Application of the XGBoost algorithm and Bayesian optimization for the Bitcoin price prediction during the COVID-19 period

Jakub Drahokoupil

FFA Working Paper 6/2022

FACULTY OF FINANCE AND ACCOUNTING

Bibliographic information:

Drahokoupil, J. (2022). *Application of the XGBoost algorithm and Bayesian optimization for the Bitcoin price prediction during the COVID-19 period.* FFA Working Paper 6/2022, FFA, Prague University of Economics and Business, Prague.

This paper can be downloaded at: wp.ffu.vse.cz

Contact e-mail: ffawp@vse.cz

Winston Churchill Sq. 1938/4, CZ-13067 Prague 3, Czech Republic, ffu.vse.cz

Application of the XGBoost algorithm and Bayesian optimization for the Bitcoin price prediction during the COVID-19 period

Author

Jakub Drahokoupil[1]

Abstract

Aim of this paper is to use Machine Learning algorithm called XGBoost developed by Tianqi Chen and Carlos Guestrin in 2016 to predict future development of the Bitcoin (BTC) price and build an algorithmic trading strategy based on the predictions from the model. For the final algorithmic strategy, six XGBoost models are estimated in total, estimating following n-th day BTC Close predictions: 1,2,5,10,20,30. Bayesian optimization techniques are used twice during the development of the trading strategy. First, when appropriate hyperparameters of the XGBoost model are selected. Second, for the optimization of each model prediction weight, in order to obtain the most profitable trading strategy. The paper shows, that even though the XGBoost model has several limitations, it can fairly accurately predict future development of the BTC price, even for further predictions. The paper aims specifically for the potential of algorithmic trading during the COVID-19 period, where BTC cryptocurrency suffered extremely volatile period, reaching its new all-time highest prices as well as 50% losses during few consecutive months. The applied trading strategy shows promising results, as it beats the B&H strategy both from the perspective of total profit, Sharpe ratio or Sortino ratio.

AMS/JEL classification: C11, C39, C61, G11

Keywords: XGBoost; Bayesian Optimization; Bitcoin; Algorithmic trading

1. Introduction

In the recent years, algorithmic trading experiences upsurge of Machine Learning (ML) algorithms usage, such as neural nets, decision tree methods, support vector machines methods and others, where the mentioned methods have multiple subcategories and modifications, each of them used for a specific purpose. An unquestionable advantage of these algorithms is fact, that they can be used for both regression and classification tasks. Furthermore, the advancement of the cryptocurrency market and rise in the crypto trading volume shows increasing attractivity of these assets, especially for young retail investors, rather than for hedge funds and similar types of institutional investors as shown by Giudici, Milne, & Vinogradov, 2019. The just elapsed period of uncertainty caused by the worldwide COVID-19 pandemic caused havoc within all the financial markets, including cryptocurrency markets. All the above-mentioned facts suggest that it could be an

[1] Jakub Drahokoupil; Department of Banking and Insurance, Faculty of Finance and Accounting, Prague University of Economics and Business, Prague, Czech Republic; Email: draj05@vse.cz.

This paper has been prepared under financial support of a grant GAČR 22-19617S "Modeling the structure and dynamics of energy, commodity and alternative asset prices", which the authors gratefully acknowledge.

ideal time for the deployment of the ML based trading strategy of the selected cryptocurrency. In this paper, the selected cryptocurrency would be Bitcoin (BTC), more specifically the timeseries of USD-BTC exchange rate, as the widely known and mostly traded cryptocurrency.

Along with the rise of the ML methods, triggered by the increase of computational capacity of common computers, Bayesian based methods have received their recognition as well. Used in a wide range of various fields (biology, math, physics etc.) as shown by Chen, Huang, Ibrahim, & Kim, 2008, Trassinelli, 2020 or Zhang, Johnson, Little, & Cao, 2010. Of course, finance could not stand aside (Chandra & He, 2021). Markov Chain Monte Carlo methods are commonly used for the parameters inference (Zheng, et al., 2017). And finally, Bayesian optimization techniques, which will be used in the paper as well, are used to search the parameters space more efficiently (Snoek, Larochelle, & Adams, 2012).

ML methods in trading strategies can have different applications from short-term trading based on the neural networks (Chandra & He, 2021), to Extreme Gradient Boosting algorithm (XGBoost) used for long-term trading strategy (Zolotareva, 2021).

The aim of this paper is to propose a trading strategy based on the Extreme Gradient Boosting algorithm (XGBoost) predictions of the BTC Close value, described further in the text, where Bayesian optimization will help select the XGBoost hyperparameters and optimise the proper weights for each n-th day prediction in the final strategy.

2. Data and Methodology

2.1. XGBoost algorithm

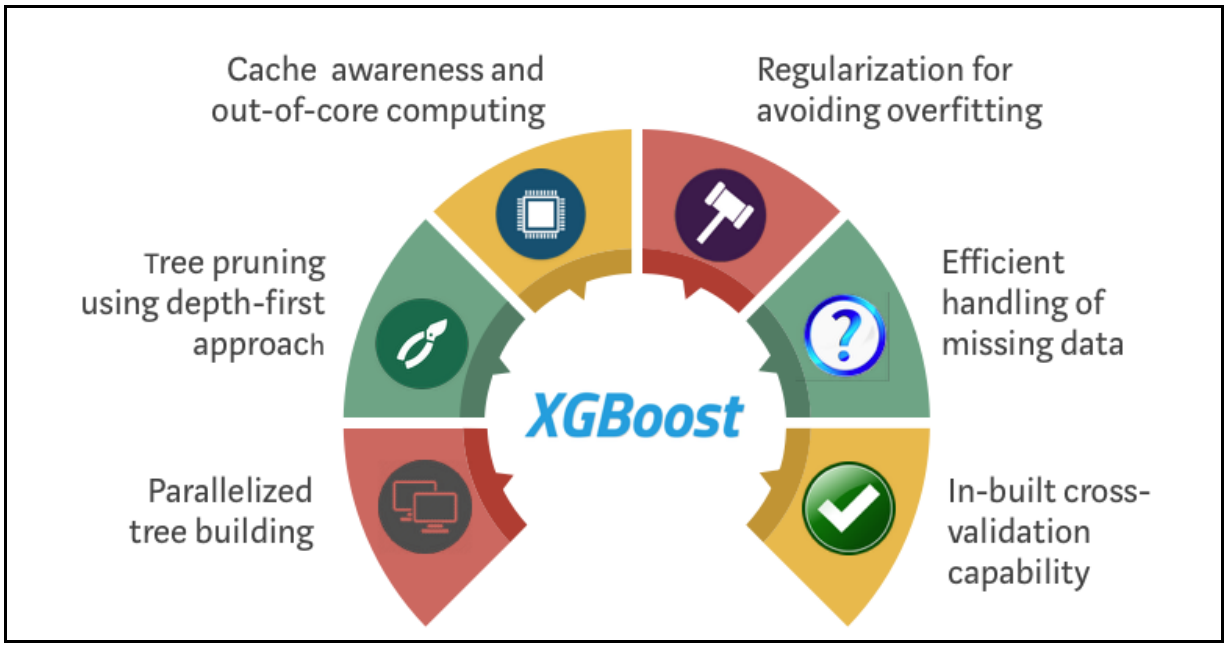
In general, gradient boosting algorithms use so called additive modelling, where in most cases, multiple weak learners (simple models) are estimated one by one into an ensembled model. Every subsequent model aims to explain remaining residuals. The errors are minimized by the Gradient Descent algorithm. Unlike the standard gradient boosted decision trees, Extreme Gradient Boost (XGBoost) algorithm is far better optimized and has several enhancements, which makes it extremely effective both in terms of prediction performance and computational efficiency. The optimization of the algorithm is achieved by (Malik, Harode, & Singh, XGBoost: A Deep Dive into Boosting (Introduction Documentation), 2020) (XGBoost: A Scalable Tree Boosting System, 2016):

- Parallelization
- Tree pruning

With the enhancements in:

- Regularization
- Sparsity Awareness
- Weighted Quantile Sketch
- Cross-validation

Figure 1: XGBoost



Data source: *https://medium.com/time-to-work/xgboost-eafd3beb3fa6*

Even though the XGBoost itself is still relatively new algorithm, it has already been applied to multiple various fields and managed to become an industry standard, mostly for classification tasks, as proved by the Kaggle competitions, where XGBoost dominates many of them. Of course, XGBoost has already some competitor algorithms, such as LightGBM from Microsoft Research or CatBoost from Yandex Technology, with promising performance. Furthermore, in comparison with other ML algorithms, which may be hard to interpret, or it is difficult to measure the importance of the features, XGBoost offers limited ways how to interpret to model, especially importance of the features and its effect on the target variables via Feature Importance plots[2] PDP (Partial Dependence Plots) plots, ICE (Individual Conditional Expectation) plots or SHAPS (SHapley Additive exPlanations) (Güneş, Tharrington, Hunt, & Xing, 2020).

A drawback of the XGBoost model is the total number of tuneable hyperparameters, which may significantly influence the prediction performance of the model or its computational efficiency. Firstly, the most important parameter is booster, where we can specify the type of the weak learner basic model, where possible models are decision tree, linear model, or DART model. The tree specific major hyperparameters, with respect to its Python XGBoost library version, are:

- **Learning rate alias eta** – defines step size shrinkage used in update to prevent overfitting
- **Gamma** – defines minimum loss reduction required to make a further partition on a lead node
- **Lambda** – L2 regularization term on weights. Increasing this value makes the model more conservative and less likely to overfit
- **Alpha** - L1 regularization term on weights. Increasing this value makes the model more conservative and less likely to overfit
- **Number of trees (estimators)** – maximum number of basic models (may not be reached if the early stopping criteria is met). Increasing this value makes the model more complex and more likely to overfit

[2] The Feature Importance plots for all the models may be seen in the Appendix.

- **Grow policy** – controls a way the new nodes are added to the tree (values: deptwise, lossguide)
- **Objective function** - specify the learning task and the corresponding learning objective. Learning task may be regression, classification, survival analysis and other and the corresponding objective may by Squared error, Huber loss for regression, logistic or hinge for classification and others for survival and multinomial classification
- **Max depth** – defines the maximum depth of each tree. Increasing this value makes the model more complex and more likely to overfit

And several minor such as:

- **Scale pos weight** - control the balance of positive and negative weights, useful for unbalanced classes.
- **Colsample** – family of parameters made for subsampling of the features for each estimated model
- **Sampling method** - the method to use to sample the training instances
- **Subsample** - subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration.
- **Minimal child weight** - minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning
- **Max delta step** - maximum delta step we allow each leaf output to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced.
- **Cross validation** – parameter defining umber of cross validated folds
- **Base score** – the initial prediction score of all instances, global bias, for sufficient number of iterations, changing this value will not have too much effect.

  and few others, which can be viewed in more detail at https://xgboost.readthedocs.io/en/stable/python/python_intro.html.

Also, to further fasten the estimation process, parameters **n threads**, or **n jobs** can be used. This will allow us to use the computer processors more effectively, thus making the whole XGBoost model more efficient.

The upper mentioned designation major and minor is defined by the author from the point of the specific task of the Bitcoin Close price prediction. For other tasks, some of the major hyperparameters may occur as minor, having little or no impact on the model, or contrary some minor may be major, for example for heavily unbalanced classification tasks, hyperparameter such as Max delta step or Scale pos weight would most likely become crucial.

The second disadvantage of the XGBoost model is its susceptibility to overfitting, which, when not controlled (again by the appropriate setting of the above-mentioned hyperparameters, such as L1, L2 regularization or cross validation), may lead to almost perfect performance on the training dataset, but modest performance on the test (or validation) dataset. The model would then try to explain most of the observations and variability at all costs, which in most cases causes loss of its general classification abilities on unobserved data. The model then resembles more of a memory than general model (Chen & Guestrin, 2016)

For the Bitcoin price prediction, we will use decision trees as weak learners and we will perform regression modelling to model/predict the Bitcoin close timeseries.

2.2. Hyperparameter optimization principles

As pointed out in the previous paragraph, XGboost hyperparameters should be optimized prior to the estimation of the final model, so as to achieve the best performance. However, several of its hyperparameters may take continuous values within a certain range, or even be theoretically unlimited (though its values will most likely be within an expected range). This gives us relatively wide parametric space, where the optimal parameters may hide. There exist several optimization techniques, which allow us to optimize the hyperparameters. Some of them bring more appealing results, while others computational efficiency (optimization time is reduced n-times). For this work, GridSearch optimization (which is a "Brute force" type of optimization) will be compared with Random Search and finally with Bayesian Optimization, which allows us to search the parameters space fast and find an optimal solution. The Grid Search and Random Search algorithms will be described briefly below, while Bayesian optimization will be explained more through roughly, as it will be also vital part of the prediction's weights assignment.

But firstly, we should specify, which of the above-mentioned parameters are to be optimized. What will be the possible range of each parameter value set for optimization and how we set the rest of the parameters and why.

Table 1: Non-optimized hyperparameters

Hyperparameter	Value	Commentary
Base score	0.5	The base score, for regression task resembles the constant of the linear regression model. For regression should be set as the mean of the y feature.
Booster	gbtree	We want to use the XGBboost based on the decision trees.
Colsample family	1	As we have low number of features, there is no need for their subsampling.
Grow policy	Loss guide	We want to split the nodes according to the loss change.
Max delta step	0	Set to default value as we don't have unbalanced sample.
Minimal child weight	1	We do not want to limit the trees shape, as we have already limited the number of trees and we are optimizing the depth of the trees.
Number of estimators	50	This value is set by the author based on his experience with the XGBoost model. The reason is to have such a number of estimators, that would ensure good regression performance of the model, at the same time feasible hyperparameters optimization and finally less chance of model overfitting.
Objective	squared loss	Set for regression task.
Random state	42	Not a hyperparameter as the others, but the random seed of the estimation is set to this value to ensure repeatability of the model estimation.

And the following hyperparameters will be optimized within the range of values:

Table 2: Optimized hyperparameters – Grid Search

Hyperparameter	Value

Gamma	[0, 0.1, 0.2, 0.4, 0.8, 1.6]
Learning rate	[0.01, 0.03, 0.06, 0.1, 0.2, 0.4, 0.7]
Max depth	[5, 6, 7, 8, 9, 11, 13, 14]
Alpha	[0.4, 0.8, 1.6, 6.4]
Lambda	[0.4, 0.8, 1.6, 6.4]

For the Bayesian Optimization (BO) and Random Search (RS), the above parameters are optimized within the intervals defined by the range of each parameter grid. For the BO and RS, the log-uniform distributions are used for all the optimized hyperparameters except the Max depth parameter, which is the only non-continuous hyperparameter.

Grid search algorithm

The grid search optimization algorithm is considered as one of the easiest to implement and understand. However, this goes along with its computational inefficiency, especially for large number of parameters. Let's assume parameter space $V = (V_1, V_2, V_3, \dots, V_m)$ over which we minimize the loss function of the XGBoost model. A basic approach to perform the Grid Search is to set vectors of lower and upper bounds

$$\begin{aligned} a &= (a_1, a, a_3, \dots, a_m) \\ b &= (b_1, b_2, b_3, \dots, b_m) \end{aligned} \tag{1}$$

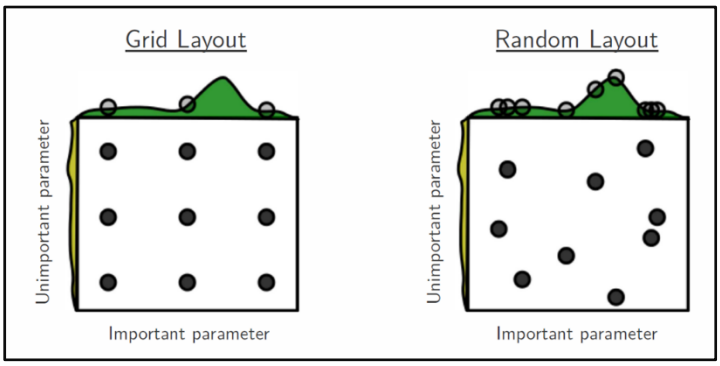
for each parameter V_i. Grid Search then divides each interval $[\mathrm{a_i}, \mathrm{b_i}]$ by the n equally spaced points. This creates a total of $\mathrm{n^m}$ possible grid points to be checked by the algorithm. It is obvious, that the number of evaluations needed for the algorithm rises exponentially with the increase of n and m. We may then lower the n, which will result in possibility that the optimal value of a parameter will be skipped, or we may reduce the number of optimized parameters m. Which may lead to totally non-optimal value of a parameter. Clearly, the Grid Search is not an ideal method for hyperparameters optimization, but can serve as a Benchmark optimization, with which we may compare the others (Dufour & Neves, 2019).

Random search algorithm

Contrary to the Grid Search algorithm, we do not specify the grid. Instead, probability distribution for each parameter is specified, from which the random values of each parameter are drawn. We may specify the probability distribution, such as normal, but in most cases uniform or log-uniform distribution is used.

The algorithm then, instead of using all the values combinations from the grid, randomly samples the parameters combination. The random search algorithm does not test all the available combinations of the parameters, and thus may not reach the best performing vector of parameters. But it was proven, that this algorithm gives good results while heavily optimizing the time spend when performing hyper optimization (Bergstra & Bengio, 2012).

Figure 2: Random Search vs Grid Search



Data source: (Bergstra & Bengio, 2012)

2.3. Bayesian Optimization

In the recent years, Bayesian statistics methods are on the rise, with Bayesian optimization algorithm among the most favourite Bayesian algorithms. In short, the Bayesian optimization idea is rather simple and intuitive: Choose the next input values to evaluate, based on the past results. What does this lead to? The algorithm is narrowing the search space towards the most promising values of the parameters. But still, to avoid possibility of a stuck in the local minimum, allows the search to diverge a bit. To do so, we need to define three important functions. First, an objective function, which is the function we want to optimize. This function is same as in the case of Random Search and Grid Search optimization.

The following two function are specific for the Bayesian optimization and are the crucial parts for the efficient optimization provided by this algorithm:

a) Surrogate Function
b) Acquisition Function

The Surrogate Function is nothing more than a probability model based on the past evaluation results of the objective function. For experimenting with different parameters, the model is used to simulate the function output instead of calling the actual costly function. In most cases the Gaussian Process Regression is used as surrogate, but for example random forest model may be used as well. The Acquisition function then decides whether the given sample is worth evaluation by the costly objective function or not. Mostly used acquisition function is so called “Expected Improvement” or **EI.** Though other functions, which can be used are “Probability of improvement” (**PI)** or “Lower Confidence Bound” (**LCB).** The above-mentioned functions are the reason, why it beats other optimization algorithms. In most optimization tasks, the **Exploration vs Exploitation** strategy needs to be set. What does this mean? When doing the parameters space exploration, the algorithm may converge into the local maximum/minimum of the objective function, where the values of the function are high/low. But, in order to get out of the local extreme, the algorithm needs to have an option to continue with the exploration in different locations. The Exploration vs Exploitation strategy is performed by the acquisition function in an iterative manner. The surrogate function simulates protentional outputs of the objective function and the acquisition function decides which of them will be evaluated. Same as all the other Bayesian approaches, the algorithm is based on the Prior/Posterior concept. The initial prior points are given as an input for the initial run, and with each next step of the iteration is the prior enriched with the calculated posterior points.

The **Expected Improvement** acquisition function can be defined by the following formula. It is important to mention, that the following function has closed form solution under the Gaussian process prior (Snoek, Larochelle, & Adams, 2012).

$$EI(x) = \mathbb{E}\max(f(x) - f(x^+), 0) \tag{2}$$

Where $f(x^+)$ is the value of the best sample given by the x^+, which is $x^+ = argmax_{x_{i \in x_{1:t}}} f(x_i)$.

Analytically under GP prior, EI is:

$$\mathrm{EI}(\mathrm{x}) = \begin{cases} (\mu(\mathrm{x}) - \mathrm{f}(\mathrm{x}^+) - \zeta)\Phi(\mathrm{Z}) + \sigma(\mathrm{x})\phi(\mathrm{Z}) & \text{if } \sigma(\mathrm{x}) > 0 \\ 0 & \text{if } \sigma(\mathrm{x}) = 0 \end{cases} \tag{3}$$

where

$$\mathrm{Z} = \begin{cases} \frac{\mu(\mathrm{x}) - \mathrm{f}(\mathrm{x}^+) - \zeta}{\sigma(\mathrm{x})} & \text{if } \sigma(\mathrm{x}) > 0 \\ 0 & \text{if } \sigma(\mathrm{x}) = 0 \end{cases} \tag{4}$$

And $\mu(x)$, $\sigma(x)$ are mean and standard deviation of the GP posterior, respectively. Φ and ϕ are obviously CDF and PDF of the standard normal distribution respectively. The $(\mu(x) - f(x^+) - \zeta)\Phi(Z)$ part of the equation (3) is stands for exploitation term, while the rest of the equation stands for exploration term. For more details see (Brochu, Cora, & de Freitas, 2010).

For more details about Gaussian surrogate function see for example (Gramacy, 2020).

2.4. Data

For the trading model development, time-series of the Bitcoin daily metrics is used from **1th January 2019** to **12th January 2022**, where we split the dataset on the training, validation, and test parts by the following key:

Training dataset: first 85% of the dataset

Validation dataset: following 5% of the dataset

Test dataset[3]: remaining 10% of the dataset

[3] The test dataset was further enhanced up to 25.3. 2022 to obtain longer testing period.

Figure 3: Dataset

Data source: own elaboration

The reason for the validation part is to perform the optimization of the prediction's weights into the final model on a separate dataset. This will ensure that the optimization of the weights will be done within the time frame, which was not used for the training of the XGBoost models, where the models may be overfitted.

The test period will be used to perform the final trading strategy and evaluate the results.

As we are going to use several basic technical indicators, we must calculate them first. The following technical indicators will be used as the features in the models:

Volume
Represents the traded amount during the given period (mostly one day).

Close price lag 1
Lagged Close price by one day.

Close price lag 2
Lagged Close price by two days.

Figure 4: BTC price development

Data source: own elaboration

SMA

Simple Moving Average indicator represent average of the asset prices during a given period (5, 10, 15, 30).

EMA

Exponential Moving Average indicator is type of moving average, where the most recent observations have the highest significance. Contrary to the SMA, EMA reacts faster to new trends in the asset price.

Figure 5: Technical indicators

Data source: own elaboration

MACD indicator and signal

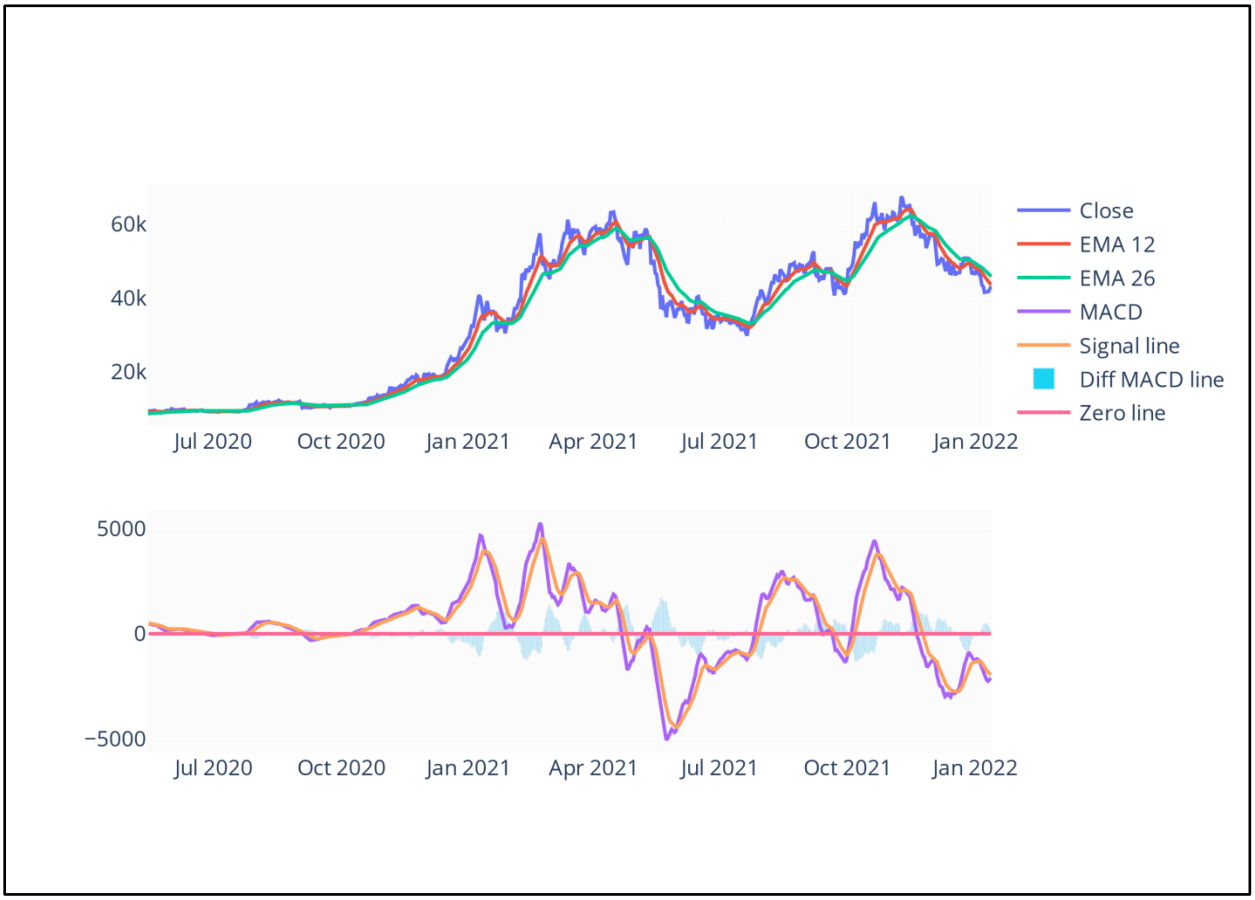
Moving average convergence divergence indicator is categorized as trend-following momentum indicator. This indicator represents the relationship between two exponential moving averages (**EMA**). The day parameter for both EMAs may be theoretically set arbitrarily, but in most applications 26-day period and 12-day period EMAs are calculated. The MACD is then calculated as follows

$$MACD = 12_{period}EMA - 26_{period}EMA, \tag{5}$$

Furthermore, the MACD signal line calculated as the 9-day EMA of the MACD values is calculated.

In most basic technical trading strategies, the Buy signal is generated when the MACD signal line crossed the MACD from below and the sell signal is generated in the opposite situation.

Figure 6: MACD technical indicator



Data source: own elaboration

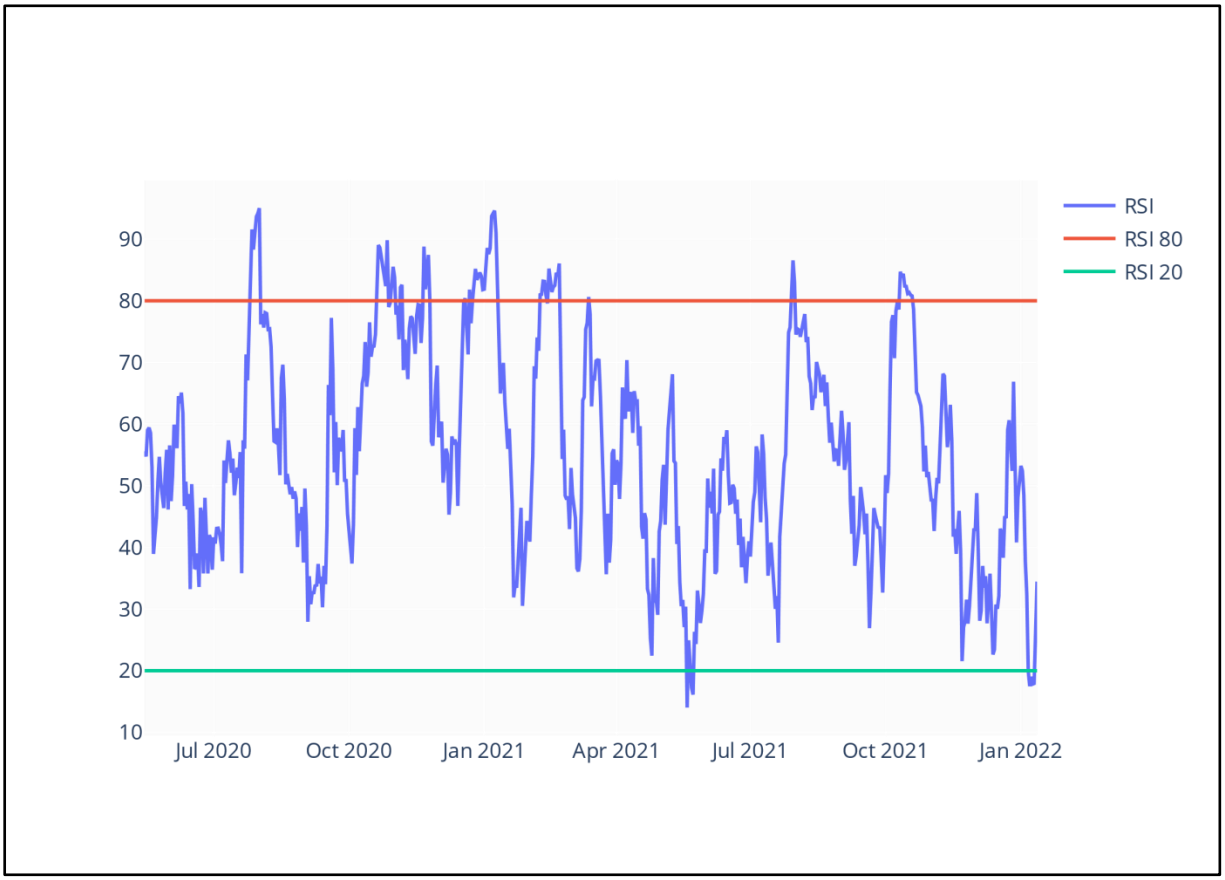
RSI

Relative strength index is a momentum indicator, which measures the magnitude of the recent price changes. RSI then indicates whether the market is overbought or oversold. I most cases, the RSI is represented as an oscillator. In most cases, the limiting values of RSI for overbought/oversold market indication are 70/30 or 80/20.

$$RSI = 100 - \left(\frac{100}{1+\frac{Average\_gain(n)}{Average\_loss(n)}}\right), \tag{6}$$

Where $Average\_gain(n)$ function is average of all the positive changes in the price of the asset during the n-th day lookback period. Similarly, the $Average\_loss(n)$ function is taken as the average of the negative changes.

Figure 7: RSI technical indicator



Data source: own elaboration

Also, we must propose the evaluation metrics, which will serve in comparison of the trading strategies.

First, and most simple, the **Cumulative Profit of the strategy**, will be the first metric. This metric is rather non-informative, as it does not include the risk of the investment. However, we still want to know the total Cumulative profit of our investment.

Second will be widely known **Sharpe** and **Sortino** ratios, which already include risk and are often used to compare investment or trading strategies.

Finally, we will also measure **number of trades** of our trading strategy, which will be further divided into **profitable** and **non-profitable** ones. Also, **number of trading days** will be measured, as the trading strategy has the option not to trade in a given day (short selling is not allowed), contrary to Buy & Hold strategy. Of course, number of trading days influences the Sharpe and Sortino ratios, as the standard deviation i.e., volatility is part of the calculation and for the trading strategy, only days when the algorithm holds the position are taken into account when calculating volatility.

2.5. Proposed trading strategy

With the above theory, we propose a following algorithmic trading strategy, which will be tested and compared with the Buy & Hold (**BAH**) strategy.

First, we estimate six XGBoost models, where n^{th} day value of the asset timeseries will be predicted. To explain the principle of n^{th} day predicted value, we will introduce the example with the 1^{st} day

predicted value, where all the remaining predicted values will be estimated reciprocally. With the above-described dataset, the target variable for the 1^{st} day prediction is following. For each set of features x at time t, the target variable of the model is the Close price of the asset at the time t+1, e.g., with the today's information, we are predicting tomorrow's Close price. In the same way, we may attempt to predict any future Close price values of the asset timeseries.

For our trading strategy, we will estimate following vector of n^{th} day predicted Close price values: {1,2,5,10,20,30}. Let's remind briefly, that for each n^{th} day prediction, unique model is estimated. Afterwards, we will get the final logarithmic return prediction as the weighted average price of all the six predictions compared to today's close value. Someone might object, that the 30-day logarithmic return prediction should not be taken on an equal with 1-day logarithmic return, as we hold the asset longer and so we expect higher return. This is undoubtedly true, but as one of our major objectives in this paper is to optimize the weights of the predictions anyway, the optimization itself should handle this. As well as the deterioration in the prediction performance with the rising n of days, as we expect that further the prediction in the future is estimated, the higher the error rate of the model is (which is quite reasonable).

Finally, with predicted logarithmic returns and optimized weights, we calculate expected logarithmic return indicating rise or decline in the asset's price. The algorithmic strategy then goes with the following simple rules:

1) If the final logarithmic return prediction is **POSITIVE** and the algorithm **DOES NOT HOLD** the asset from the previous days, it will **BUY** the asset.
2) If the final logarithmic return prediction is **POSITIVE** and the algorithm **DOES HOLD** the asset from the previous days, it will **HOLD** the asset for another day.
3) If the final logarithmic return prediction is **NEGATIVE** and the algorithm **DOES HOLD** the asset from the previous days, it will **SELL** the asset.
4) If the final logarithmic return prediction is **NEGATIVE** and the algorithm **DOES NOT HOLD** the asset from the previous days, it will **DO NOTHING**.

Where Short selling is not allowed.

What is relatively clear is the fact, that even with the very precise algorithmic trading strategy, for the asset, which will be bullish most of the time, when the backtesting of the strategy takes place, the BAH strategy may generate better results (and vice versa for bear asset, where BAH will be non-profitable). This is due to the several reasons, existence of the bid-ask spread, fees for each trade and an error rate (even the best model has some). That is why we need an asset, which is very volatile (which cryptocurrencies in general are). Next, the selected backtesting period, its start and end date respectively, heavily influence the result of both BAH and algorithmic strategy.

Also, during a practical deployment of the strategy, there should be several hard KOs rules in order not to sustain heavy losses. For example, when the number of non-profitable trades exceeds a threshold based on the backtesting during a certain period (more specifically the ratio of profitable and non-profitable trades per unit of time). Or, when the prediction error during the past days strongly surpasses the error metric (RMSE, MSE) indicating deteriorative performance of the model. Of course, there may exist other rules, but as we are aiming primarily on the comparison of raw strategies and the advantages of the XGBoost and Bayesian optimization, these will not be included in the backtesting (Chan, 2009).

It is important to mention, why we estimate six models in total. The reason is simple, we want to somehow smooth the trading strategy, with the long-term models serving rather than an absolute

value prediction as a state prediction (a sort of analogy to a Markov Switching model). We expect that this will force the strategy to make less trades and hold the asset longer, saving the costs per trade (ignoring small within or one day fluctuations). Nevertheless, as the final weights will be decided by the optimization algorithm, it may happen that the final strategy will be guided primarily by short term one- or two-days predictions, without the smoothing effect.

3. Results and Discussion

All the calculations and model estimations are performed in the Python programming language, within its Jupyter and PySpark interpreters.

3.1. Comparison of hyperparameters optimization algorithms

Before we will construct the XGBoost models and the trading model itself, let's compare the possible XGBoost optimization techniques. The comparison will be performed on the first estimated XGB model, with the 1st day prediction target. All three algorithms were tested on the same dataset, with the same combination of optimized hyperparameters and the same values of the non-optimized ones. The following table shows the Mean-squared error (MSE) metric on test dataset and so-called wall-time of each algorithm run. Wall time is equal to the elapsed time in the real world (there exists also CPU time or system time).

Table 3: Optimization algorithms - comparison[4]

Algorithm	MSE test	Wall time
Grid Search	3447.1	45min 7s
Random Search	3725.2	1min 2s
Bayesian optimization	3273.3	6min 8s

As can be seen, the difference between the algorithms is tremendous and fulfils our expectations both from the point of the prediction accuracy and computational power demand. The random search finds its optimal parameter combination incredibly fast. The Grid Search algorithm is more accurate than Random Search, but it takes him long time to converge, even though we are optimizing rather small number of hyperparameters in quite narrow intervals. Finally, as expected, Bayesian optimization finds the most suitable combination of hyperparameters (the MSE is almost 12% lower than in case of Random Search) within relatively short time, compared to Grid Search (it takes more than five times less time than Grid Search). We may conclude that not only optimization of the hyperparameters is extremely important for XGBoost performance as such, but also the selected optimization algorithm may improve the performance of the model significantly and at the same time shorten the necessary time for the optimization (the difference in wall time between the GS and BO algorithms would even increase in case of more hyperparameters optimized).

For visual comparison of the 1-day prediction XGboost models on the validation and test data combined, see the graphs below.

[4] For both Bayesian Optimization and Random Search, 50 iteration steps were set as an upper limit to be comparable.

Figure 8: Bayesian optimization

Data source: own elaboration

Figure 9: Random Search optimization

Data source: own elaboration

Figure 10: Grid Search optimization

Data source: own elaboration

It is important to mention, that the hyperparameter optimization was done on the validation sample as well as the strategy optimization, whereas the training of all the models was performed on the training dataset. This should ensure that the model would not be highly overfitted.

3.2. Comparison of Buy & Hold strategy and trading strategy with multiple weights vectors

Now, we may estimate six separate XGBoost models with Bayesian optimized hyperparameters (each XGBoost has its own set own hyperparameters) and build the final trading strategy. As mentioned above, we will calculate the proxy of the future close price, given by our model predictions and consequently proxy for the nth day logarithmic return with the following logic:

$$Logret_{proxy\ t+n} = \ln\left(\frac{XGBpred_n}{Close_t}\right) \tag{7}$$

where each nth day prediction $Logret_{proxy\ t+n}$, given by a separate model, must have certain weight w_n, which determines the total effect of the $Logret_{proxy\ t+n}$ on the $Logret_{proxy}$ prediction. We calculate the proxy for logarithmic return as

$$Logret_{proxy} = w_1 Logret_{proxy\ t+1} + w_2 Logret_{proxy\ t+2} + w_5 Logret_{proxy\ t+5} + w_{10} Logret_{proxy\ t+10} + w_{20} Logret_{proxy\ t+20} + w_{30} Logret_{proxy\ t+30} \quad (8)$$

Where the $Logret_{proxy\ t+n}$ is given by the previous formula and Close_t is realized Close value at time t. The $Logret_{proxy}$ value is then the input into trading strategy described previously in the chapter **Chyba! Nenalezen zdroj odkazů.**. To properly test all the possibilities of the trading strategy, and to reveal its true maximum potential, we will test the following vectors of weights.

Table 4: Trading strategies

Trading strategy	Weights	Comment
0	/	Buy and Hold strategy as the benchmark
1	Equal weights	Each prediction has equal weight of 1/6
2	MSE proportional weights	Each weight is given inversely by the MSE of the XGBoost model. This means, that the most accurate model has the highest weight and vice versa.
3	Bayesian optimized weights – Total profit	All the weights are optimized within separate Bayesian optimization. The optimized criterion is total profit of the strategy.
4	Bayesian optimized weights – Sharpe ratio	All the weights are optimized within separate Bayesian optimization. The optimized criterion is the Sharpe ratio of the strategy.

Overall, we are going to test 4 trading strategies, based on the same XGBoost predictions, plus the B&H as benchmark. Our expectation is that the trading strategies with equal weights will be the least efficient both from the point of total profit and Sharpe or Sortino ratio. Also, we expect the Bayesian optimized strategy with Sharpe ratio taken as the optimization criterion as the most efficient. In the Table 7, the total profit, Sharpe ratio and other observed criteria (number of trades, trading days etc.). We expect all the algorithmic strategies to be more efficient than the B&H strategy. The following table shows the MSE weights derived from the MSE of the XGBoost models.

Table 5: MSE weights for the final trading strategy

Model	MSE	MSE^{-1}	Sum MSE^{-1}	Weight
XGB n1	3273.3	0,000305502	0,001011094	30,2%
XGB n2	4109.6	0,000243333		24,1%
XGB n5	6358.6	0,000157267		15,6%
XGB n10	8753.8	0,000114236		11,3%
XGB n20	9418.5	0,000106174		10,5%
XGB n30	11822.9	8,45816E-05		8,4%

Table 6: BO weights for the final trading strategy

Model	BO - Profit	BO – Sharpe ratio
XGB n1	9.78%	26.78%
XGB n2	14.17%	14,69%
XGB n5	0.51%	6,75%
XGB n10	23.55%	0,55%

XGB n20	43.75%	4,85%
XGB n30	8.23%	46,36%

For the trading strategy, we assume we have exactly 1 unit of any currency/portfolio at the beginning of the trading period. This means that the hypothetical value of total profit 1.1 indicates that the value of the portfolio has risen by 10 % over the trading period.

Table 7: Performance of the trading strategies

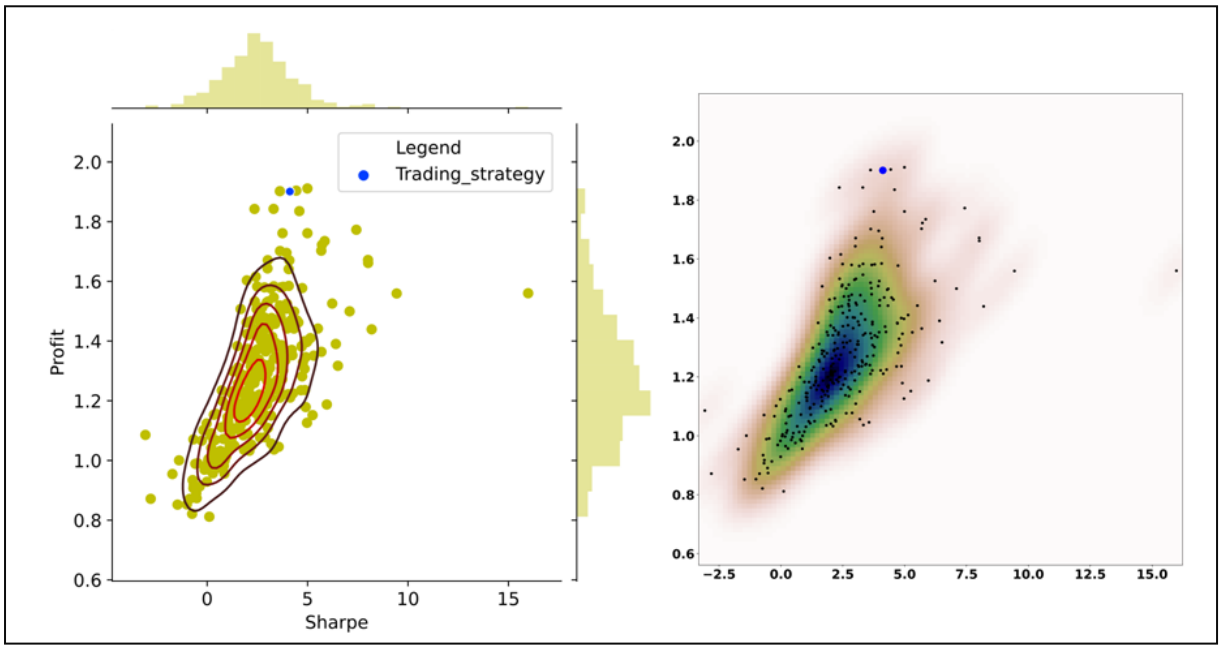
Trading strategy	Strategy name	Total Profit Gross	Sharpe ratio Net[5]	Sortino ratio	No. of trades	No. profitable trades	No. loss trades	No. trading days
0	**B&H**	1.511	0.913	1.377	/	/	/	156
1	**Equal Weights**	1.322	1.798	1.771	5	3	2	74
2	**MSE Weights**	1.195	1.148	1.144	5	4	1	74
3	**BO weights – Profit optimized**	1.623	3.169	2.891	4	4	0	73
4	**BO weights - Sharpe ratio optimized**	1.274	1.544	1.547	4	4	0	73

The results shown in the Table 7 show that from the point of total profit, both Bayesian Optimized strategies beat the B&H on the test dataset. This is not surprising, because the weights for the strategies were optimized on the valid sample to maximize the profit and Sharpe ratio. With all the trades profitable and the highest Sharpe ratio, the BO Sharpe ratio strategy is by far the best one. The second BO trading strategy (BO Profit optimized) gives also all the trades profitable, but overall performs worse than the BO Sharpe optimized strategy. The MSE weighted trading strategy performs the worst. Along with the final BO weights in the Table 6, it is shown, that the XGBoosts, which predict the more distant Close value have higher MSE, are more suitable to be used in the final strategy than the short-term ones. This proves our expectations, that the longer-term predictions will smooth the trading strategy from the short-term fluctuation and cause the trading strategy to trade less with longer trades.

To further verify the robustness of the trading strategy, we create 10 000 random trades vectors, each with the six trades such as our final BO Sharpe optimized strategy. For each of the random trades vector, we will calculate the Profit and Sharpe ratio. Afterwards, we create the empirical multivariate distribution given by the Profit and Sharpe ratio univariate distributions and examine, whether our strategy is in fact rare, which would indicate non-randomness of our strategy, or whether our final strategy combination of Profit and Sharpe ratio may be easily achieved by a random guess.

[5] The Net Profit is calculated with the fees and spread of the BTC/USD included, based on the Bitfinex Crypto Exchange. The fee per trade is 0.1% of the traded volume and the average spread during the trading period was 0.004%. The average BTC/USD spread on the biggest Crypto Exchanges ranges from 1.5 to 20, with Bitfinex on the lower side with 1.9 average spread during the trading period.

Figure 11: Empirical multivariate distribution of the simulations for the original trading period – BO Sharpe strategy



Data source: own elaboration

Plus, the Cumulative returns of both the B&H strategy and our trading strategy can be seen in the graph below. The portfolio initianl value is set to 100k of the fiat currency units.

Figure 12: Cumulative returns of the B&H and trading strategy – BO Sharpe optimized

Data source: own elaboration

Figure 13: Cumulative returns of the B&H and trading strategy – BO Profit optimized

Data source: own elaboration

As can be seen in the graphs above[6], our trading strategy marked by the blue dot is outside of the 90% multivariate empirical confidence interval, which indicates that our combination of Profit and Sharpe ratio is not easily achievable by a random guess.

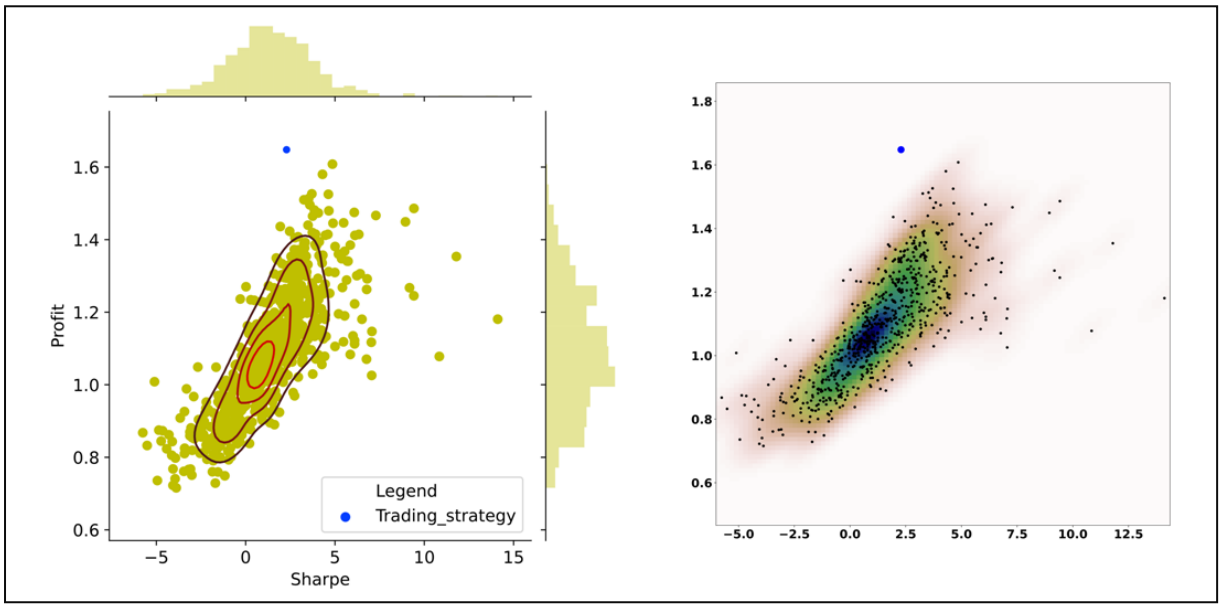
Finally, to obtain longer (and thus more robust) testing period, the testing window was additionally extended from original ending date 12.1.2022 up to 25.3. 2022, with additional 66 days for potential trading strategy performance testing. The results and graphs as in case of the original trading period can be seen below, with additional cumulative returns for both the B&H strategy and our final trading strategy.

Table 8: Performance of the trading strategies – extended testing period

Trading strategy	Strategy name	Total Profit Gross	Total Profit Net	Sharpe ratio	Sortino ratio	No. of trades	No. profitable trades	No. loss trades	No. trading days
0	**B&H**	0.984	/	-0.076	-0.107	/	/	/	222
1	**Equal Weights**	1.161	1.135	0.736	-0.691	11	8	3	96
2	**MSE Weights**	1.108	1.083	0.502	-0.607	11	7	4	96
3	**BO weights – Profit optimized**	1.372	1.350	1.717	-0.079	8	6	2	83
4	**BO weights - Sharpe ratio optimized**	1.648	1.602	2.289	1.273	13	11	2	101

[6] The empirical distributions are estimated from all 10 000 simulations, however the dots are based only on the part of the dataset, so the graphs are concise.

Figure 14: Empirical multivariate distribution of the simulations for the extended trading period – BO Sharpe strategy

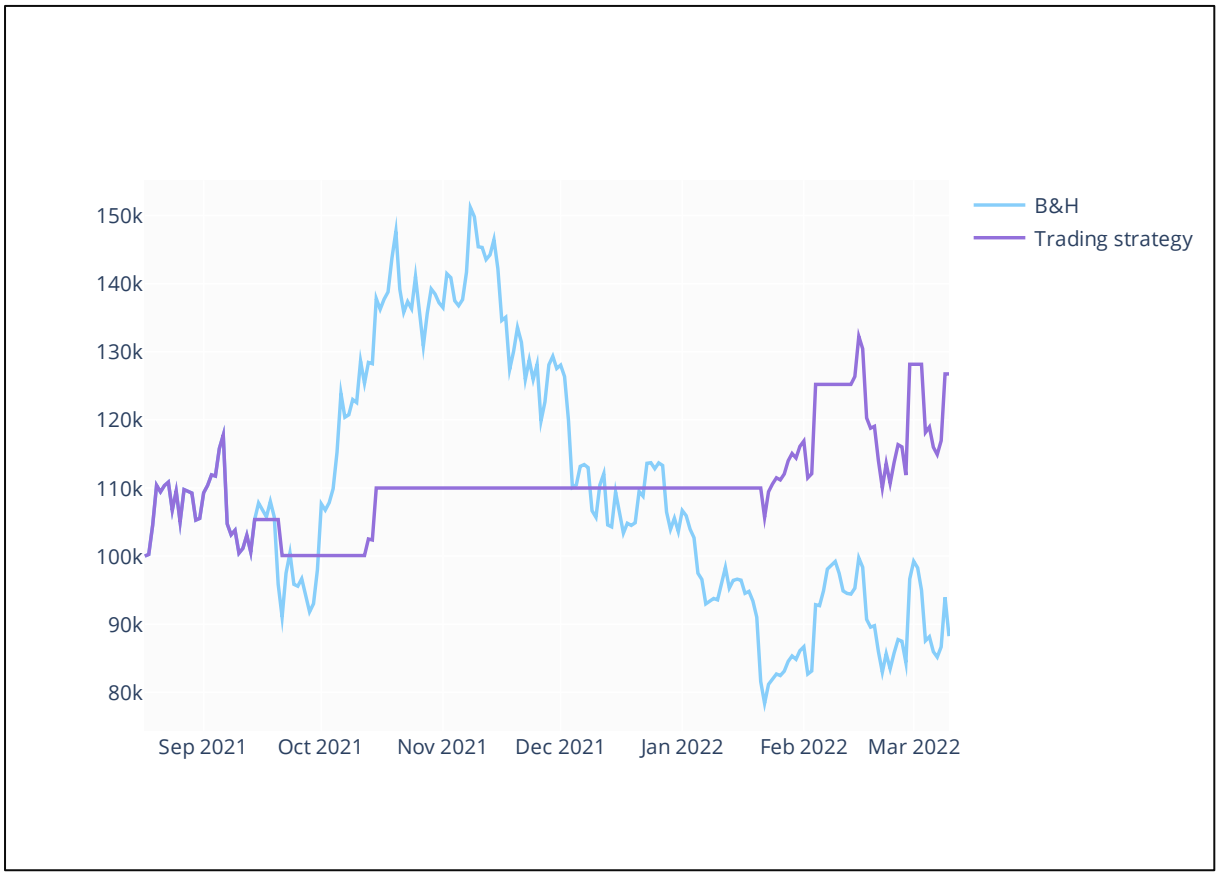


Data source: own elaboration

Figure 15: Cumulative returns of the B&H and trading strategy – BO Sharpe optimized (extended trading period)

Data source: own elaboration

Figure 16: Cumulative returns of the B&H and trading strategy – BO profit optimized (extended trading period)



Data source: own elaboration

We can see that our strategy (especially the BO Sharpe optimized one) continued with its top performance. Also, we can see that both trading strategies stopped the trading slightly before the peak of the BTC price during autumn 2021. This is most likely caused by the previously mentioned all time high of the BTC price, where the XBoost struggle with the precise price prediction. On the other hand, the strategy correctly revealed the fall in the BTC price and waited with the trading until the rise in January and February 2022.

The enhanced trading window along with the multivariate distribution testing proved the potential good performance of the trading strategy.

4. Conclusion

Despite its complexity and the indisputable fact, that XGBoost model remains huge black box, it was shown that this model may be successfully incorporated into the thriving trading model, compared to standard B&H strategy, especially in case of highly volatile asset. Though, it has its limitations, such as high number of hyperparameters to be optimized, propensity to overfitting and inferior extrapolation performance, it proves to be one of the most promising ML algorithm these days, even during the COVID-19 pandemic period.

It was shown that the hyperparameters optimization techniques is vital and fundamental part of the ML model building, which can immensely improve its performance. With Bayesian techniques showing very promising results, proving increasing importance of this part of the statistics in the recent years.

The Sharpe and Sortino ratios, total profit, show that the final trading strategy, with Bayesian optimized weights of the XGBoost predictions is very promising and may offer benefits in comparison with B&H strategy, especially for high volatility cryptocurrencies such as BTC. Even though, there was used low number of simple technical indicators as features for the XGBoost models, the performance of the models was sufficient in terms of prediction power, also proved on the enhanced trading period. Also, as the fees and spreads on most of the important Crypto exchanges are not high, the application of the algorithmic strategy may not be limited by them. The biggest drawback of this model is the extrapolation problematics of the Decision Tree based XGBoost models. The asset must undergo an all-time peak in the past, specifically within the training period, in order for the model to learn the full scale of the potential values of both, the target variable and the input features. This disqualifies the applicability of the model for the majority of the stocks, which are in most periods rising and thus reaching new hights. The model may be applied to fiat currency pairs, such as USD/EUR etc. but these pairs lack the high volatility of the cryptocurrencies. Intuitively, the solution of this problem seems to be the change of the target variable, from the non-stationary Close value to stationary Log-returns or difference Close value. However, the reason for the target variable being the Close value is the existence of the memory within the time-series, which would not be, most likely, present within the stationary time-series of the log-returns, not allowing us to predict the Log-returns in the more distant future. Given by the previously mentioned problematics, the only feasible solution of the problem is, from our point of view, application of the fractionally differenced timeseries which solves the non-stationarity problem and at the same time avoids obliteration of the long-term memory of the time-series.

For further research, it may be worth to attempt to include fractional differencing, more complex indicators, such as entropy, fractal dimension or Hurst exponent as well as to combine the ML algorithms with more traditional statistical approaches, such as Markov Switching models, State space models. Or take even more advanced ML approach, for example with Gated Bayesian Neural Networks, which can simulate the asset behaviour in multiple states (high – low volatility, bear-bull market etc.) and do not suffer from the extrapolation issue such as XGBoost in this paper does.

5. References

[1] Ayala, J., García-Torres, M., Noguera, J. L., Gómez-Vela, F., & Divina, F. (2021). Technical analysis strategy optimization using a machine learning approach in stock market indices. Knowledge-Based Systems, 225, p. 107119.

[2] Bergstra, J., & Bengio, Y. (2012, 03). Random Search for Hyper-Parameter Optimization. The Journal of Machine Learning Research, 13, pp. 281-305.

[3] Brochu, E., Cora, V. M., & de Freitas, N. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. ArXiv, abs/1012.2599.

[4] Dufour, J.-M., & Neves, J. (2019). Chapter 1 - Finite-sample inference and nonstandard asymptotics with Monte Carlo tests and R. In H. D. Vinod, & C. R. Rao (Eds.). Elsevier.

[5] Giudici, G., Milne, A., & Vinogradov, D. (2019, 09). Cryptocurrencies: market analysis and perspectives. Journal of Industrial and Business Economics, 47.

[6] Gramacy, R. (2020, 03). Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences.

[7] Güneş, F., Tharrington, R., Hunt, R., & Xing, E. (2020). How to Explain Your Black-Box Models in SAS. SAS GLOBAL FORUM Paper SAS4502-2020.

[8] Chan, E. (2009, 01). Quantitative Trading : How to Build Your Own Algorithmic Trading Business / E.P. Chan.

[9] Chandra, R., & He, Y. (2021, 07). Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. PLOS ONE, 16(7), pp. 1-32.

[10] Chang, K. C., & Tian, Z. H. (2015). Market analysis and trading strategies with Bayesian networks. 2015 18th International Conference on Information Fusion (Fusion), pp. 1922-1929.

[11] Chen, M.-H., Huang, L., Ibrahim, J., & Kim, S. (2008, 08). Bayesian Variable Selection and Computation for Generalized Linear Models with Conjugate Priors. Bayesian analysis (Online), 3, pp. 585-614.

[12] Chen, T., & Guestrin, C. (2016). XGBoost: {A} Scalable Tree Boosting System. CoRR, abs/1603.02754.

[13] Jaquart, P., Dann, D., & Weinhardt, C. (2021). Short-term bitcoin market prediction via machine learning. The Journal of Finance and Data Science, 7, pp. 45-66.

[14] Malik, S., Harode, R., & Singh, A. (2020, 02). XGBoost: A Deep Dive into Boosting (Introduction Documentation).

[15] Rashmi, K., & Gilad-Bachrach, R. (2015, 05). DART: Dropouts meet Multiple Additive Regression Trees.

[16] Snoek, J., Larochelle, H., & Adams, R. (2012, 06). Practical Bayesian Optimization of Machine Learning Algorithms. Advances in Neural Information Processing Systems, 4.

[17] Trassinelli, M. (2020, 01). An introduction to Bayesian statistics for atomic physicists. Journal of Physics: Conference Series, 1412, p. 062008.

[18] Zhang, X., Johnson, T., Little, R., & Cao, Y. (2010, 03). A Bayesian Image Analysis of Radiation Induced Changes in Tumor Vascular Permeability. Bayesian analysis (Online), 5, pp. 189-212.

[19] Zheng, X., Zaheer, M., Ahmed, A., Wang, Y., Xing, E., & Smola, A. (2017, 11). State Space LSTM Models with Particle MCMC Inference.

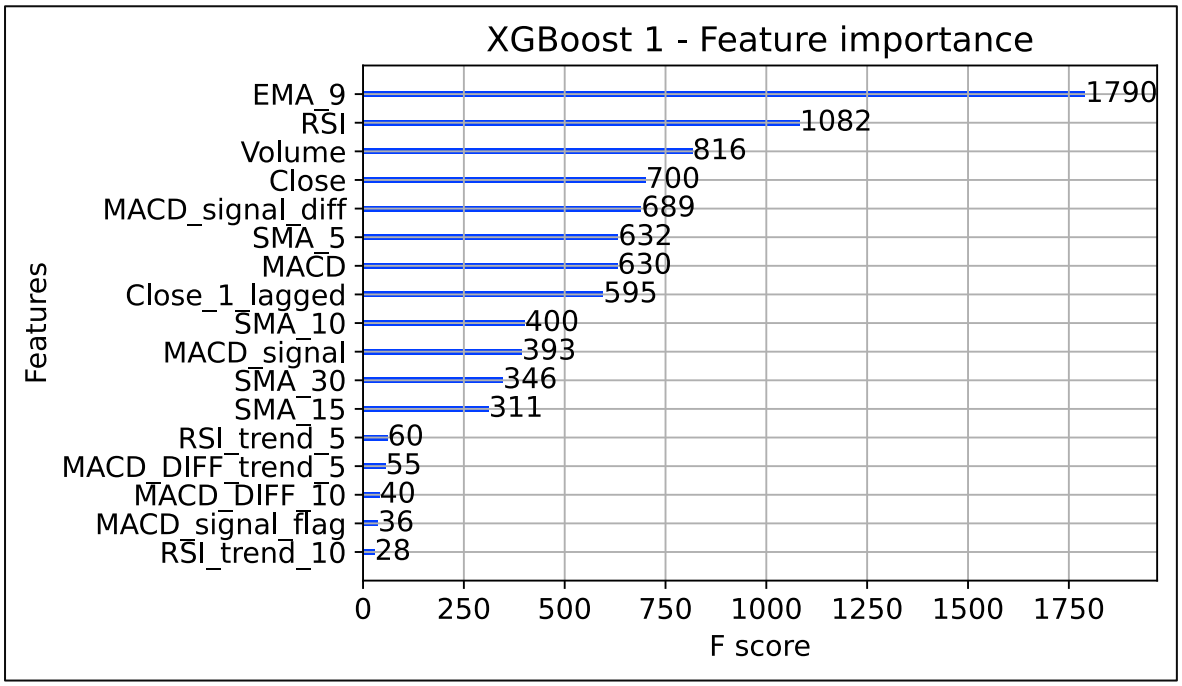
[20] Zolotareva, E. (2021, 10). Aiding Long-Term Investment Decisions with XGBoost Machine Learning Model.

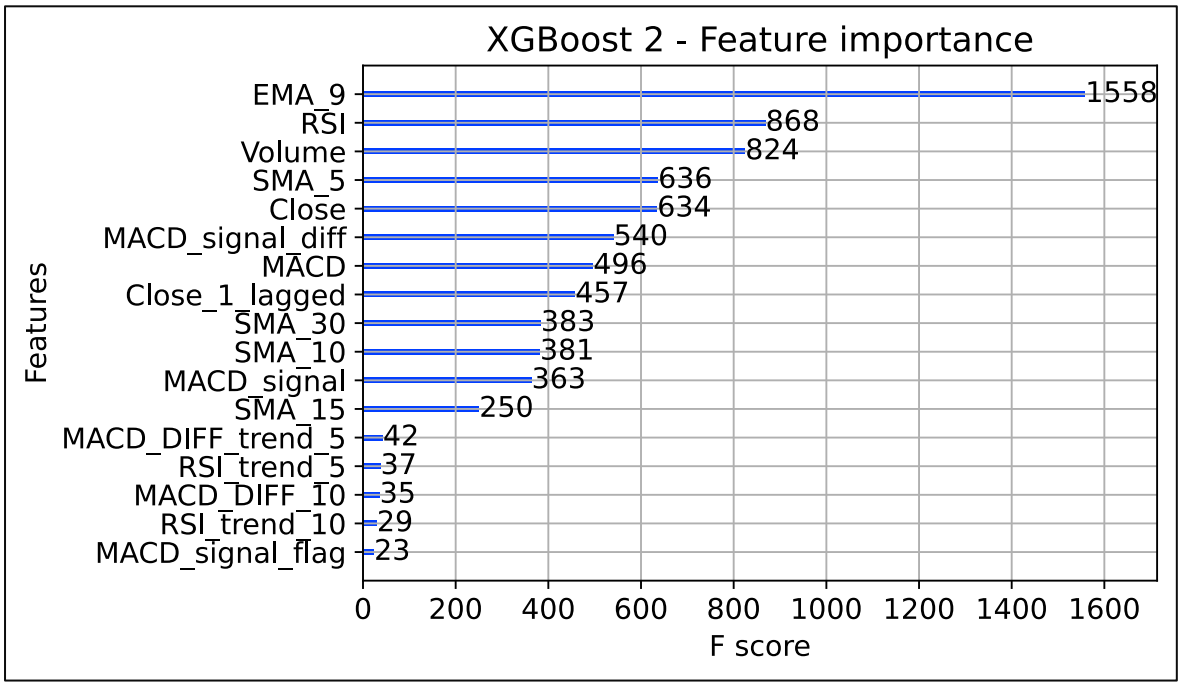
6. Appendices

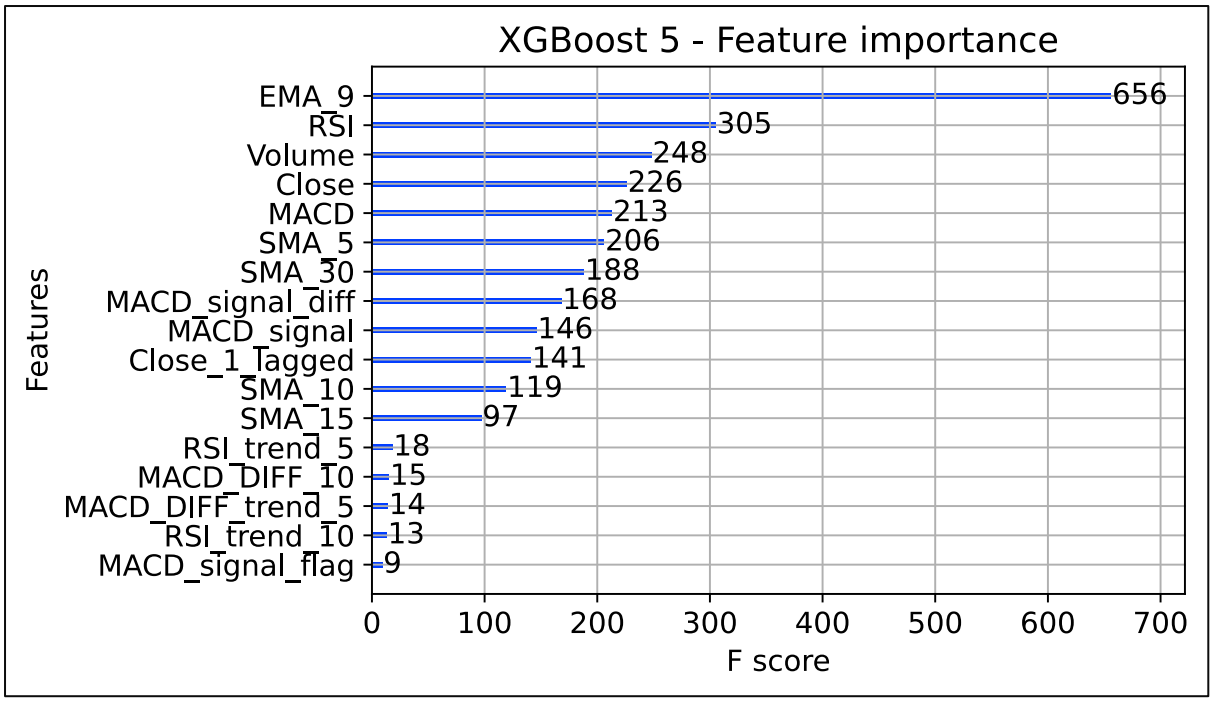
6.1. Appendix 1: XGBoost variables PDP and Feature importance plots

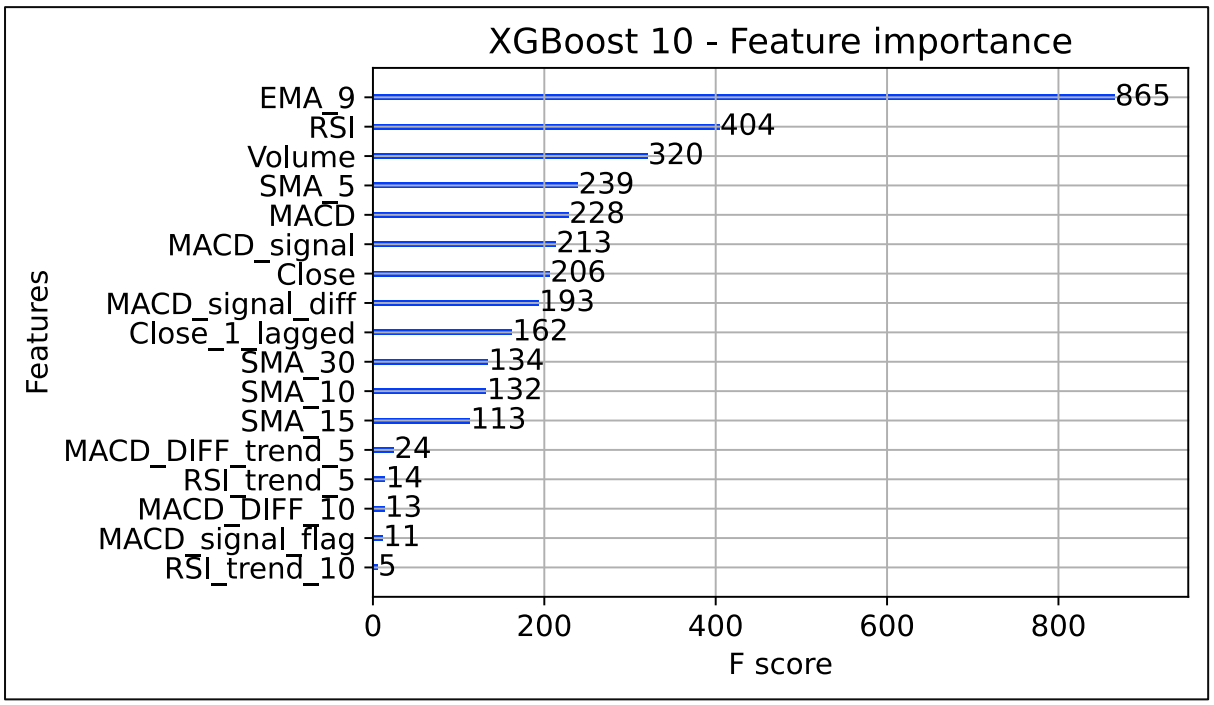
Feature importance plots for each XGboost.

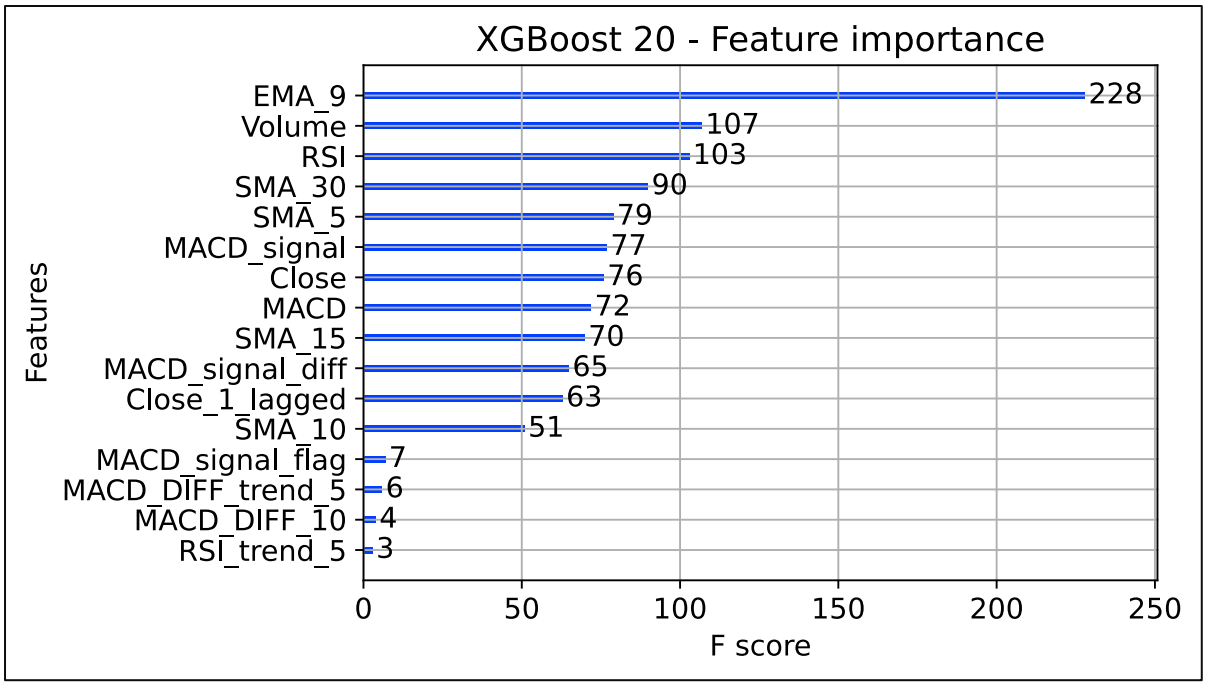
Figure 17: Feature importance graphs for each XGBoost model

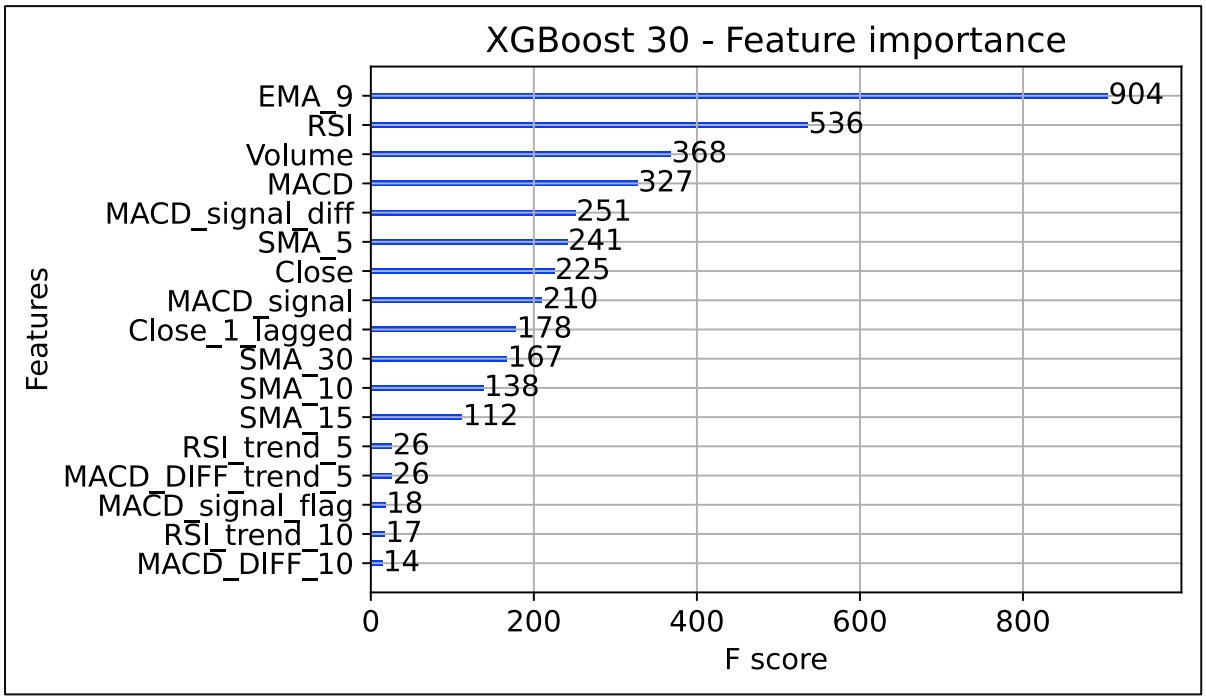





XGBoost 5 - Feature importance
Features
EMA_9
RSI
Volume
Close
MACD
SMA_5
SMA_30
MACD_signal_diff
MACD_signal
Close_1_lagged
SMA_10
SMA_15
RSI_trend_5
MACD_DIFF_10
MACD_DIFF_trend_5
RSI_trend_10
MACD_signal_flag
656
305
248
226
213
206
188
168
146
141
119
97
18
15
14
13
9
0
100
200
300
400
500
600
700
F score


XGBoost 10 - Feature importance
Features
EMA_9
RSI
Volume
SMA_5
MACD
MACD_signal
Close
MACD_signal_diff
Close_1_lagged
SMA_30
SMA_10
SMA_15
MACD_DIFF_trend_5
RSI_trend_5
MACD_DIFF_10
MACD_signal_flag
RSI_trend_10
865
404
320
239
228
213
206
193
162
134
132
113
24
14
13
11
5
0
200
400
600
800
F score


XGBoost 20 - Feature importance
EMA_9
Volume
RSI
SMA_30
SMA_5
MACD_signal
Close
MACD
SMA_15
MACD_signal_diff
Close_1_lagged
SMA_10
MACD_signal_flag
MACD_DIFF_trend_5
MACD_DIFF_10
RSI_trend_5
228
107
103
90
79
77
76
72
70
65
63
51
7
6
4
3
Features
0
50
100
150
200
250
F score


XGBoost 30 - Feature importance
EMA_9
RSI
Volume
MACD
MACD_signal_diff
SMA_5
Close
MACD_signal
Close_1_lagged
SMA_30
SMA_10
SMA_15
RSI_trend_5
MACD_DIFF_trend_5
MACD_signal_flag
RSI_trend_10
MACD_DIFF_10
904
536
368
327
251
241
225
210
178
167
138
112
26
26
18
17
14
Features
0
200
400
600
800
F score

FFA Working Paper Series

2019

1. Milan Fičura: Forecasting Foreign Exchange Rate Movements with k-Nearest-Neighbor, Ridge Regression and Feed-Forward Neural Networks.

2020

1. Jiří Witzany: Stressing of Migration Matrices for IFRS 9 and ICAAP Calculations.
2. Matěj Maivald, Petr Teplý: The impact of low interest rates on banks' non-performing loans.
3. Karel Janda, Binyi Zhang: The impact of renewable energy and technology innovation on Chinese carbon dioxide emissions.
4. Jiří Witzany, Anastasiia Kozina: Recovery process optimization using survival regression.

2021

1. Karel Janda, Oleg Kravtsov: Banking Supervision and Risk-Adjusted Performance in the Host Country Environment.
2. Jakub Drahokoupil: Variance Gamma process in the option pricing model.
3. Jiří Witzany, Oľga Pastiranová: IFRS 9 and its behaviour in the cycle: The evidence on EU Countries.

2022

1. Karel Janda, Ladislav Kristoufek, Binyi Zhang: Return and volatility spillovers between Chinese and U.S. Clean Energy Related Stocks: Evidence from VAR-MGARCH estimations.
2. Lukáš Fiala: Modelling of mortgage debt's determinants: The case of the Czech Republic.
3. Oľga Jakubíková: Profit smoothing of European banks under IFRS 9.
4. Milan Fičura, Jiří Witzany: Determinants of NMD Pass-Through Rates in Eurozone Countries.
5. Sophio Togonidze, Evžen Kočenda: Macroeconomic Responses of Emerging Market Economies to Oil Price Shocks: An Analysis by Region and Resource Profile.
6. Jakub Drahokoupil: Application of the XGBoost algorithm and Bayesian optimization for the Bitcoin price prediction during the COVID-19 period.

All papers can be downloaded at: wp.ffu.vse.cz

Contact e-mail: ffawp@vse.cz

Faculty of Finance and Accounting, Prague University of Economics and Business, 2022

Winston Churchill Sq. 1938/4, CZ-13067 Prague 3, Czech Republic, ffu.vse.cz