engineered following this principle is detailed in appendix C

2.2 Algorithm

2.2.1 Random Forest Classifier

The baseline algorithm chosen by the AMF for its benchmark is a Random Forest Classifier. We used the XGBoost implementation (wrapped in Scikit-Learn) to improve the benchmark performance. The choice of a Random Forest Classifier results from the following points:

- The number of features provided by the AMF is too small to design an efficient Deep Neural Network.
- Random Forests have the ability to draw complex non-linear decision boundaries.
- Random Forests prevent the tendency of decision trees to overfit by implementing several features described below.
- Random Forests offer the possibility to predict a probability for each class, representing the disagreements of the predictions between the estimators. Predicting the probabilities of each class was very useful when aggregating the records by traders as developed in the section 2.2.3.

After optimizing the hyper-parameters of the model using a Grid Search approach, we selected the following parameters:

- Loss function: The 'multi:softprob' objective is used for multi-label classification tasks and outputs a matrix of shape *n_records* * *n_labels*, containing the predicted probability of each data point belonging to each class. This objective is associated to the cross-entropy loss.
- n_estimators and learning rate: The n_estimators parameter determines the number of decision trees to aggregate in the random forest. The learning rate or shrinkage parameter limits the decrease of the loss function of each extra-tree, in order to prevent overfitting. We chose to set 500 estimators to balance accuracy (which usually increases with more trees) and time-performance (which decreases with more trees). We set the learning rate to 0.05, a relatively small value that compensates the relative high number of trees.
- colsample_bytree: This parameter determines the share of features to be used by each tree. We set this parameter to a quite low level of 0.3. This results from the facts that certain features are highly correlated as described in section 1 and that we used a relative high number of estimators.
- max_depth. This parameter was set to 10 and refers to a maximum depth for each decision tree, that is to say the number of successive decision nodes before reaching a leaf.
- min_child_weight. This parameter sets a minimum number of records to constitute a child node. Below this threshold, the records are grouped in a leaf. Setting the parameter above 1 aims at preventing overfitting. We set it to 3.
- subsample: This parameter provides the share of training data to be used by each tree. This parameter was set to 0.9. As the training set is unbalanced (50% of *MIX*, 30% of *HFT* and 20% of *non-HFT*), a high subsample was chosen to prevent a minority class from vanishing from certain trees.
- booster: The booster was set to 'gbtree' in order to support non-linear relationships.

2.2.2 Pseudo Labelling

Pseudo-labelling is a machine learning technique which consists in enriching the training dataset by the entries from the testing dataset for which the algorithm has a high confidence level in it own predictions. We used pseudo-labelling in a loop as follows:

Algorithm 2: Pseudo-labelling

```
  Input: training set, testing set, threshold=0.9, min_number_observations=10
1 while True do
2     Fit the model on the training set
3     Predict the probability of each class for the testing set
4     Aggregate the testing set per trader taking the mean of the probabilities for
        each class
5     if the dominant class has a probability >threshold for at least one trader
        AND the number of entries of this trader >min_number_observations
        then
6         Add these records to training set with dominant class as label
7         Remove these records from the testing set
      end
8     else
9         break
      end
  end
  Output: Prediction of the original testing set using the last trained model
```

The use of pseudo-labelling allows to train the final model with a much larger dataset, which normally leads to a higher accuracy. Two hyper-parameters must be defined by the user:

- threshold $\in [0, 1]$: it is the minimum probability that a class must have to be considered as safe to be incorporated in the training dataset. This threshold must be high enough to prevent the model from misclassifying data that will be incorporated into the training set, which could lead to poor performances.
- min_number_observations: it is the minimum number of records that a trader must have to be incorporated in the training dataset. Indeed, traders which have a very small number of records have more chance to be misclassified and therefore should not be pseudo-labelled.

2.2.3 Softmax Regression

At the exit of the pseudo-labelling loop, the algorithm makes the final prediction for each record of the testing set. Each record, identified by its primary key (trader, share, date), is associated to a probability vector for each of the three classes (*HFT*, *MIX*, *NON HFT*). The final step of the algorithm consists in aggregating the records by trader to label each trader with a unique class.

The benchmark algorithm recommends to use user-defined thresholds and fixed rules to do this final attribution. Instead, we decided to fit another machine learning algorithm (Algorithm 3) to do this classification in order to avoid hard-coded rules.

Algorithm 3: Softmax regression

```
  Input: training set, testing set, fitted random forest classifier
1 y_train_pred = predictions of the training set using the fitted random forest
    classifier
2 y_test_pred = predictions of the testing set using the fitted random forest
    classifier
3 X_train = aggregation of the predictions of y_train_pred, taking the mean of the
    predicted probabilities for each class
4 X_test = aggregation of the predictions of y_test_pred, taking the mean of the
    predicted probabilities for each class
5 y_train = true label of each trader
6 Fit a softmax regression using X_train and y_train
7 final_predictions = predictions of X_test using the fitted softmax regressor
  Output: final_predictions to be submitted
```

This algorithm highlights the issue of overfitting. Indeed, the training set is predicted back using the random forest classifier in order to create the training set for the final softmax regression. Therefore, it is important that the training and testing sets achieve similar performances on the random forest classifier, otherwise the softmax would perform poorly.

2.2.4 Hard-coded rule

The softmax regression described in section 2.2.3 aimed at removing the hard-coded rules suggested by the AMF. Nevertheless, we kept one hard-coded rule to determine our predictions. In the case where the AMF collected very few data on the activity of a trader, this trader is labelled as *Non-HFT* as the AMF lacks evidence to convict the trader as *HFT* or *MIX*. We set the maximum number of records to be automatically labelled as *Non-HFT* to 2.

3 Results

The submission currently graded on both the public and the private dataset reaches a mean f1-score of 0.9762 and 0.9535 respectively. Marginal improvements have been made since the ranking of the private dataset.

On average, the algorithm attributes an 88% probability to the class finally labelled, highlighting that usually, the class of a trader appears obvious. The 1^{st} quantile reaches 86%. Only one trader is classified with less than 50% of confidence.

4 Setup

This project is supported on GitHub and accessible at the following URL: `https://github.com/hehlinge42/AMF_challenge`.

The Pipeline described in Section 2 can be executed to output the submission that led to the results provided in Section 3 using the following command:

```
python3 src/amf_classifier.py --directory [path]
                              --submission [submission_name]
                              --loop [nloops]
```

Source Code 1: Running our implementation via command line.

The program accepts the following options:

- -d or –directory [directory]: path to submission directory.
- -s or –submission [submission_name]: base name of submission. The program will output the file directory/submission_name.csv, formatted to be submitted, and the file directory/submission_name_full.csv which outputs the probability of each class for each trader.
- -l or –loop [nloops]: the maximum number of iterations of the pseudo-labeller. If set to -1, the model will be retrained until the pseudo-labeller finds no trader in the testing set with a sufficient confidence level.

Conclusion

The pipeline we designed has managed to distinguish very well the *HFT*, *MIX* and *non-HFT* traders, reaching a mean f1-score of 0.9649. This highlights the fact that the typical behaviour of these three types of traders are still very distinct. Nevertheless, we can expect *HFT* developers to adapt to the use of machine learning tools to detect them. It is very likely that machine learning will also be used to learn the *non-HFT* behaviour and make robot-traders behave like humans not to be detected. This will challenge the AMF algorithms, which will need to detect more subtle differences of behaviours in order to detect non-human traders.

Appendix

A Data Description

X_train and X_test data exhibit in their rows the same 35 features calculated for a given market player i on a certain stock and a specific trading date:

1. *NbTradeVenueMic*: number of trading venues on which the market player trades
2. From all trading venues, statistics over the number of trades observed per second:
   - mean: *MeanNbTradesBySecond*
   - max: *MaxNbTradesBySecond*
3. Statistics over the observed time delta between two trades on the trading venue TV_1:
   - min: *min_dt_TV1*
   - mean: *mean_dt_TV1*
   - median: *med_dt_TV1*
4. Statistics over the observed time delta between two trades occurring on trading venue TV_1 and then on trading venue TV_2:
   - min: *min_dt_TV1_TV2*
   - mean: *mean_dt_TV1_TV2*
   - median: *med_dt_TV1_TV2*
5. Statistics over the observed time delta between two trades occurring on trading venue TV_1 and then on trading venue TV_3:
   - min: *min_dt_TV1_TV3*
   - mean: *mean_dt_TV1_TV3*
   - median: *med_dt_TV1_TV3*
6. Statistics over the observed time delta between two trades occurring on trading venue TV_1 and then on trading venue TV_4:
   - min: *min_dt_TV1_TV4*

    - mean: *mean_dt_TV1_TV4*
    - median: *med_dt_TV1_TV4*

7. *NbSecondWithAtLeatOneTrade*: from all trading venues, number of seconds during the trading day where at least one trade of the market player i is observed
8. On trading venue TV_1, three ratios between the number of all types of events sent to the LOB and:
    - the number of trades (*OTR*)
    - the number of cancellation-type event (*OCR*)
    - the number of modification-type event (*OMR*)
9. On trading venue TV_1, statistics over the observed time delta between two all-type events sent:
    - min: *min_time_two_events*
    - mean: *mean_time_two_events*
    - 10th percentile: *10_p_time_two_events*
    - 1st quartile: *25_p_time_two_events*
    - median: *med_time_two_events*
    - 3rd quartile: *75_p_time_two_events*
    - 90th percentile: *90_p_time_two_events*
    - max: *max_time_two_events*
10. On trading venue TV_1, statistics over the observed lifetime of cancelled orders:
    - min: *min_lifetime_cancel*
    - mean: *mean_lifetime_cancel*
    - 10th percentile: *10_p_lifetime_cancel*
    - : 1st quartile: *25_p_lifetime_cancel*
    - median: *med_lifetime_cancel*
    - 3rd quartile: *75_p_lifetime_cancel*
    - 90th percentile: *90_p_lifetime_cancel*
    - max: *max_lifetime_cancel*

The features above are not detailed in the same order as in the challenge files.

B Conditional means across features and labels

Trader Name	HFT	MIX	Non HFT
OTR	18%	40%	66%
OCR	72%	40%	20%
OMR	10%	20%	14%
min_time_two_events	26,42	4,52	365,83
mean_time_two_events	2405,04	1186,68	6950,96
10_p_time_two_events	31,75	6,21	429,29
med_time_two_events	1093,53	100,45	2993,54
25_p_time_two_events	102,06	13,60	905,06
75_p_time_two_events	4676,17	2519,53	9922,36
90_p_time_two_events	8209,92	6289,91	20092,89
max_time_two_events	15587,23	26104,40	32561,41
min_lifetime_cancel	879,22	620,71	6519,44
mean_lifetime_cancel	4390,83	2499,79	14548,43
10_p_lifetime_cancel	1085,51	695,80	6644,12
med_lifetime_cancel	3887,95	1670,31	11633,64
25_p_lifetime_cancel	1910,57	974,55	7836,81
75_p_lifetime_cancel	7073,92	4488,17	19548,93
90_p_lifetime_cancel	9307,23	7096,86	25194,09
max_lifetime_cancel	14832,86	19649,81	28100,25
Nb Trade Venue Mic	6,67	6,62	1,55
Max Nb Trades By Second	17,91	36,21	10,49
Mean Nb Trades By Second	2,29	2,09	1,81
min_dt_TV1	517,23	301,56	666,31
mean_dt_TV1	992,58	730,80	2112,29
med_dt_TV1	641,42	442,48	1199,01
Nb Second With At Leat One Trade	480,75	364,77	53,50

C Feature Engineering

- Nb_trade = NbSecondWithAtLeatOneTrade * MeanNbTradesBySecond
- min_vs_max_two_events = max_time_two_events / min_time_two_events
- 10_vs_90_two_events = 90_p_time_two_events / 10_p_time_two_events
- 25_vs_75_two_events = 75_p_time_two_events / 25_p_time_two_events
- min_vs_max_cancel = max_lifetime_cancel / min_lifetime_cancel
- 10_vs_90_two_cancel = 90_p_lifetime_cancel / 10_p_lifetime_cancel
- 25_vs_75_two_cancel = 75_p_lifetime_cancel / 25_p_lifetime_cancel