ENS Data Challenge 2022: Return Forecasting of Cryptocurrency Clusters

Victor Hoffmann

March 2023

Abstract

This challenge tackled the issue of the predication of the mean return of cluster's assets relatively to the bitcoin during the last hour of the day, given the last 23 hours. My approach was to build a stacked regressor containing three base models: XGBoost Regressor, Random Forest Regressor and Light GBM Regressor. Some preprocessing steps had to be made in order to get the best results, including missing value imputation and feature selection. My global rank is currently 11th out of the 51 participants.

1 Introducing the problem

During the year 2022, the ENS Ulm and the Collège de France organized data science challenges, which allowed several companies to publish a challenge on their website. *Napoleon X* published a cryptocurrency challenge, where the goal of the challenge was to predict the returns vs. bitcoin of clusters of cryptoassets. *Napoleon X* is interested in detecting which assets are likely to move together in the same direction, i.e. assets whose returns (absolute price changes) are statistically positively correlated. Such assets are regrouped into "clusters", which can be seen as the crypto equivalent of equity sectors or industries. The knowledge of such clusters can then be used to optimize portfolios, build advanced trading strategies (long/short absolute, market neutral), evaluate the systematic risk, etc. In order to build new trading strategies, it can be helpful to know whether a given sector/cluster will outperform the market represented by the bitcoin. For this reason, given a cluster $C = \{A_1, \ldots, A_n\}$ composed of n assets A_i, this challenge aims at predicting the return relatively to bitcoin of an equally weighted portfolio composed of $\{A_1, \ldots, A_n\}$ in the next hour, given series of returns for the last 23 hours for assets in the cluster.

Therefore, this problem is a regression problem. The measure for this challenge was the root mean squared error (RMSE).

2 My approach

Since this is a regression problem, my approach was to build a stacked regressor containing three base models: an XGBoost Regressor, a Light GBM Regressor and a Random Forest Regressor. The meta regressor is a linear regression. Those three base models have proven to give the good results and to train quite fast compared to the amount of data and stacking them alltogether helps to improve performance. All of them had their hyperparameters optimized thanks to a randomized search.

3 Description of the data

Input datasets comprised 29 columns. Each line is indexed by a unique ID, which corresponds to a cluster (defined by "cluster"), a cluster sample day (defined by "day"), and a cryptocurrency (defined by "asset"). $ret_1, \ldots, ret_{23}$ correspond to the first 23 hours of the day returns relatively to bitcoin, arranged in chronological order. md and bc are two secret quantities. The input data contains a certain number of `NaN` values.

Output datasets comprise only 2 columns: `sample_id` a unique sample identifier for a given $\{cluster, day\}$ pair, and `target`, a float, the mean return of cluster's assets relatively to the bitcoin during the last hour of the day.

4 Preprocessing

As one can notice, the number of output data and input data is not the same, since the output only matches the target for a certain $\{cluster, day\}$ pair, whereas the input also depends on cryptocurrencies ("assets"). Therefore, the first job was to merge the input with the output by following the rule: $\texttt{sample\_id} = \texttt{cluster} \times 21 + \texttt{day}$.

Since the data is only numerical, no categorical encoding had to be made, which made the preprocessing easier.

Nonetheless, one of the main problem to deal with was the missing values. I decided to opt for a simple approach, which was to replace each missing value by the median value of the variable. I have also made a K-nearest-neighbours imputation, however, the results happened to be worse than the median imputation.

Since the models I used (XGBoost, Random Forest and Light GBM) happen to be decision trees, no normalization of the data had to be made.

Before training the final model, I used an XGBoost Regressor models whose hyperparameters were optimized in order to do a feature selection. The variables I decided to remove were the following ones: `id, md, asset, cluster, day`.

5 The model

For this problem, I decided to use a stacked regressor model, which had three base models: XGBoost, Light GBM and Random Forest Regressors. All of these base models had their hyperparameters optimized with randomized search. The following hyperparameters for each base models can be found on the 8th cell of the jupyter notebook file.

A stacked regressor can outperform single models when the base models that are stacked have independant results from each other, which happened to be the case here. Thanks to the `seaborn` library, I was able to plot a pairplot of the three models predictions and was able to check that there wasn't any sort of dependance.

The final model of the stacked regressor is a linear regression, since the final model always has to be a simple model when stacking in order to avoid overfitting.

6 Results

On the test set, this model has achieved a root mean squared error of $0,0062$, achieving the 11th place out of the 51 participants.

7 Ways to improve the model

We might think of a couple ideas to improve the model.

Indeed, I believe that median imputation doesn't seem like the best way to handle missing values. Even though K-nearest-neighbours-impute happened to give weaker results, a better imputation might be found.

Moreover, other models might be used in order to solve this problem. Some models that would be adapted to these kind of problems would be Recurrent Neural Networks, Transformers, or any model that deals with sequences. However my results using such models were lower than the ones obtained by this stacked regressor.