Erasmus University Rotterdam
Erasmus School of Economics
Master Thesis - Econometrics and Management Science - Quantitative Finance

Trading in Ethereum Perpetual: Predicting High and Low prices of next hour using Machine Learning Techniques

Sunny Hsieh (604455)

Supervisor:	Alberto Quani
Second assessor:	Anastasija Tetereva
Date final version:	8th July 2023

Abstract

This thesis paper presents a comprehensive comparative analysis of various machine learning methods for predicting cryptocurrency prices. Specifically, we focus on the price prediction of Ethereum Perpetual, and we find that Support Vector Regression (SVR) outperforms other machine learning techniques in accurately predicting both high and low prices. Moreover, we demonstrate the suprisingly notable accuracy of linear models, compared to the one of advanced nonlinear models. However, we highlight the crucial importance of considering additional evaluation metrics when applying these models in real-world scenarios. Furthermore, we demonstrate that dynamic trading strategies using machine learning models, which adjust position sizing based on the predictions, can considerably outperform static strategies. This finding emphasizes the potential of machine learning in enhancing investment decision-making processes and generating profitable trading strategies. Ultimately, the research emphasizes the importance of conducting thorough experimentation and evaluation of various machine learning methods before implementing a solution in the fintech industry. This approach advances our empirical understanding of financial derivatives and contributes to the improvement of investment practices.

Keywords

Machine Learning, Prediction, Ethereum, Perpetual contracts, Financial Derivatives, Linear Regression, Support Vector Regression, Random Forest, LSTM, (Deep) Neural Networks, Model AVeraging, Fintech

Acknowledgements

I would like to express my sincere gratitude and appreciation to the following individuals for their invaluable contributions to the completion of this thesis:

First and foremost, I extend my heartfelt thanks to my first supervisor, Alberto Quani. His unwavering guidance, support, and encouragement throughout the research process have been instrumental in shaping this work. His expertise and insightful feedback have significantly enriched the quality of this thesis.

I would also like to extend my gratitude to Yun Song, my company supervisor. His support, valuable insights, and practical guidance have played a crucial role in the successful execution of this research project.

Lastly, I would like to acknowledge and thank all the individuals who have contributed to this research in any way. Your assistance, input, and support, whether big or small, have been immensely valuable and deeply appreciated.

I am truly grateful for the opportunity to work with such dedicated individuals who have helped me in this academic journey.

Contents

Chapter 1

Introduction

1.1 Background and Literature

The use of cryptocurrencies has rapidly expanded in recent years, with Ethereum being one of the most widely traded digital assets. As the cryptocurrency market continues to grow, there is an increasing need for accurate price predictions and trading strategies to improve investment outcomes.

The cryptocurrency market is characterized by high volatility and unpredictability, which makes it difficult for traders and investors to make informed decisions. Predicting the high and low price ranges of cryptocurrencies can provide valuable insights into the potential risk and return of a financial asset, helping market participants to manage risk and optimize their portfolios.

Machine learning (ML) techniques have been widely applied in the financial industry to forecast asset prices and develop profitable trading strategies.

A comprehensive explanation and argument for the use of ML in Finance is well explained and argued by Israel, Kelly and Moskowitz (2020) in their paper "Can machines learn Finance?". They have shown that machine learning has promising applications in factor investing, risk management, trading cost management, and return prediction.

According to research conducted by Akyildirim, Goncu and Sensoy (2021), At daily or minute-level frequencies, Machine Learning classification algorithms demonstrate an average predictive accuracy ranging from 55% to 65% for the top 12 cryptocurrencies based on market capitalization. Among the algorithms tested, support vector machines consistently demonstrated the highest level of predictive accuracy, outperforming logistic regression, artificial neural networks, and random forest classification algorithms.

Livieris, Pintelas, Stavroyiannis and Pintelas (2020) applies long short-term memory (LSTM) models to predict cryptocurrency prices and reports promising results. Also Lahmiri, Bekiros and Salvi (2018) signify the existence of long-range memory in Bitcoin market volatility. This means that past volatility levels can have a persistent effect on future volatility levels. In other words, the volatility of Bitcoin does not quickly revert to a mean value but rather exhibits a memory of past fluctuations. Furthermore, the findings of Yaya, Vo, Ogbonna and Adewuyi (2020) revealed that various cryptocurrencies exhibit cointegration in both the stationary and non-stationary aspects of their volatility. Combining these studies lead to the use of LSTM

models in forecasting price ranges for Ethereum derivatives, as proposed in our research, is a logical extension of the existing literature and has the potential to contribute to the advancement of the emerging field of cryptocurrency trading.

While there have been several studies on cryptocurrency price prediction, few have focused on predicting price ranges, and even fewer have used machine learning techniques to make these predictions. Therefore, in this study, we selected machine learning models based on previous literature that have shown success in predicting cryptocurrency prices through either classification or regression techniques. These various ML-based models are Random Forest, Support Vector Machine and LSTM.

We explore the potential of these models to predict price ranges for Ethereum derivatives and enhance trading performance. Specifically, we aim to develop a model that can accurately predict price ranges for Ethereum derivatives and outperform the MA benchmark.

By applying these models to predicting price ranges, we aim to contribute to the emerging field of cryptocurrency trading and provide insights that can assist investors and traders in making more informed decisions, since cryptocurrencies are known for their high volatility, which makes it difficult for traders and investors to make informed decisions. Predicting price ranges can provide valuable insights into the potential risk and return of a financial asset, helping market participants to manage risk and optimize their portfolios.

As the usage of cryptocurrencies continues to grow, the ability to predict price ranges becomes increasingly important for merchants and consumers to determine the fair value of goods and services. 300+ million people around the world use/own cryptocurrencies in 2022, the global crypto market cap is $1.06 trillion as of August 1, 2022 and approximately 15174 businesses worldwide accept Bitcoin, with around 2,300 of those businesses operating in the US shown by Zippia (2022).

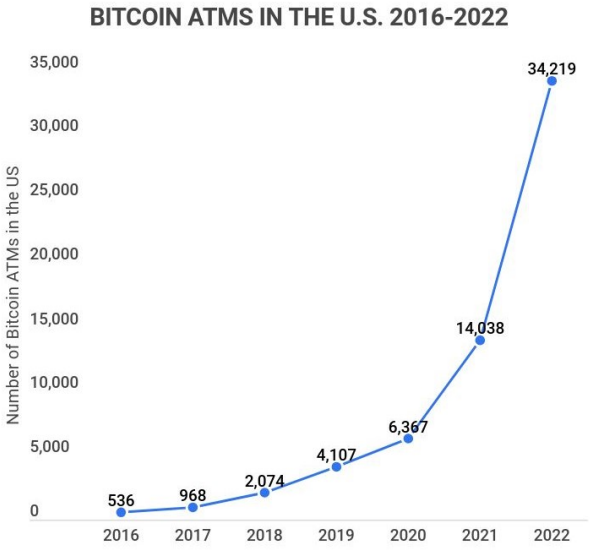


Figure 1.1: Bitcoin ATM

In figure 1.1 we see an exponential growing amount of ATMs in the past few years in the U.S..

Finally, the cryptocurrency market is highly competitive, and accurate price range predic-

tions can give traders and investors a competitive edge by allowing them to make informed decisions and execute trades with greater precision. Therefore, developing accurate price range prediction models can benefit not only individual traders and investors but also financial institutions, merchants, and regulators.

Our research builds upon the growing body of literature on the application of ML in finance and contributes to the emerging field of cryptocurrency trading. Through our investigation, we hope to provide insights that can assist investors and traders in making more informed decisions and ultimately improve their trading outcomes.

1.2 Research Goals

The purpose of this study is to enhance derivative trading in Ethereum by predicting price ranges through the use of machine learning techniques. To achieve this goal, we address several research questions. Firstly, we explore the machine learning algorithms that perform best in predicting Ethereum price ranges. Secondly, we identify the most significant predictor variables for Ethereum price ranges, using different machine learning techniques. Additionally, we compare the predictive accuracy and economic gains of machine learning methods with those of traditional techniques in the context of Ethereum derivative trading. Finally, we investigate how incorporating machine learning techniques to predict price ranges can affect the performance of derivative trading in Ethereum. Specifically, we examine whether a dynamic trading strategy, which utilizes the machine learning model's price range predictions, can outperform a static trading strategy and the benchmark, thereby demonstrating the potential of machine learning in improving derivative trading.

Chapter 2

Data

2.1 Ethereum Perpetual Dataset

2.1.1 Origin of the data

The study utilizes data obtained from Deribit, a cryptocurrency derivatives exchange platform that offers trading in futures, perpetual swaps, and options. The data was accessed using an API key provided by Panda Analytics.

The Ethereum Perpetual price data and associated market variables, such as volume, open interest, and bid-ask spreads along with several associated market variables such as volume, open interest, and bid-ask spreads were collected at an one-minute interval. The dataset encompasses the period starting from March 2019, which coincides with Deribit's establishment, and extending up to February 2023. This timeframe covers a period marked by significant market volatility, notably during the crypto bull run observed in 2021, followed by a subsequent market correction., which covers a period of significant market volatility, particularly during the crypto bull run in 2021 and the subsequent market correction.

2.1.2 Data quality and size

The length of the dataset is also worth noting, as it contains 2144148 observations, which is a significant amount of data. This large dataset can be both a strength and a weakness, as it allows for a more comprehensive analysis and more accurate predictions. However, it also poses some challenges in terms of data processing, storage, and analysis. Therefore, proper data cleaning, filtering, and preprocessing techniques are applied to ensure the dataset's quality and validity. The data quality and reliability were confirmed by conducting robustness checks with other trading platforms such as Yahoo Finance.

2.1.3 Target variables

To predict the price range of Ethereum perpetual contracts, we need target variables that represent the high and low prices of the next n timeframes. In this study, the target variables will be extracted by taking the highest and lowest price of the next 60 minutes.

The reason for choosing the next 60 minutes as the timeframe to extract the target variables is that it provides a reasonable tradeoff between accuracy and practicality. Longer timeframes,

such as several hours or days, can result in more accurate predictions, but they are less practical for short-term trading decisions. On the other hand, shorter timeframes, such as one minute or five minutes, can be more practical but may be too noisy and less accurate. Figure 2.4 showcases the erratic fluctuance of minute based data for an hour. By choosing a timeframe of 60 minutes, we can capture enough price movement to make accurate predictions while still providing useful information for short-term trading decisions.

Additionally, the company that conducted this study supports this choice of target variables as it aligns with their goal of enhancing derivative trading in Ethereum. If the results of the model using the 60-minute timeframe as target variables are promising, the company may consider exploring shorter timeframes to predict the price range of Ethereum perpetual contracts using machine learning. This would involve extracting the high and low prices of the next few minutes instead of the next hour. However, this decision will depend on the accuracy and reliability of the model's predictions for the 60-minute timeframe.

Figure 2.1 showcases a compelling visualization of the price movements of the Ethereum Perpetual, providing valuable insights for traders and investors alike.

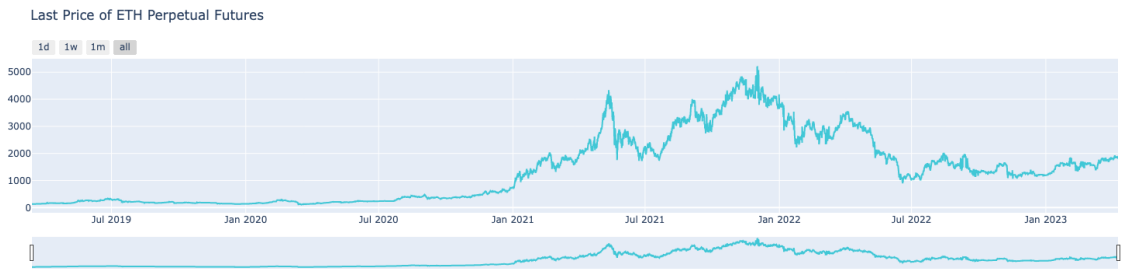


Figure 2.1: Hourly price of the Ethereum Perpetual

Meanwhile, Figure 2.2 presents a striking depiction of the high and low target variables, shedding light on the volatility and potential price range of this digital asset. The figure shows that most of the time, the high and low move in a similar direction, reflecting the overall trend of the market. However, there are moments when the two variables diverge, indicating high volatility and potentially profitable trading opportunities.

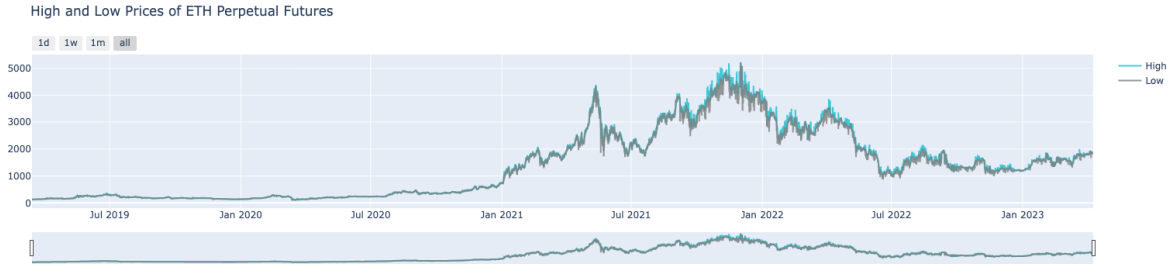


Figure 2.2: Hourly High and Low of the Ethereum Perpetual

To take a closer look at the relationship between high and low, we can refer to Figure 2.3,

which shows the hourly range of the Ethereum Perpetual. At the beginning of the time series, the range was relatively small, indicating a relatively stable market. However, as time passed, the range started to fluctuate more, reflecting the increasing volatility of the market.

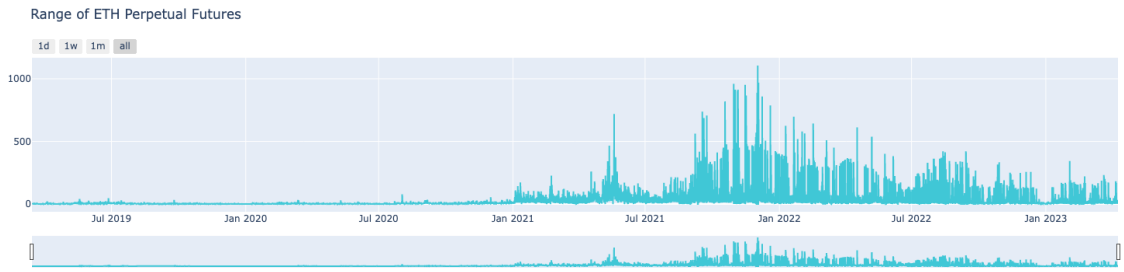


Figure 2.3: Hourly Range of the Ethereum Perpetual

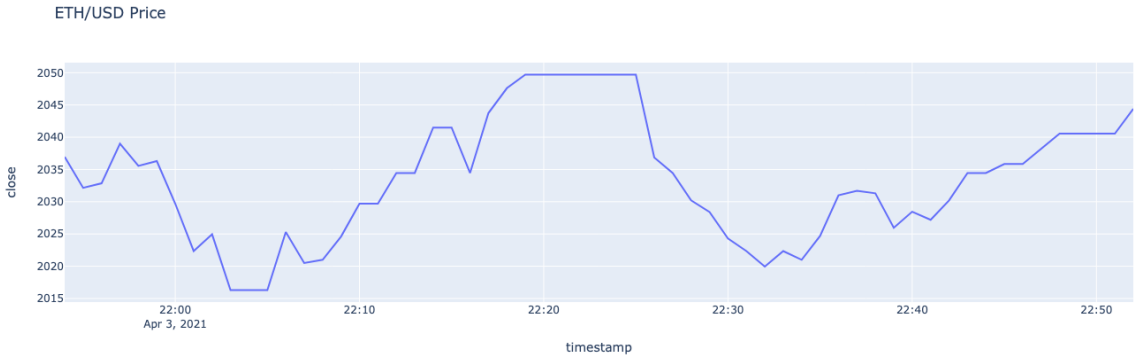


Figure 2.4: Price of Ethereum Perpetual over an interval of 60 minutes

In Appendix A, you can find visualizations of the other key features in our dataset, including the hourly return percentage, and volatility. These plots can provide additional insights into the behavior of these features over time and can help us understand how they may relate to the target variables we aim to predict.

Overall, the availability of high-quality data from Deribit and Panda Analytics, combined with the application of machine learning techniques, offers a promising approach to optimize derivative trading in Ethereum.

2.1.4 Features

The perpetual data comprises several valuable features that can help traders make informed decisions, including the high, low, range, volatility, returns in percentage and price per minute. We refer to this set of data as group 1, and it provides insightful information about the Ethereum market's price movements.

In addition to group 1, there are several other key features that can provide valuable insights, such as funding rates, best ask price, best bid price, current funding, funding 8 hours, index price, interest value, last price, mark price, max price, min price, open interest, and settlement price. We label this set of data as group 2.

However, there is a discrepancy between the time period for which group 1 and group 2 data were collected. The company started collecting group 2 features since the inception of this project in January 2023, while the price data for Ethereum perpetual contracts began on March 13, 2019. Consequently, group 2 data can only be collected and stored in real-time and cannot be retrieved historically through the Deribit platform, creating a temporal mismatch.

Due to the discrepancy in the time period for which group 1 and group 2 data are collected, there is a significant difference in the amount of data available for each group. Since group 1 data begins on March 13, 2019, it contains more data compared to group 2 data, which is only collected since January 2023.

Furthermore, it is worth noting that the data from March 2019 to January 2021 might not be very useful in predicting future trends in the Ethereum Perpetual market. This is due to the low volatility observed during this period, as depicted in the volatility graph in Appendix A. The range of Ethereum Perpetual during this period was also relatively small, as shown in Figure 2.3. Therefore, it may not provide much insight into the current market conditions and price movements.

As a result, we have decided to focus our analysis on the data from January 2021 onwards, which corresponds to a period of high volatility and significant price movements. This period provides a more accurate representation of the current market conditions and allows us to generate more meaningful insights and predictions.

Overall, the group 1 data provides a rich dataset for analyzing the Ethereum Perpetual market's price movements, and our research focuses on utilizing this data to predict price ranges. However, the inclusion of group 2 data could offer additional insights and avenues for further research, particularly for analyzing the Ethereum Perpetual market's liquidity and depth.

Additionally, since the primary goal of our research is to predict price ranges in Ethereum derivative trading, we have decided to only use group 1 data. This is because group 1 data includes the data that can be transformed to the target variables, which are required to predict price range.

Nevertheless, the group 1 data starting from January 2021 alone still offers a comprehensive dataset for analyzing Ethereum's price movements and predicting future trends, making it a valuable resource for traders.

2.1.5 Feature Selection

Since our data is a time series, we will include lagged features to capture the autocorrelation in the data. We use the autocorrelation function (ACF) and partial autocorrelation function (PACF) to determine how many lags we should include. The ACF measures the correlation between a time series and its lags, while the PACF measures the correlation between a time series and its lags after accounting for the correlation at shorter lags.

In addition to using ACF and PACF to determine the number of lags for the high and low features, we also apply the same technique to the other features, including last price, volatility, range, and returns. The reason for doing this is that these features can also contain important information for predicting future high and low values.

After applying the ACF and PACF analysis to determine the number of lags for the high

and low features, it is also important to assess the significance of these lags. This can be done by performing the t-test. The t-test can be used to test the null hypothesis that the coefficient of a particular feature is zero, indicating that the feature has no significant impact on the prediction. The alternative hypothesis would be that the coefficient is non-zero, indicating that the feature is indeed significant.

By using ACF and PACF to analyze these features, we can identify the most significant lags and include them in our predictive model. This helps ensure that we are utilizing all available information to make accurate predictions. Additionally, it allows us to potentially identify any patterns or correlations between these features and the high/low values, which could further enhance the predictive power of our model.

Therefore, by applying ACF and PACF to all features, we can create a more comprehensive and accurate predictive model for the next hour's high and low values.

We have generated ACF and PACF plots for each of our six features: last price, high, low, range, volatility and return percentage. Based on the plots, we have identified the number of lags to include for each feature. Appendix B provides a more detailed explanation of how we have used the ACF and PACF plots and significant tests to determine the lag order for each feature.

In summary, we have included the following lags for each feature:

- Last price: 3 lags
- High: 4 lags
- Low: 4 lags
- Range: 2 lags
- Return percentage: 1 lag
- Volatility: 4 lags

In order to investigate the correlation between the features, we compute the correlation matrix and in Figure 2.5 we see how many of the features in within a certain range. There were no features with negative correlations between -0.1 and -1. We do see that around 70% of the features has a lower correlation than 0.5. Also we see that almost 30% has a correlation higher than 0.9 which are the lagged variables.

These results indicate that there is generally a low to moderate correlation between the features, which is desirable when building a machine learning model as highly correlated features can lead to overfitting and decreased model performance. However, it is important to note that correlation is just one aspect of feature selection, and other methods such as feature importance and regularization should also be considered in the model building process. For more detailed information about the correlation between the features, refer to Appendix C where the correlation matrix of the features is shown. The matrix reveals that some of the features have high correlation values, especially the lagged variables of a feature. Therefore, we need to be careful when selecting features for the models in order to avoid multicollinearity.

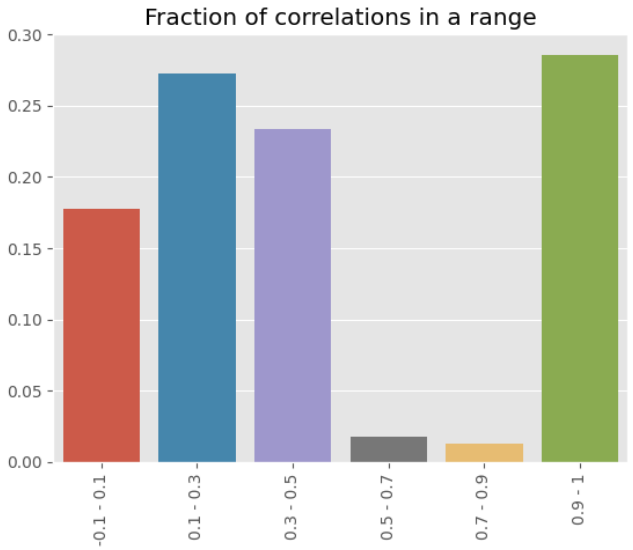


Figure 2.5: Fraction of features with a correlation in a range

In our analysis, we choose to include a total of 60 explanatory variables related to the high, low, range, volatility, price, and return of an asset. These variables are measured at hourly intervals for the current time period (t) and for the 12 previous hours (t-1, t-2, t-3, ..., t-12). This provides a rich set of features that can be used to train our machine learning models.

Chapter 3

Methodology

3.1 Predicting High and Low prices

3.1.1 Models

The methodology for this research involves the prediction of the high and low prices of Ethereum Perpetual using both traditional and machine learning techniques. In this study, the high price refers to the maximum value reached within a specific time unit, while the low price represents the minimum value within the same time unit. It is important to consistently reiterate the time unit over which these prices are measured to ensure accurate analysis and predictions. Regarding the time-unit throughtout this reserach we use a time-unit of 60 minutes. For the models, we use both traditional and machine learning techniques for predicting the high and low prices. Linear Regression (LR) will serve as a benchmark for the machine learning models. The LR model provides a linear prediction based on the historical high and low.

For the non-linear models we have:

- The Random Forest model.
- Support Vector Machine.
- LSTM, which is a specialized type of Neural Network that is designed to handle the long-term dependencies that are common in time-series data.

Random Forest (RF)

The Random Forest model, which is an ensemble learning method that uses decision trees to make predictions. It is a popular choice for financial forecasting due to its ability to handle non-linear and high-dimensional data, unlike traditional linear models, this approach is fully non-parametric.

The process of building a tree involves a series of iterations. At each iteration, the data is divided into different bins based on one of the predictor variables. This categorizes observations into groups that share similar behavior, resulting in rectangular divisions in the predictor space. The unknown function $g(\cdot)$ is then represented by the resulting tree is estimated by the average value of the outcome within each partition.

The equation 3.1 depicts a function that is piece wise constant and determines the average outcome variable value within each partition of the predictor space. In this equation, $z_{i,t}$ denotes the value of the ith predictor variable at time t, while $C_k(L)$ is the kth partition of the predictor space, defined by the Lth tree in the ensemble. The contribution of the kth partition to the final predicted value of the outcome variable is represented by the parameter θ_k.

$$g\left(z_{i,t};\theta,K,L\right)=\sum_{k=1}^{K}\theta_k\mathbf{1}_{\{z_{i,t}\in C_k(L)\}} \tag{3.1}$$

In practical applications, the algorithm used to estimate the unknown function g(·) initially builds a large number of trees, typically in the hundreds or thousands. The data is recursively split into partitions based on the values of the predictor variables and each tree predicts the outcome variable by taking the average value of the response variable within each partition.

The process of combining the predictions of multiple trees into a single estimate involves using an ensemble method such as boosted regression trees or random forests. Boosted regression trees build trees sequentially, where each subsequent tree is trained to correct the errors of the previous tree. On the other hand, random forests build trees independently, where each tree is trained on a random subset of the data and predictor variables.

The ensemble method produces the final prediction by taking the average of all the trees' predictions, with each tree's contribution weighted by its performance on a validation set. By utilizing this approach, the model can capture intricate relationships between predictor variables, resulting in improved accuracy and resilience compared to traditional linear models.

Support Vector Machine (SVM)

SVM is a powerful and widely used machine learning algorithm for classification and regression and works by finding the best possible boundary that separates data points of different classes or predicts values for a given set of data. It is particularly useful in dealing with high dimensional data with complex relationships between variables. Examples for where SVM is often used are classification problems, outlier detection, and regression.

SVMs are widely recognized for their effectiveness in solving classification problems, as highlighted in a study by Huang, Nakamori and Wang (2005) which investigated the predictability of the direction of the movement of NIKKEI 225 index using SVM as a classifier. In Gu, Kelly and Xiu (2020), the authors justified not incorporating SVM in their research.

In contrast to the general perception of SVMs being applicable mainly to classification problems, Rahman and Rijal (2012) proposes their application in limited data environments. While their research focuses on classification problems, we will extend the application of SVMs to regression and employ support vector regression (SVR). Notably, previous studies such as Cao and Tay (2001) and Kim (2003) have explored the use of SVR in financial time series forecasting and compared it with back-propagating neural networks, nut did not provide an detailed explanation of SVR. Therefore, in our study we will provide a more detailed explanation of our SVR implementation.

The principle of SVR is based on the same idea as SVM's. However, unlike other regression models that aim to minimize the error between the predicted and actual values, SVR seeks to

find a hyperplane that has the maximum number of points within a certain threshold. This threshold is defined as the distance between the hyperplane and the boundary line. One of the main advantages of using SVR instead of linear regression is its capability to model non-linear relationships. Our SVR implementation follows the scikit-learn library's SVR implementation (Pedregosa et al., 2011). Their SVR solves the following optimization problem:

$$\begin{aligned} \min_{w,b,\zeta,\zeta^*} \tfrac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*) \\ \text{subject to } y_i - w^T \phi(z_i) - b \leq \varepsilon + \zeta_i, \\ w^T \phi(z_i) + b - y_i \leq \varepsilon + \zeta_i^*, \\ \zeta_i, \zeta_i^* \geq 0, i = 1, \ldots, n. \end{aligned} \tag{3.2}$$

where:

w and b are the weight vector and bias term, respectively,

ζ_i and ζ_i^* are the slack variables,

$\phi(z_i)$ represents the feature mapping of input z_i,

y_i is the target variable,

C is the regularization parameter,

ε is the epsilon-insensitive loss function parameter.

This formulation allows SVR to find the optimal hyperplane that maximizes the margin while allowing for deviations (slack variables) within a specified tolerance ε.

Samples whose predicted target value deviates from their true target value by at least a threshold ϵ are penalized in SVR. The penalty is represented by the slack variable ζ_i for samples whose predicted value is above the ϵ tube, and ζ_i^* for samples whose predicted value is below the ϵ tube.

By using the Lagrangian multiplier, equation 3.2 can be solved as an optimization problem subject to constraints. The resulting dual problem is:

$$\begin{aligned} \min_{\alpha,\alpha^*} \tfrac{1}{2} (\alpha - \alpha^*)^T Q (\alpha - \alpha^*) + \varepsilon e^T (\alpha + \alpha^*) - y^T (\alpha - \alpha^*) \\ \text{subject to } e^T (\alpha - \alpha^*) = 0 \\ 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, \ldots, n. \end{aligned} \tag{3.3}$$

where:

α, α^* are the Lagrange multipliers,

Q is the positive semi-definite kernel matrix,

ε is the epsilon-insensitive loss function parameter,

e is a vector of ones,

y is the vector of target variable values,

C is the regularization parameter,

α_i, α_i^* are the Lagrange multipliers for the constraints,

n is the number of samples.

The prediction can be obtained by solving the optimization problem described in Equation 3.2, using the Lagrangian multiplier. In the dual problem given by Equation 3.3, the vector e consists of ones and Q is a positive semi-definite matrix with a corresponding kernel $Q_{ij} = K(z_i, z_j) = \phi(z_i)^T \phi(z_j)$. Here, ϕ maps the training vectors z_i and z_j into a higher dimensional space. The solution to the optimization problem yields the following prediction:

$$E_t(r_{i,t+1}) = \sum_{i \in SV} (\alpha_i - \alpha_i^*) K(z_i, z) + b. \tag{3.4}$$

where:

$E_t(r_{i,t+1})$ represents the predicted error of the target variable $r_{i,t+1}$ at time t,
α_i, α_i^* are the Lagrange multipliers for the constraints,
$K(z_i, z)$ is the kernel function that computes the similarity between input vectors z_i and z,
b is the bias term.

Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) architecture that is commonly used in time series analysis and prediction. LSTM networks are designed to deal with the problem of vanishing gradients in traditional RNNs, which occur when the gradient of the loss function becomes very small during backpropagation, making it difficult for the network to learn from previous timesteps.

LSTM networks consist of multiple memory cells that can store information over a long period of time. Each memory cell has three gates - input, output, and forget - that control the flow of information into and out of the cell. The input gate determines how much new information is added to the cell, the output gate determines how much information is output from the cell, and the forget gate determines how much information is retained in the cell.

The architecture of a LSTM network is typically defined by the number of memory cells, or hidden units, in each layer. The input to the network is typically a sequence of past observations, and the output is a prediction for the next observation in the sequence.

The LSTM model is implemented using the Keras library in Python, which provides a high-level interface for building and training neural networks. The model will be trained using a variant of the stochastic gradient descent optimization algorithm known as Adam, which is commonly used in deep learning applications.

For the other hyperparameters, a more detailed explanation can be found in Appendix D.

Model Averaging

Model averaging is a technique commonly used in various fields, including economics, to improve the predictive performance of statistical models. It involves combining predictions from multiple individual models to create a more accurate and robust final prediction. The basic idea behind model averaging is to reduce the variance of predictions by taking into account different sources of uncertainty.

One of the primary advantages of model averaging is its ability to reduce the impact of model-specific biases and errors. By averaging the predictions of multiple models, the individual weaknesses and idiosyncrasies of each model can be mitigated. This approach helps to capture a broader range of information and provides a more balanced and reliable prediction.

In the context of economics, Moral-Benito ((2015)) demonstrated the effectiveness of model averaging techniques. Their research showed that by combining the predictions of the best individual models, significant improvements in predictive accuracy can be achieved.

In our analysis, we will employ model averaging by considering the best two individual models. By combining the predictions of these models, we aim to leverage their strengths and mitigate their weaknesses. This approach can help us obtain a more robust and accurate prediction, as it takes into account different perspectives and sources of information.

Model averaging can be particularly beneficial in situations where individual models exhibit high variance or have limited data. By combining multiple models, we can effectively smooth out the variability in predictions and obtain more stable estimates. However, it is important to note that model averaging may also introduce additional computational complexity and require careful consideration of model selection and weighting strategies.

3.1.2 Traning Methodology

Validation-based Hyperparameter Optimization

We adopt the training approach proposed by Gu et al. (2020) where the dataset is split into training and testing subsets with a 70:30 ratio, respectively. The training subset is further divided into an 80:20 ratio for actual training and validation data, respectively, to prevent overfitting. The validation subset, which constitutes 20% of the dataset, is used for hyperparameter tuning.

The validation objective is optimized using the coefficient of determination (R^2), which measures the proportion of variance in the target variable that is explained by the model. The R^2 value ranges between 0 and 1, where a higher value indicates a better fit of the model to the data.

The reason for choosing the R^2 as our objective function is because it provides a measure of the model's goodness-of-fit and allows for a comparison of different models' performance. Additionally, Gu et al. (2020) used the same objective function for hypertuning the models.

The remaining 30% of the dataset constitutes the testing subset, which is truly out-of-sample and is used to evaluate the model's predictive performance. The testing subset is not used for either estimation or tuning and provides a realistic assessment of the model's performance.

Appendix D contains a summary of the hyperparameter tuning procedures used for each model.

Predicting out of sample

For predicting out of sample, we utilize an expanding window approach, with retraining being carried out after each month. This allows the models to be continuously updated with new data and adapt to changing market conditions. In contrast to Gu et al. (2020), we deviate by implementing rehypertuning, which involves periodic re-evaluation and tuning of the hyper-

parameters to ensure that the models remain optimal over time. This approach is implemented on a quarterly basis to ensure that the models are consistently performing optimally with the most up-to-date information, where the same hyperparameter grid is utilized as in Appendix D.

Constraints

One of the primary and obvious constraints in our methodology is the requirement that the high price is always greater than the low price. This is a fundamental constraint in financial forecasting as the high and low prices are the two extremes in a given time period. Therefore, it is imperative that the high price is always higher than the low price, as it reflects the actual price movement.

To address this constraint, we incorporate a check in our model to ensure that the predicted high price is always greater than the predicted low price. This is done through the use of constraints by overwriting the classes of the machine learning we are using.

Incorporating this constraint in our methodology ensures that the predicted prices are consistent with the behavior of the financial markets and helps to maintain the credibility of our model. However, it may result in some loss of predictive accuracy, as the constraint limits the range of possible predictions. Therefore, we closely monitor the impact of this constraint on our predictive performance and adjust our methodology as necessary.

Hardware and packages

To expedite the training process, we utilize graphics processing unit (GPU) acceleration during the training of the models. This is because GPUs can perform complex computations in parallel, which significantly speeds up the training time of deep learning models. By using GPUs, we can reduce the time needed to train the models and thus increase the efficiency of our research.

3.1.3 Model Evaluation

Mean Squared Error (MSE)

We evaluate the performance of our models using the mean squared error (MSE) as the evaluation metric. This is in line with previous research that has also used MSE to evaluate models for financial time series prediction in derivatives (Almeida et al., 2021). Equation 3.5 shows the formula of MSE.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{actual}_i - \text{predicted}_i)^2 \tag{3.5}$$

Mean Absolute Error (MAE)

Furthermore, we also incorporate the mean absolute error (MAE) as an additional evaluation metric. Equation 3.6 shows the formula of MAE. MAE measures the average absolute difference between the predicted values and actual values, providing a better understanding of the magnitude of errors. It is worth noting that MAE is considered to be more robust in terms of breakdown point and influence function compared to mean squared error. This means that

MAE is less affected by outliers and extreme values, making it a more robust metric for evaluating model performance in the presence of such data points. Therefore, by using both MSE and MAE, we can assess the performance of our models from different perspectives, taking into account both the overall accuracy and the robustness to outliers.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\text{actual}_i - \text{predicted}_i| \tag{3.6}$$

As part of the collaborative decision-making process with the company, we have chosen to use the Mean Absolute Error (MAE) as the evaluation metric. This decision aligns with both our thinking and the company's preference for a straightforward and easily understandable assessment of the model's performance. By selecting MAE, we aim to provide a measure that accurately reflects the deviation between predicted and actual values, highlighting the model's performance in a manner that resonates with the company's objectives. This may be important for decision-making purposes, as stakeholders may use the model's output to inform important business decisions. Additionally, using MAE alongside MSE can provide a more comprehensive understanding of the model's performance, as each metric captures different aspects of the prediction error.

Conservative Range Prediction Metric (CRPM)

At last, a custom-designed metric is developed to evaluate the performance of the predictive models. The custom metric takes into account the range between the predicted high and low prices and the actual high and low prices for each time period.

With the company's objective of prioritizing conservative predictions in mind, a custom metric has been designed to favor models that make conservative predictions over those that make risky predictions. The custom metric penalizes the models that predict wider ranges than the actual ranges and more importantly, outside the boundaries of the actual high and low prices.

For instance, if the actual high price for a given time period is 20 and the actual low price is 10, and the model predicts a high of 18 and a low of 12, a penalization is taken into account. Similarly, if the model predicts a high of 22 and a low of 8, the model is also penalized since the actual range is wider than the predicted range.

We have named this evaluation metric the Conservative Range Prediction Metric (CRPM), reflecting the company's objective of favoring conservative predictions.

The CRPM is used in combination with other evaluation metrics to provide a comprehensive assessment of the performance of the predictive models. This allows for a more nuanced understanding of the strengths and weaknesses of each model, helping to identify the models that are best suited to the company's needs.

Equation 3.7 calculates the proportion of correct predictions for the high prices, where the indicator function evaluates to 1 if the predicted high price is lower than the actual high price for a given time period, and 0 otherwise. Similarly, Equation 3.8 calculates the proportion of correct predictions for the low prices, considering whether the predicted low price is higher than the actual low price.

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(\text{pred\_high}[i] < \text{actual\_high}[i]) \tag{3.7}$$

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(\text{pred\_low}[i] > \text{actual\_low}[i]) \tag{3.8}$$

Additionally, it is important to note that in the context of the CRPM, a higher score indicates better performance. A higher proportion in Equation 3.7 and Equation 3.8 implies that the predictive models are making more accurate and conservative predictions, aligning with the company's objective. Therefore, when evaluating the models using the CRPM, a higher score is desirable as it indicates a stronger adherence to conservative prediction principles and suggests a higher level of accuracy in forecasting the price ranges.

Diebold and Mariano test

We employ the Diebold and Mariano (1995) test to determine if the predictions generated from our method are significantly different. To account for error dependence, we use an adjusted version of the test, as recommended by Gu et al. (2020), who acknowledged the potential for error correlation in the financial instruments. Given that our analysis focuses on Ethereum perpetual, we adopt the same approach to address the same risk.

For each pair of models, the test statistic is calculated using the mean and Newey-West standard error of the time series, as follows: $DM_{12} = \bar{d}12/\hat{\sigma}\bar{d}12$. Here, $\bar{d}12$ and $\hat{\sigma}\bar{d}12$ represent the mean and standard error, respectively, of the time series $d_{12,t+1}$, which is defined as:

$$d_{12,t+1} = \frac{1}{n_{3,t+1}}\sum_{i=1}^{n_{3,t+1}}\left(\left(\widehat{ei}, t+1^{(1)}\right)^2 - \left(\widehat{ei}, t+1^{(2)}\right)^2\right), \tag{3.9}$$

where $\widehat{ei}, t^{(j)}$ represents the prediction error of method j on the Ethereum Pepertual i's high and low for time t, and $n3, t+1$ is the number of samples in the test set.

As suggested by (Gu et al., 2020), asymptotic normality is now more likely to hold. Therefore, it can be used to infer the significance of the predictions.

At last, we only employ the Diebold and Mariano test for the individual models and not the model averaging model. This decision is based on the specific focus and objectives of our analysis.

The Diebold and Mariano test is a statistical test commonly used to compare the forecasting accuracy of different models. It assesses whether one model significantly outperforms another in terms of predictive accuracy. By conducting this test for the individual models, we can gain insights into their individual strengths and weaknesses.

While model averaging has its advantages, including reduced variance and improved robustness, it is not specifically designed to be evaluated using the Diebold and Mariano test. Model averaging aims to combine the predictions of multiple models to achieve better overall performance, rather than directly comparing against individual models. As a result, applying the Diebold and Mariano test to the model averaging approach may not provide meaningful or interpretable results.

Furthermore, focusing on the Diebold and Mariano test for the individual models allows us to gain a deeper understanding of the specific characteristics and performance of each model. This knowledge can inform model selection decisions in future applications or provide insights into the drivers of forecasting accuracy for the individual models.

3.1.4 Feature Importance

When working with machine learning methods, it is crucial to understand which features have the most impact on the predicted outcomes. We utilize the univariate feature selection methodology following Jović, Brkić and Bogunović (2015) to determine feature importance. This involves fitting the model using each feature separately and evaluating the performance metric (e.g., MSE) to determine the importance of each feature. The features are then ranked based on their performance, and the top-ranked features are selected as the most important features.

Despite the important metric CRPM, looking at the CRPM is only useful when the MSE score is not too bad. Therefore, we choose to use MSE as our primary metric for identifying feature importance. If the MSE scores of different features are similar, it would be useful to consider CRPM as a secondary metric for identifying feature importance.

However, it is important to note that we do not perform feature importance analysis specifically for the model averaging approach. This is because the focus of model averaging is to combine the predictions of multiple models rather than assess the individual importance of features within each model.

The rationale behind this decision is that the feature importance obtained from analyzing individual models may not directly translate to the model averaging model. Since model averaging involves combining the predictions from multiple models, the importance of features within the individual models may vary when aggregated. Consequently, evaluating feature importance for the model averaging model might not provide meaningful insights or accurately represent the relative importance of features.

3.2 Trading Strategy

In order to assess the practical value of the predictive models, we expand the company's static trading strategy using the predicted price ranges. The previous static strategy is designed to maintain a non-directional position by always having a buy and sell order in the order book. By utilizing the predicted high and low price ranges generated from the machine learning models, the strategy can be dynamically adjusted to improve the placement of the orders.

3.2.1 Trading Strategy Evaluation

The performance of the trading strategy is evaluated a virtual environment. This allows for a controlled testing environment where the performance of the dynamic trading strategy can be evaluated without risking real capital. However, the real market live simulation environment is still in production by the company, so we test on historical market conditions.

The evaluation of the trading strategy takes into consideration various performance metrics such as the total profit before and after fee and Sharpe ratio as performance measures. The

evaluation provides a comprehensive analysis of the trading strategy, which helps to determine its viability and effectiveness in generating returns. Based on the results, future adjustments can be made by the company to optimize the performance of the trading strategy, and the evaluation process is repeated until the desired performance is achieved or the achieved performance can no longer be improved.

Moreover, the assessment of the strategy plays a pivotal role in gauging the practical significance of the predictive models developed in this research. It also offers valuable insights into the models' effectiveness in generating profits and help identify the top-performing models in various market scenarios. Ultimately, the evaluation results inform the company's decision on whether to integrate the predictive models into its trading operations. Of course, before implementing the predictive models and the dynamic trading strategy in real-time trading operations, a careful testing process should be conducted to verify their effectiveness and reliability. The testing process should consider a variety of market scenarios, including both normal and abnormal conditions, and evaluate the performance of the models and the trading strategy over an extended period.

Chapter 4

Results

4.1 Model evaluation on predicting the high and low

In this section, we present the results of our model evaluation on predicting the high and low values of our target variable. We use three standard evaluation metrics, first one is Mean Squared Error (MSE) and we apply the Diebold-Mariano test to examine whether any of our models perform significantly better than others. Next we use Mean Absolute Error (MAE) and our own designed metric called Conservative Range Prediction Metric (CRPM) as evaluation metrics. Through this evaluation, we aim to identify the best-performing model for our prediction task.

	LR	RF	SVR	LSTM	Model Averaging (LR + SVR)
Mean Squared Error	1241.81	1245.23	1204.44	2037.75	1243.63
Mean Absolute Error	14.67	17.43	11.58	22.31	16.49
CRPM	0.22	0.22	0.46	0.33	0.29

Table 4.1: Evaluation Metrics Comparison for predicting the next hour Low

	LR	RF	SVR	LSTM	Model Averaging (LR + SVR)
MSE score	1170.13	1199.81	1165.25	1354.47	1183.77
Absolute error	14.00	17.58	11.46	19.17	16.03
CRPM	0.22	0.20	0.43	0.36	0.27

Table 4.2: Evaluation metrics Comparison for predicting the next hour High

4.1.1 Mean Squared Error and Diebold Mariano test

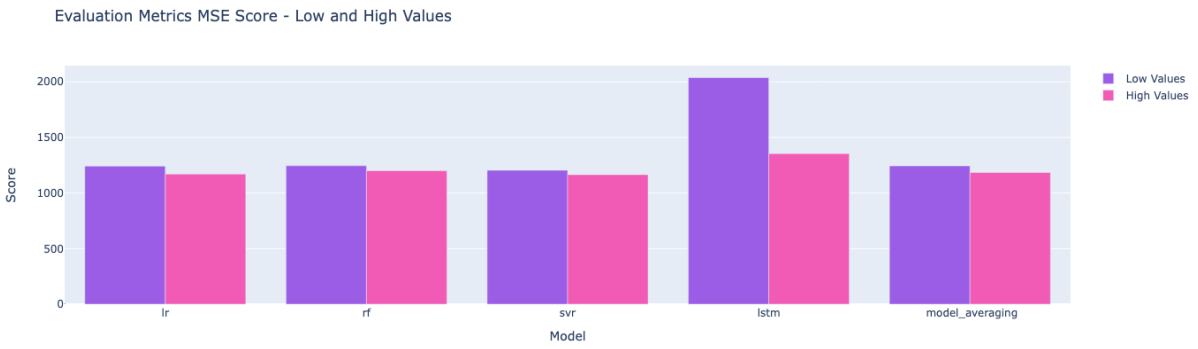


Figure 4.1: MSE comparison of the four models

MSE Low

Next to Table 4.1, Figure 4.1 shows the comparison of MSE scores among the four models for predicting the next hour low. We notice that the SVR model performs the best with the lowest MSE score of 1204.44, followed closely by the LR model with a score of 1241.81. On the other hand, the RF model and the LSTM model have higher MSE scores of 1245.23 and 2037.75, respectively.

One possible reason for the superior performance of the LR model over the non-linear models (RF, and LSTM) could be the overfitting of the non-linear models. Since the LR model is a linear model, it is less prone to overfitting than the non-linear models. However, there could be other reasons as well, such as the selection of hyperparameters or the quality of the training data.

Furthermore, the model averaging model, which is a combination of the best two performing models LR and SVR results in the third best performing model of having a MSE of 1243.63.

Diebold-Mariano Test Low

The Diebold Mariano test shows that the SVR model outperforms all other models, as indicated by the significant DM test statistics against all other models. This suggests that the SVR model has the best forecasting accuracy in predicting the low price of Ethereum Perpetual among all the models we evaluated.

Table 4.3: Diebold-Mariano tests between pairs of models (Low)

	LR	RF	SVR	LSTM
LR		1.57	**-32.57**	**11.94**
RF			**-38.71**	**7.45**
SVR				**43.59**

Note: The table reports the Diebold-Mariano (adjusted) test statistic between pairs of models. Positive values indicate that the model in the column outperforms the one in the row. Bold font indicates the value is significant at a 5% level.

Moreover, the LR model performs better than the RF model and significantly better than the LSTM model, as indicated by the positive and significant DM test statistics of 1.57 and 11.94, respectively. This suggests that the SVR model has better forecasting accuracy in predicting the low price of Ethereum Perpetual than the RF and LSTM models.

Furthermore, we observe that the RF model performs significantly better than the LSTM model, as indicated by the positive and significant DM test statistic of 7.45. This suggests that the RF model is better suited for predicting the low price of Ethereum Perpetual than the LSTM model.

MSE high

Table 4.2 and Figure 4.1 show the MSE score among the different models. We observe that the SVR model still performs the best with the lowest MSE score of 1165.25, followed again by the LR model with a score of 1170.13, which is a small difference. Regarding the model averaging model, it is performing again worse than LR and SVR but is performing better than RF and LSTM with a MSE of 1183.77. The RF model has a slightly higher MSE score of 1199.81, while the LSTM model has the highest MSE score of 1354.47.

It is worth noting again that LR has a superior performance of the LR model over the other models two non-linear models RF and LSTM in predicting the next hour high. Possible explanation could be the simplicity and interpretability of the linear model. However, similar to the previous case, other factors such as the selection of hyperparameters or the quality of training data could also have contributed to the results.

Diebold-Mariano Test (High)

Table 4.4 shows the results of the DM test for the high price predictions. Similar to the results for the low price predictions, we observe that the SVR model outperforms all other models in predicting the high price of Ethereum Perpetual, as indicated by the significant DM test statistics against all other models except for the DM test statistics againts the LR model. This implies that the LR model exhibits similar forecasting accuracy to the SVR model when predicting the high price of Ethereum Perpetual, based on our evaluation of various models.

The LR model performs significantly better than the RF and LSTM models here as well, as indicated by the significant DM test statistics of 4.89 and 37.57, respectively. This suggests that the SVR model has better forecasting accuracy for the high price of Ethereum Perpetual than the RF and LSTM models.

At last, the RF model performs significantly better than the LSTM model. These results are exactly the same as in the previous case for predicting the low.

Table 4.4: Diebold-Mariano tests between pairs of models (High)

	LR	RF	SVR	LSTM
LR		**4.89**	-1.14	**37.57**
RF			**-21.62**	**24.53**
SVR				**43.92**

Note: The table reports the Diebold-Mariano (adjusted) test statistic between pairs of models. Positive values indicate that the model in the column outperforms the one in the row. Bold font indicates the value is significant at a 5% level.

4.1.2 MAE

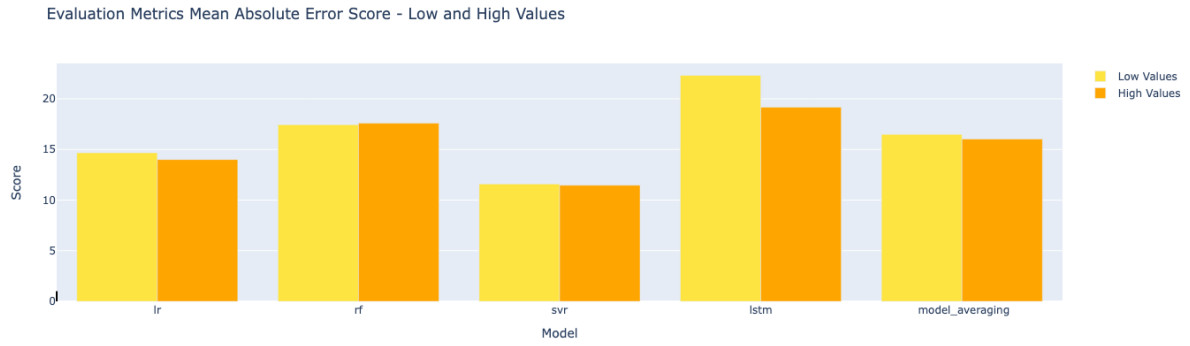


Figure 4.2: MAE comparison of the four models

MAE Low

Figure 4.2 displays the MAE scores for the four models in predicting the next hour's low value. We observe that the SVR model has the lowest MAE score of 11.58, followed by the LR model with a score of 14.67. The model Averaging comes next with a MAE score of 16.49. On the other hand, the RF model and LSTM model have higher MAE scores of 17.43 and 22.31, respectively.

It is interesting to note that the LR model, which performs significantly the best in terms of MSE, has a higher MAE score than the SVR model. This can be because MAE is more sensitive to outliers, and the SVR model may be better at fitting to these outliers than the LR model.

MAE high

Similar to the low values, Figure 4.2 shows the MAE scores for the four models in predicting the next hour's high value. We observe that the SVR model has the lowest MAE score of 11.46, followed by the LR model with a score of 14.00. Model Averaging has a MAE score of 16.03. On the other hand, the RF model and LSTM model have higher MAE scores of 17.58 and 19.17, respectively.

Again, the SVR model has the lowest MAE score for high values, whereas it had the lowest MAE score for low values. This may indicate that the SVR model is better at capturing the patterns in the high values as well than the other models. On the other hand, the LSTM model's higher MAE score suggests that it may not be the best model for predicting high values among

these models.

4.1.3 CRPM

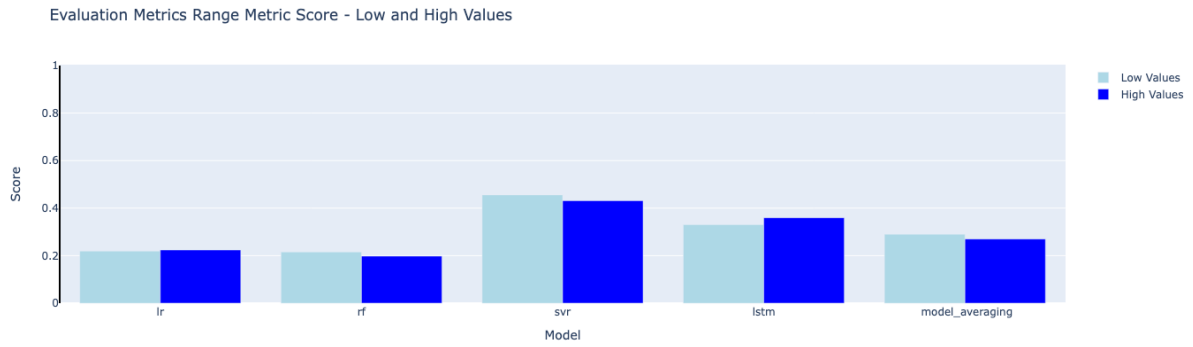


Figure 4.3: CRPM comparison of the four models

CRPM Low

We can observe from Table 4.1 and Figure 4.3 that the SVR model has the highest CRPM score of 0.46, indicating that it has made more conservative predictions compared to the other models. The LR and RF models have the lowest-highest CRPM score of 0.22, followed by the LSTM model, which has the second-highest CRPM score of 0.33. The lower CRPM score of LR and RF models indicate that the models have made riskier predictions compared to the models LSTM and SVR.

It is worth noting that the CRPM metric is specifically designed to evaluate the models based on the company's objective of favoring conservative predictions. Therefore, the SVR model is likely the most suitable model for the company's needs.

Also, this time the model averaging is having a desired effect. It has now a better CRPM than LR.

CRPM high

Table 4.2 and Figure 4.3 display the CRPM scores for the four models in predicting the next hour's high value. The SVR model has the highest CRPM score of 0.43, indicating that it has made the most conservative predictions among the models. The RF model has again the worst CRPM score of 0.20, followed by the LR model with a score of 0.22. The LSTM has a very similar CRPM score of 0.36, which is again the second best model regarding to CRPM score. Regarding the model averaging, it has a higher CRPM score than LR but lower than SVR.

Similar to the evaluation of the next hour's low value, we can see that the CRPM metric highlights the SVR model as the most suitable model for the company's needs in predicting the next hour's high value. Despite the better performance of LR model over LSTM on MSE, the LR model has made riskier predictions than LSTM and may not be the most appropriate model for the company's needs.

Results interpretation

Based on the evaluation metrics, namely MSE, MAE, CRPM, and DM test, we observe certain patterns among the four models (LR, RF, SVR, and LSTM) used for predicting the next hour low and high values of Ethereum Perpetual.

The SVR model demonstrates the lowest MSE score, suggesting good performance in terms of minimizing squared differences between predicted and actual values. Based on the results of the DM test, it is evident that the SVR model outperforms the other models in terms of accuracy when predicting the low value. Besides the best MSE score, SVR has also the best MAE and CRPM score. This suggests that the SVR model is a well-rounded model that performs consistently well across different evaluation metrics. The SVR model's ability to make more conservative predictions, as reflected in its high CRPM score, makes it a suitable choice for companies that prioritize risk management. In addition to its ability to handle non-linear relationships, the SVR model is also able to handle high-dimensional feature spaces. This may have been an advantage in this scenario, where the dataset contained a large number of features.

The LR model performed better comparing to LSTM and RF when we looked at the MSE and MAE score. However, it does not have the best CRPM score indicating that it makes less conservative predictions compared to the other models. It is worth noting that MSE measures the average of the squared differences between predicted and actual values, while MAE measures the average absolute difference between predicted and actual values. It is surprising that the LR model has the best MSE score among all models, as linear models are typically not well-suited for capturing non-linear relationships between variables. However, in a high auto-correlation environment at short lags, the LR model may still perform well due to the strong signal present in the data generating process. Additionally, in a high signal-to-noise ratio (SNR) environment, it may not be necessary to complicate the models by introducing more complex features or models, as they may start to fit the noise. On the other hand, in a low SNR environment, more complex models may be needed to extract whatever signal is present, as linear models may be subpar in capturing the non-linear relationships in the data.

A possible interpretation for the performance rankings could be the dataset size and its impact on the RF and LSTM models. It is worth noting that the dataset has been reduced from millions to around 20,000 observations, which might still be relatively small for RF and LSTM models to exhibit their full potential. Both RF and LSTM models typically require a larger dataset to effectively capture complex patterns and temporal dependencies present in the data.

In contrast, the SVR model's ability to handle non-linear relationships and high-dimensional feature spaces might have allowed it to perform better even with a smaller dataset. SVR can leverage the available data more efficiently and capture the underlying patterns, making it relatively less affected by the dataset size limitation.

Therefore, the smaller dataset size could be a contributing factor to the relatively lower performance of RF and LSTM models compared to SVR and LR. However, it is important to consider that dataset size is just one aspect influencing model performance, and other factors such as feature selection, hyperparameter tuning, and data quality also play significant roles.

Regarding the model averaging model, we have observed that it performs worse than the individual models LR and SVR. However, when considering the CRPM metric, it scores better

than LR but worse than SVR.

This outcome is logical because by averaging the predictions, the model inherently tends to make more conservative predictions. As a result, it will consistently have a higher CRPM score compared to the model with the lowest CRPM score, and a lower CRPM score compared to the model with the highest CRPM score. In our case, SVR achieves the highest CRPM score.

In this particular case, the model averaging approach did not yield superior performance when compared to SVR, and it also did not outperform LR.

These findings suggest that the model averaging technique did not provide substantial benefits over the individual models in terms of predictive accuracy, specifically when evaluated using the CRPM metric. The conservative nature of the averaged predictions likely limited its ability to outperform SVR, which achieved the highest CRPM score. Furthermore, the model averaging model did not surpass the performance of LR, indicating that LR may be more suitable for this particular forecasting task.

The observed performance of the model averaging model, where it performs worse than the individual models LR and SVR, can be attributed to several factors.

Firstly, the model averaging approach inherently introduces a level of conservatism by combining the predictions of multiple models. This conservatism aims to reduce the potential impact of extreme predictions and uncertainties associated with individual models. However, in situations where one of the individual models (in this case, SVR) consistently produces more accurate and less conservative predictions, the averaged predictions may be pulled towards a more conservative stance. This can result in the model averaging model underperforming compared to the superior individual model. Secondly, the nature of the CRPM metric, which measures the accuracy of predictions relative to a reference series, also plays a role. The CRPM score is influenced not only by the absolute accuracy of predictions but also by the variation and distribution of the reference series. If the reference series exhibits characteristics that align more closely with the predictions of SVR, the model averaging model, with its inherent conservatism, may not be able to match or surpass the performance of SVR in terms of CRPM score. Furthermore, the specific characteristics and dynamics of the dataset being analyzed can impact the performance of different models. It is possible that the dataset used in this case favors the modeling approach of SVR, which allows for more flexibility and capturing non-linear relationships. The model averaging model, which combines predictions from LR and SVR, may not fully leverage the strengths of SVR in this context, resulting in its inferior performance.

Overall, the SVR model is the the top-performing model for predicting the next hour low of Ethereum Perpetual depending on our chosen evaluation metrics and the specific evaluation metric of the company.

4.2 Feature Importance

A feature importance analysis is performed to evaluate the influence of individual features on the predictive performance of the models by using the method mentioned in subsection 3.1.4. In Figure 4.4(a), it is observed that the first 12 features are highly important for all models, while the remaining 10 features are comparatively less important. Notably, these features comprise

high, low, close, and their lags, indicating that range, volatility, and returns have less impact on improving the models' predictive power.

For a more detailed comparison within the first 12 features, Figure 4.4(b) provides a breakdown of the features' importance rankings for each model. Although the ranking of the first 12 features is consistent across all models, the weight that each feature carries differs for each model.

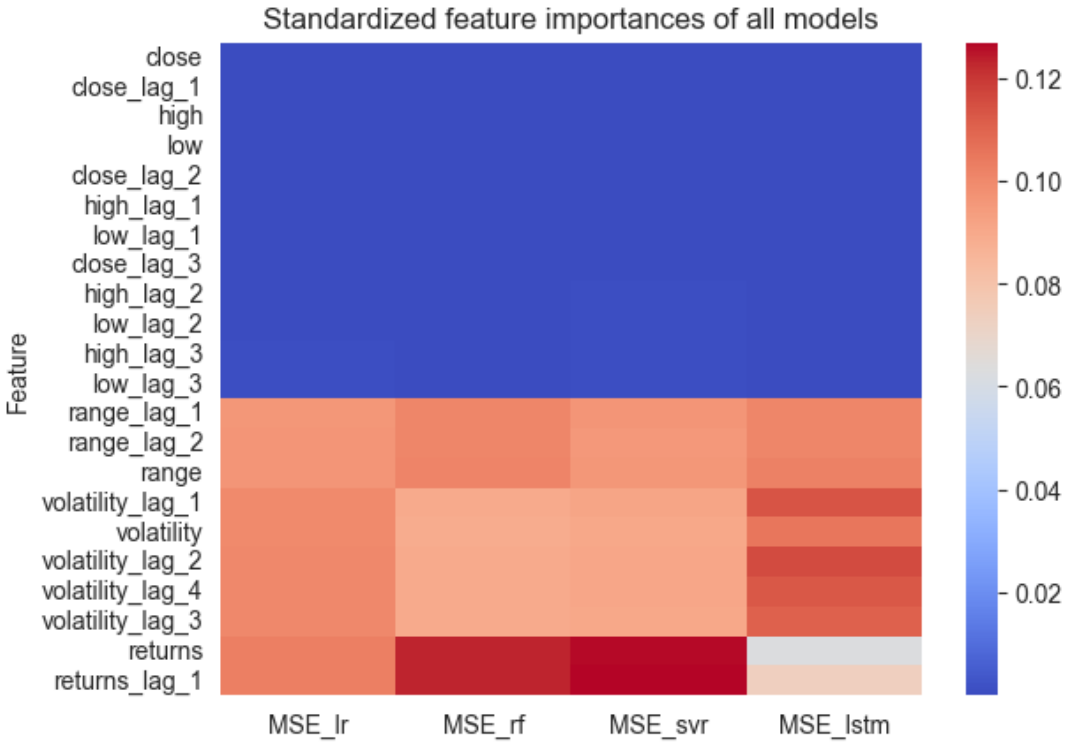
According to Figure 4.4(b), the most important feature among the first 12 for all models is "close," followed by "close lag 1," "high," "low," "close lag 2," "high lag 1," "low lag 1," "close lag 3," "high lag 2," and "low lag 2", "high lag 3" and "low lag 3." This order is reasonable as the most recent information, contained in the last lag, should be the most relevant in predicting future prices. Note that close refers the the last observed price.

Surprisingly, the rankings of the last 10 features are not as important as the first 12 features, which are all high, low, close and their lags. This can be due to the fact that these features carry less information for predicting Ethereum Perpetual high and low prices compared to the first 12 features. Additionally, the last 10 features include volatility, returns, ranges and their lags, which are more related to the historical performance of the Ethereum Perpetual rather than its current trend or momentum. This can be why they are less important for predicting future prices.

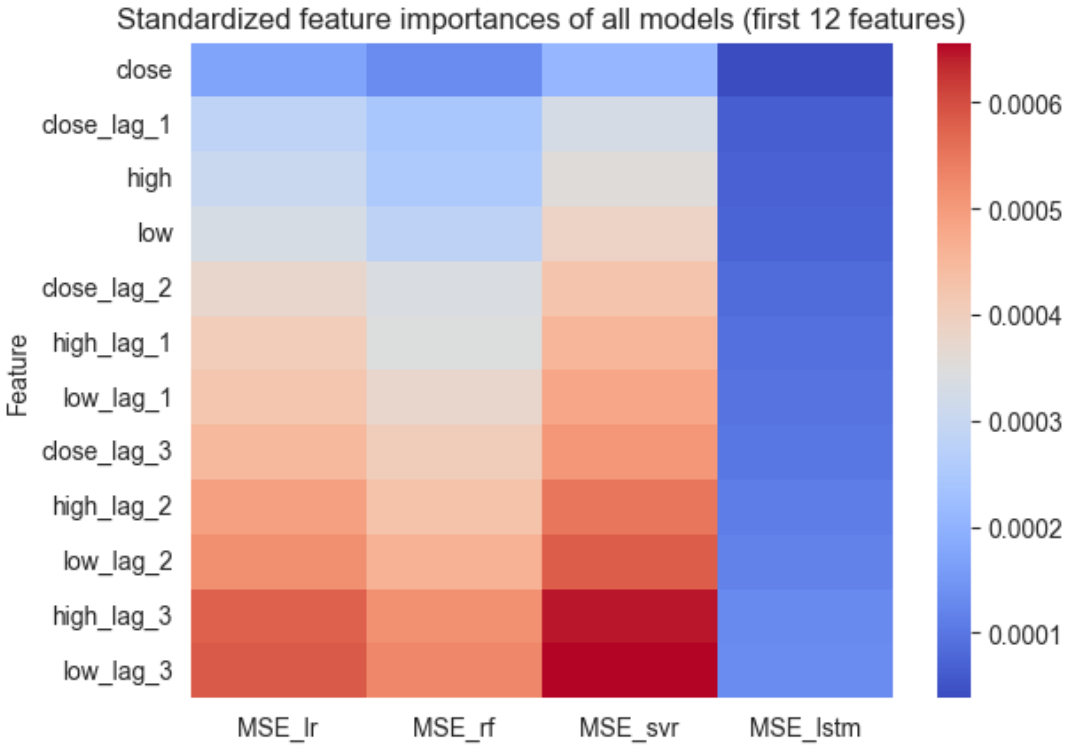
Besides, the high, low, and last close prices of a financial instrument can potentially contain information about momentum and trend. For instance, when prices are on an upward trend, the high and low prices tend to be higher than the previous ones, and the closing price tends to be closer to the high. Similarly, when prices are on a downward trend, the high and low prices tend to be lower than the previous ones, and the closing price tends to be closer to the low. This behavior can indicate the presence of momentum and trend in the price movements, and thus the high, low, and close prices may play a crucial role in predicting future price movements.

Additionally, it is worth noting that features such as volatility and returns may not be as informative on a short time frame. This can explain why they are not as important as the high, low, and last close features. It is possible that the information contained in volatility and returns is indirectly captured by the high, low, and last close features of the previous hour.

At last, the distinct rankings of the last 10 features across all models can suggest that these features have less predictive power and might not be as relevant for modeling the Ethereum Perpetual prices. However, the fact that the first 12 features have consistent importance rankings across all models suggests that they are more influential and can be considered the key drivers of predicting Ethereum Perpetual prices. Therefore, it might be more beneficial for investors to focus on analyzing the trends and momentum of high, low, and close prices and their lags, rather than relying on historical performance metrics such as volatility, returns, and ranges.



(a) Feature Importance of all features



(b) Feature Importance of the first 12 features

Figure 4.4: Feature Importance of the features

4.3 Trading Strategy

In this trading strategy, we use ETH Futures & Perpetual, which have a 0.00% maker fee and a 0.05% taker fee. We have set some constraints for the testing environment. Every trade has a lot size of 1, and there is a maximum of 50 positions that we can hold. If we reach the maximum number of positions, we won't buy or sell anymore, depending on what the current positions are.

We set up the same environment for both static and dynamic strategies across all models. For the dynamic models, we use exactly the same dynamic strategy and do not tune it for each model to compare more fairly. The only element that can make a difference between the performance of the dynamic trading strategies is the prediction of the high and low of each model. This means that the performance of the models in predicting these values is crucial for the overall performance of the dynamic trading strategies.

4.3.1 Static strategy

The static strategy produces profitable results based on the virtual environment and on historical market conditions. Figure 4.6(a) shows the buy and sell orders placed during the testing period for the static strategy. As can be seen from the figure, the red triangles indicates a sell order being fulfilled and a green triangle means a buy order being fullfilled. Figure 4.6(b) shows the position of the asset over time, where positive values represent a long position and negative values represent a short position. It can be observed that there are more sell opportunities in this market condition, since the position has reached the max short position limit multiple times.

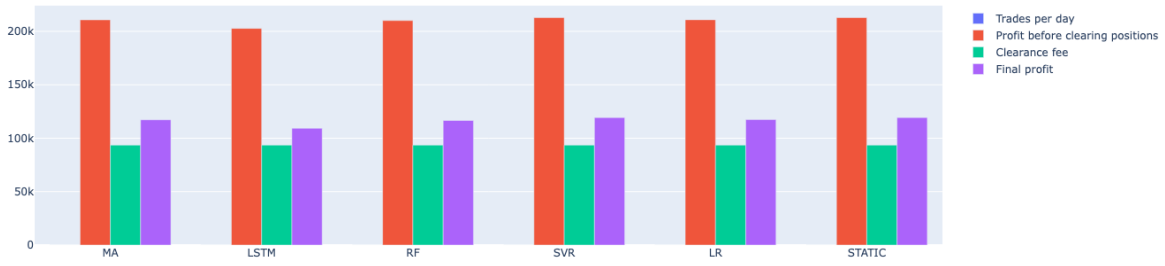
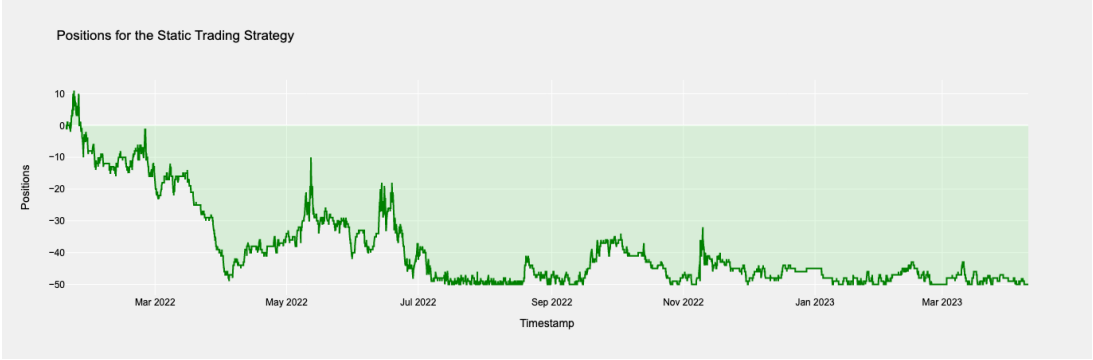


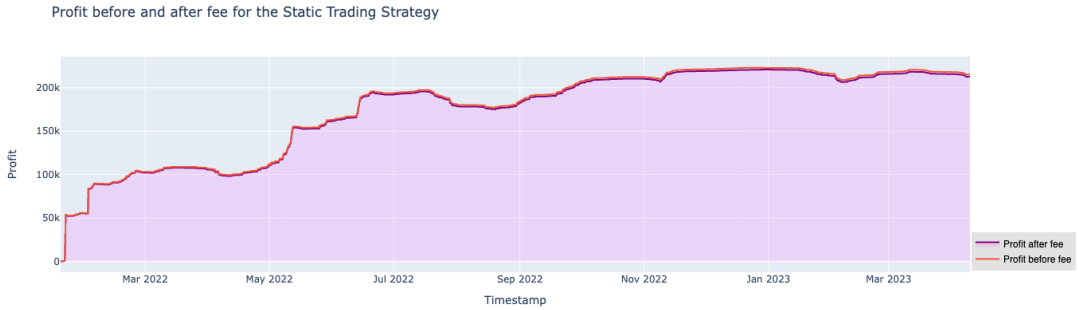
Figure 4.5: Trading results of all models

Table 4.5 and Figure 4.5 summarize the trading results of the static strategy. The strategy yields a profit of 210.840.26 USD before clearance and 119,321.56 USD after clearance, the highest among the five models. A clearance means that at the end of the test we clear our positions to zero using the market price at that time. The clearance fee for all models is the same, 93,497.50 USD, meaning that all models hold the same position at the end, 50 short. The Sharpe ratio of the static strategy is 43.18, which indicates a very high risk-adjusted return. The number of trades per day is 5.6, with the static strategy having the highest number of trades. It can be noted that this strategy is a low-latency strategy. It is obvious that The static strategy performes better than all the dynamic models.

(a) Buy and Sell orders



(b) Positions



(c) Feature Importance of the first 12 features

Figure 4.6: Profit before and after trading fee

4.3.2 Dynamic strategy

We present the trading results summary for four different models: LSTM, RF, SVR, and LR. Besides we have also incorporated the model averaging model of LR and SVR. Plots like Figure 4.5 for the four models and model averaging model can be found in the Appendix F.

All models exhibit a similar number of trades per day, ranging from 5.41 to 5.46. The profit before clearing positions is also consistent across models, with values ranging from 210068.39 to 212782.32. After taking into account the clearance fee, the final profit ranges from 109247.07 to 119284.82. The Sharpe ratio is high and similar across models, ranging from 42.45 to 42.64. Overall, the trading results suggest that the four models perform not too differently in terms of profitability and fee-adjusted returns.

However, despite the similar trading results, the SVR model stands out with the best trading performance among the tested models, aligning with our previous evaluation of each model individually. This can be partly attributed to its highest CRPM score, lowest MAE and MSE scores, indicating a strong balance between conservative predictions and prediction accuracy. The SVR model's highest final profit and Sharpe ratio further reinforce its superior performance. While the performance of SVR did not surpass the static model, it came remarkably close. These findings indicate that the SVR model holds promise as an effective approach for predicting Ethereum Perpetual returns and making profitable trading decisions.

The results suggest that by further enhancing the CRPM score, which reflects the conservatism of predictions, the performance of the SVR model could potentially outperform the static strategy. Note that the CRPM score of SVR is even below 0.50. This implies that efforts to improve the model's conservative predictions could lead to even better trading results and increased profitability. The encouraging performance of the SVR model motivates further exploration and refinement to leverage its predictive capabilities for successful trading strategies in the realm of Ethereum Perpetual.

Furthermore, the LSTM model performes poorly, which may be attributed to the difficulty in hyperparameter tuning. On the other hand, the RF and LR models perform similarly, as observed in the MAE and CRPM scores. This finding reinforces the idea that focusing on multiple evalution metrics such as MSE, MAE and CRPM scores. However, despite the LSTM model having a better CRPM score, its MAE (and MSE) is significantly worse than the RF and LR models, which may explain its inferior performance.

While the CRPM score is essential for evaluating the conservatism of predictions, it should not overshadow the importance of accurate forecasting. Achieving a balance between accurate and conservative predictions is crucial for a successful trading strategy. Therefore, it is advisable to prioritize models with superior MSE and MAE scores initially, as they lay the foundation for reliable predictions. Once accurate predictions are established, efforts can be directed towards enhancing the CRPM score to further refine the trading strategy and mitigate risk.

It is worth noting that the trading performance results align with the ranking of MSE and MAE scores among the models. This observation suggests a high correlation between the model's prediction accuracy, as reflected by MSE and MAE, and its trading performance. Models with lower MSE and MAE scores tend to exhibit better trading performance, as demonstrated by their higher final profits. This highlights the importance of accurate forecasting in achieving successful trading strategies and reinforces the need to prioritize models with superior MSE and MAE scores during model selection and development.

Table 4.5: Trading results summary of different models

Model	LSTM	RF	SVR	LR	STATIC	MA
Trades per day	5.45	5.41	5.46	5.43	5.60	5.45
Profit before clearance	202744.57	210068.39	212782.32	210840.26	212819.06	210717.21
Clearance fee	93497.5	93497.5	93497.5	93497.5	93497.5	93497.5
Final profit	109247.07	116570.89	119284.82	117342.76	119321.56	117219.71
Sharpe ratio	42.51	42.45	42.64	42.53	43.18	42.58

Chapter 5

Discussion

Cryptocurrencies are still in their nascent stages, and the lack of regulation and oversight means that the market is highly susceptible to manipulation and fraud. Accurate price range predictions can help investors and regulators to identify abnormal market behavior and take necessary actions to protect the interests of investors, but it is important to acknowledge the limitations of this study.

The results presented in this study are based on the dynamics of the Deribit exchange, including features such as the high and low prices of Ethereum perpetual contracts traded on the platform. Therefore, the findings may not be immediately transferable to other cryptocurrency exchanges, as the underlying market dynamics and data structures may differ. Caution should be exercised when interpreting and applying the findings of this study in contexts outside of the data source.

It is worth noting that despite the inherent volatility and unpredictability of the cryptocurrency market, the models developed in this study are able to capture some of the extreme price movements in the test set. This suggests that the models may still perform well under certain market conditions, such as periods of high volatility.

However, more research is needed to verify the reliability and accuracy of the models, especially in the face of unexpected events or major market disruptions. Given the dynamic nature of the cryptocurrency market, it is important to continuously evaluate and update the models based on new data and emerging market trends. The models developed in this study are still based on past data and may not accurately reflect future market conditions.

Another important point of discussion is how machine learning models can enhance trading strategies. In this study, we use machine learning models to transform a static trading strategy into a dynamic one. Furthermore, the use of machine learning models in trading is a relatively new field, and further research is necessary to refine and improve the models' performance. As the market continues to evolve, so must the models and strategies used to trade cryptocurrencies. It is also important to note that the machine learning models did not significantly enhance the static method in terms of generating more profit. This may be attributed to the fact that the models were selected based on the mean squared error (MSE) score, which may not be the best metric to evaluate the performance of trading models. As mentioned earlier, the cumulative rank probability metric (CRPM) and mean absolute error (MAE) metrics could be more meaningful in the context of trading. Thus, selecting models based on these metrics rather than solely

relying on the MSE score might yield better results. Nonetheless, the results suggest that machine learning models can be valuable tools in developing and improving trading strategies, but caution should be exercised when using them as the sole basis for making trading decisions.

In addition, during the training of the models, it is possible to enhance their performance by incorporating additional features. For example, incorporating the low and high values of Bitcoin as features can help capture market trends that may influence the price of the Ethereum Perpetual. A study by Vassiliadis, Papadopoulos, Rangoussi, Konieczny and Gralewski (2017) suggests that cryptocurrencies exhibit a high level of correlation. Incorporating additional features into the model can help to refine the predictions and make them more robust to market fluctuations. Note that the models may not perform as well in periods of extreme market turbulence or unexpected events, such as regulatory changes or major security breaches.

Further research can also explore the potential for incorporating additional features into the models to improve their predictive power. For example, incorporating data on social media sentiment, news articles, or regulatory announcements can help to capture market sentiment and provide a more comprehensive picture of the underlying market dynamics.

The choice of time interval is an important aspect to consider when developing trading strategies. In this study, the trading strategy employed in the virtual environment has a low latency and an average of 5-6 trades per day. This indicates that a longer interval, such as 4-5 hours, might be more appropriate than the 1-hour interval used in this analysis. Additionally, the virtual setting we use in this study may not be representative of real-world trading conditions and the number of trades per day may vary when the settings of the trading environment are set to real trading environment. Therefore, historical data of the real trading performance and setting should be used to determine the optimal time frame for predicting high and low values and to provide feedback for adjusting the strategy accordingly.

At last, when using machine learning models for real implementation, it is crucial to tune the model and the trading strategy to the specific market conditions and trading objectives. In this study, the trading strategy was not explicitly tuned for the dynamic models, but it can be adjusted by fine-tuning how the price is adjusted based on the predictions of the model. This tuning process involves setting aside a portion of the data for validation and testing, in addition to the training set used to train the model. By doing so, the adjustments made to the trading strategy can be evaluated and optimized based on their impact on the overall performance of the strategy. Ultimately, tuning the trading strategy to the specific model and market conditions is essential for achieving optimal results in real-world trading scenarios.

Chapter 6

Conclusion

In our comparative analysis of machine learning methods for predicting the high and low prices of Ethereum Perpetual, we find that Support Vector Regression (SVR) outperforms other machine learning techniques. SVR provides valuable insights into the relevance of nonlinear and interaction effects when predicting the prices of Ethereum Perpetual. Given the significant reduction in our dataset from millions of observations to approximately 20,000, we have a compelling rationale to harness the potential of deep learning. However, despite this advantage, our results also show that a simple Linear Regression model outperforms non-linear models such as Random Forest and LSTM in terms of accuracy. This may be attributed to the signal-to-noise ratio in asset pricing. Deep learning and non-linear models are specifically designed to capture complex and non-linear relationships between predictors and outcomes. However, these models can sometimes suffer from overfitting and instability. It is important to consider these factors when choosing a suitable model. Additionally, it is worth noting that while Linear Regression performed well in terms of accuracy, it may not be the optimal choice for real-world applications where other evaluation metrics, such as the CRPM used by the company, are employed to assess model performance. In fact, our evaluation using the CRPM metric demonstrates that all machine learning models outperform Linear Regression. Therefore, when selecting the best machine learning model for a specific application, it is crucial to consider not only accuracy but also other relevant evaluation metrics.

When it comes to model averaging, the practice of combining the predictions of multiple models, it does not always guarantee improved performance. In our study, we employed model averaging by averaging the predictions of LR and SVR models. However, despite the potential benefits of ensemble modeling, the averaged model did not outperform the individual LR and SVR models across all metrics, including MSE, MAE, and trading performance. While the model averaging approach showed an improvement in the CRPM metric, indicating a higher level of conservatism in predictions, it fell short in terms of prediction accuracy and trading performance compared to the individual models. This outcome suggests that the averaging process, which inherently produces more conservative predictions, may hinder the model's ability to capture the true underlying patterns in the data. These findings highlight the importance of carefully considering the trade-offs associated with model averaging. While it may offer the advantage of increased stability and reduced variance in predictions, it can also result in a compromise in terms of accuracy and performance. Therefore, it is crucial to thoroughly evaluate the performance of

both individual models and the averaged model across multiple metrics and consider the specific objectives and requirements of the task at hand before deciding to employ model averaging as a strategy.

Furthermore, we have developed a dynamic trading strategy to expand the utilization of machine learning forecasts as a means to construct a profitable trading strategy. We observed that a limited set of dominant predictive signals were consistent across all methods, including the last price, last hourly high, and last hourly low. The success of machine learning algorithms in predicting the hourly high and low prices brings significant potential for the development and enhancement of trading strategies and risk management techniques. By employing machine learning techniques, we can reduce estimation errors in high and low price predictions, thus facilitating the identification of underlying economic mechanisms that drive asset pricing phenomena.

This study aimed to assess the effectiveness of four machine learning models in improving a static trading strategy within the realm of cryptocurrency trading. The performance of the trading strategy was evaluated using important metrics, including profit, clearance fees, and the Sharpe ratio. These metrics were used as indicators to gauge the overall success and efficiency of the trading strategy when integrated with machine learning models.

According to the findings, the SVR model demonstrated superior performance compared to the other models. The exceptional performance of SVR can be attributed to its capability to effectively utilize relatively smaller datasets, whereas RF and LSTM may require larger datasets to achieve optimal results in this particular use case. On the other hand, the LSTM model exhibited poor performance. The RF and LR models performed similarly, with similar MAE and CRPM scores.

It is worth noting that none of the models improved the static trading strategy in terms of profit. This outcome could be attributed to the selection of models based solely on the MSE metric. These results suggest that alternative metrics such as CRPM and MAE may be more suitable for evaluating the performance of machine learning models in trading applications. Incorporating these metrics into the model selection process could potentially lead to more effective enhancements of trading strategies.

The implications of this study are significant for the development of machine learning-based trading strategies. The results highlight the potential of machine learning models in improving trading performance but also emphasize the importance of model selection and tuning. Furthermore, the findings underscore the need for caution in using machine learning models as the sole basis for trading decisions and suggest that models should be integrated into a broader trading strategy.

To conclude, our study emphasizes the need for careful experimentation and evaluation of different machine learning methods before deploying a solution. Our results confirm the increasing importance of machine learning in the fintech industry and its potential to advance our empirical understanding of financial derivatives. Moreover, despite our finding that linear regression outperformed non-linear models in accuracy, it is worth mentioning that all machine learning models tested in our study surpassed the evaluation metric of the company, CRPM.

References

Akyildirim, E., Goncu, A. & Sensoy, A. (2021). Prediction of cryptocurrency returns using machine learning. *Annals of Operations Research*, *297*(1), 3-36. Retrieved from https://EconPapers.repec.org/RePEc:spr:annopr:v:297:y:2021:i:1:d:10.1007$_s$10479−020 − 03575 − y

Almeida, M., Welker, J., Siddiqui, S., Luiken, J., Ekker, S., Clark, K., ... McGrail, M. (2021). Endogenous zebrafish proneural cre drivers generated by crispr/cas9 short homology directed targeted integration. *Scientific Reports*, *11*(1), 1732.

Cao, L. & Tay, F. (2001). Financial forecasting using support vector machines. *Neural Computing and Applications*, *10*, 184–192. doi: 10.1007/s005210170010

Gu, S., Kelly, B. & Xiu, D. (2020, 02). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223-2273. Retrieved from https://doi.org/10.1093/rfs/hhaa009 doi: 10.1093/rfs/hhaa009

Huang, W., Nakamori, Y. & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers Operations Research*, *32*(10), 2513-2522. Retrieved from https://www.sciencedirect.com/science/article/pii/S0305054804000681 (Applications of Neural Networks) doi: https://doi.org/10.1016/j.cor.2004.03.016

Israel, R., Kelly, B. T. & Moskowitz, T. J. (2020). Can machines 'learn' finance? *ERN: Other Econometrics: Econometric & Statistical Methods - Special Topics (Topic)*.

Jović, A., Brkić, K. & Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (mipro)* (p. 1200-1205). doi: 10.1109/MIPRO.2015.7160458

Kim, K.-J. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1), 307-319. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231203003722 (Support Vector Machines) doi: https://doi.org/10.1016/S0925-2312(03)00372-2

Lahmiri, S., Bekiros, S. & Salvi, A. (2018). Long-range memory, distributional variation and randomness of bitcoin volatility. *Chaos, Solitons Fractals*, *107*, 43-48. Retrieved from https://www.sciencedirect.com/science/article/pii/S0960077917305209 doi: https://doi.org/10.1016/j.chaos.2017.12.018

Livieris, I. E., Pintelas, E. G., Stavroyiannis, S. & Pintelas, P. (2020). Ensemble deep learning models for forecasting cryptocurrency time-series. *Algorithms*, *13*, 121.

Moral-Benito, E. (2015). Model averaging in economics. *Journal of Economic Surveys*, *29*(1), 46-

75. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/joes.12044 doi: https://doi.org/10.1111/joes.12044

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, *12*(Oct), 2825–2830.

Rahman, A. H. A. & Rijal, A. (2012). Support vector machine for solving small dataset problem..

Vassiliadis, S., Papadopoulos, P., Rangoussi, M., Konieczny, T. & Gralewski, J. (2017). Bitcoin value analysis based on cross-correlations. *Journal of Internet Banking and Commerce*, *22*(S7), 1.

Yaya, O., Vo, X., Ogbonna, A. & Adewuyi, A. (2020, 06). Modelling cryptocurrency high-low prices using fractional cointegrating var. *International Journal of Finance Economics*, *27*. doi: 10.1002/ijfe.2164

Zippia. (2022, June). How many businesses accept bitcoin? [2023]: 21 important bitcoin statistics. *Zippia.com*. Retrieved from https://www.zippia.com/advice/how-many-businesses-accept-bitcoin/

Appendix A

Plot of features

The appendix provides a detailed view of the three main features we use to analyze the Ethereum Perpetual market besides the high, low and previous price of Ethereum Perpetual. The figures in this section illustrate the past hour returns percentage and volatility of this cryptocurrency. Figure A.1 showcases the returns percentage of the Ethereum Perpetual over the past hour, offering a glimpse into the potential price range of this asset.

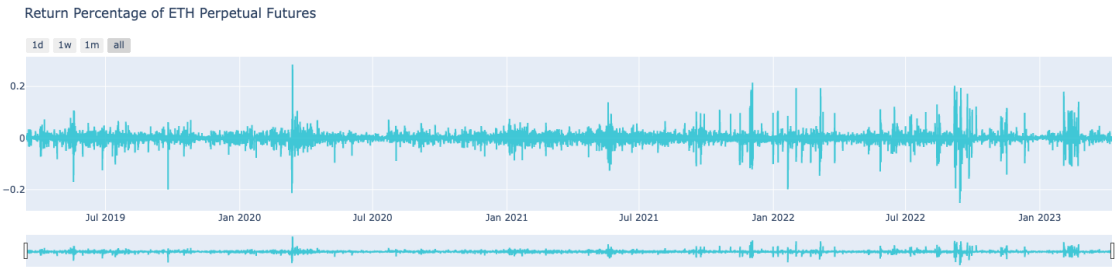


Figure A.1: Paste hour Returns percentage of the Ethereum Perpetual

Similarly, Figure A.2 highlights the volatility of the Ethereum Perpetual, revealing how much its price has fluctuated over the past hour. These features are crucial in understanding the behavior and dynamics of the Ethereum Perpetual market, and can be valuable tools for traders and investors seeking to make informed decisions.

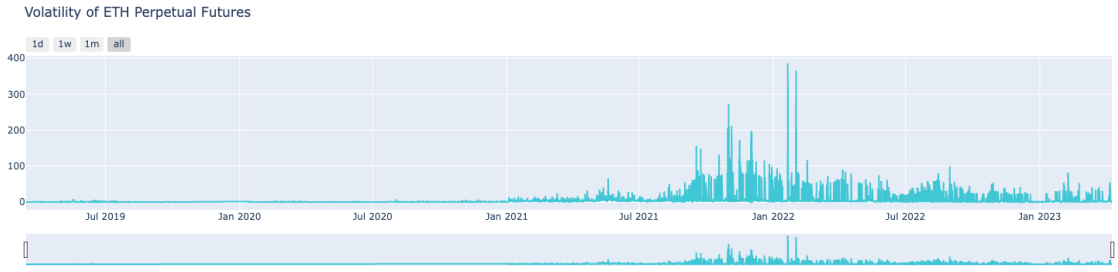


Figure A.2: Past hour Volatility of the Ethereum Perpetual

Appendix B

Feature Selection

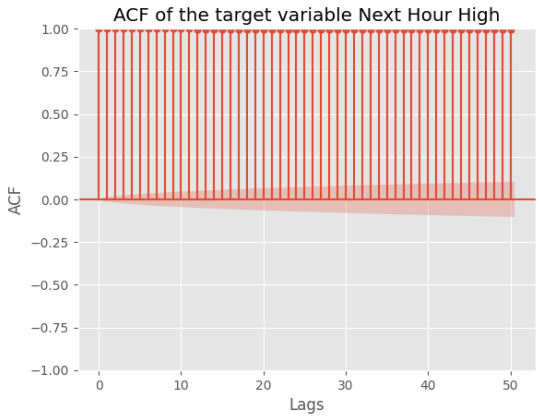
B.1 Hourly high

We use the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots to identify the number of lags to include in our model for the hourly high feature. The ACF plot, shown in Figure B.1(a), indicates a high value of autocorrelation for a constant lag, which is not informative for our time series analysis. On the other hand, the PACF plot, shown in Figure B.1(b) and the T-test in Table B.1 shows a significant correlation for the first four lags.

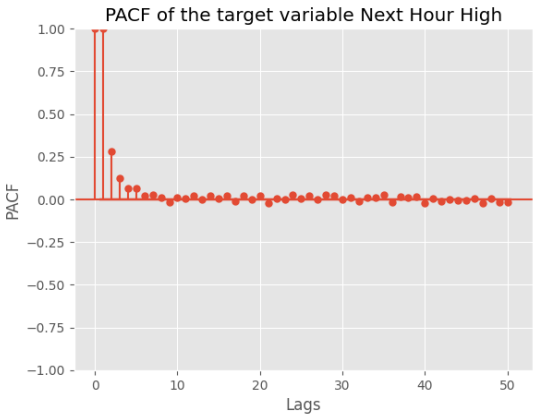
Table B.1: T-test Results for hourly high

Lag	**p-value**	**Significant**
Lag 1	0.012	Yes
Lag 2	0.036	Yes
Lag 3	0.042	Yes
Lag 4	0.047	Yes
Lag 5	0.152	No
Lag 6	0.269	No
Lag 7	0.451	No
Lag 8	0.605	No

Based on these observations, we decide to include up to four lags in our model for the hourly high feature.



(a) Autocorrelation plot of the hourly high feature



(b) Partial autocorrelation plot of the hourly high feature

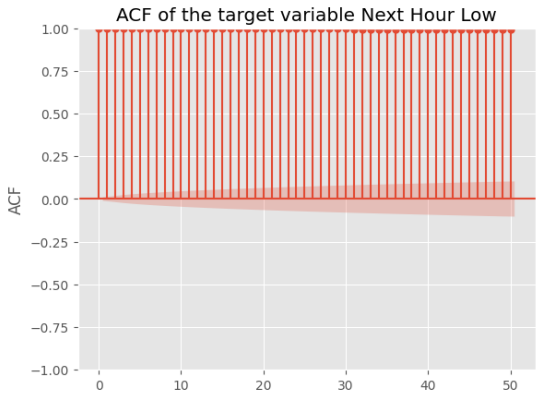
Figure B.1: Autocorrelation and partial autocorrelation plots of the hourly high feature

B.2 Hourly low

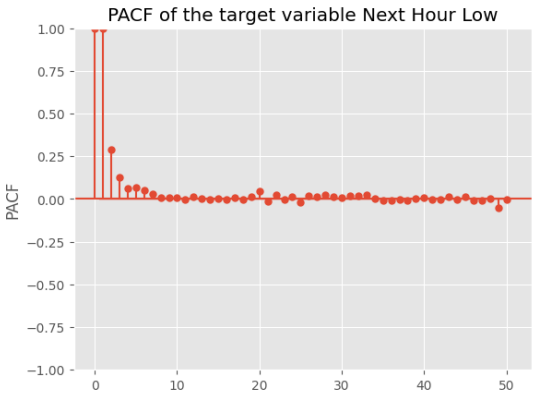
Similar to the analysis done for the hourly high feature, we use ACF and PACF plots to determine the number of lags to include in our model for the hourly low feature. The ACF plot, shown in Figure B.2(a), has high values of autocorrelation for a constant lag, indicating no informative lag structure. In contrast, the PACF plot, shown in Figure B.2(b) exhibits significant autocorrelation for the first two lags, followed by decreasing values up to the fourth lag. The T-test in Table B.2 confirms this finding. Based on these observations, we decide to include up to four lags as well in our model for the hourly low feature.

Table B.2: T-test Results for hourly low

Lag	**p-value**	**Significant**
Lag 1	0.021	Yes
Lag 2	0.029	Yes
Lag 3	0.036	Yes
Lag 4	0.038	Yes
Lag 5	0.182	No
Lag 6	0.299	No
Lag 7	0.461	No
Lag 8	0.618	No



(a) Autocorrelation plot of the hourly low feature



(b) Partial autocorrelation plot of the hourly low feature

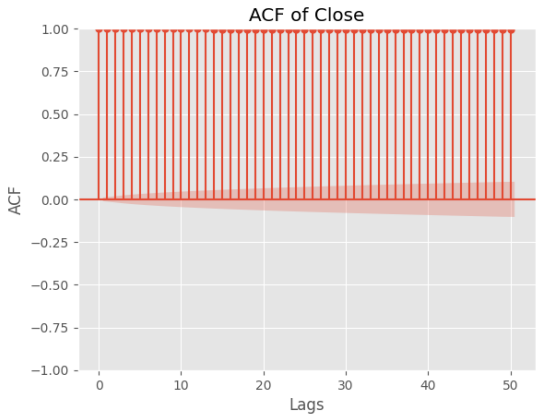
Figure B.2: Autocorrelation and partial autocorrelation plots of the hourly low feature

B.3 Hourly last price

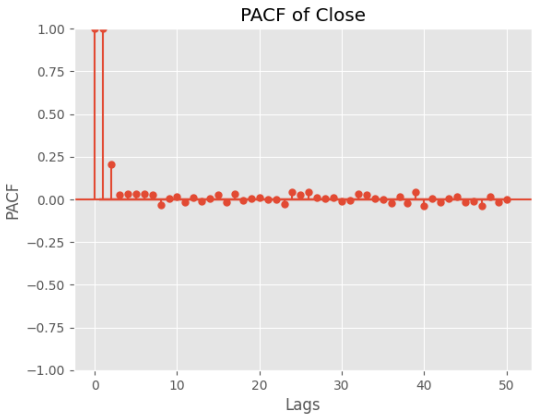
For the hourly last price feature, the ACF plot in Figure B.3(a) shows that all lags are highly correlated with the previous observation. The PACF plot in Figure B.3(b) , on the other hand, shows that the first two lags have very high correlation coefficients, and the third lag is also observable higher than the rest. The T-test in Table B.3 also shows a significant p-value for the first 3 lags. Therefore, we include the first three lags of the hourly last price feature in the model.

Table B.3: T-test Results for hourly last price

Lag	**p-value**	**Significant**
Lag 1	0.012	Yes
Lag 2	0.036	Yes
Lag 3	0.042	Yes
Lag 4	0.067	No
Lag 5	0.182	No
Lag 6	0.299	No
Lag 7	0.461	No
Lag 8	0.697	No



(a) Autocorrelation plot of the hourly last price feature



(b) Partial autocorrelation plot of the hourly last price feature

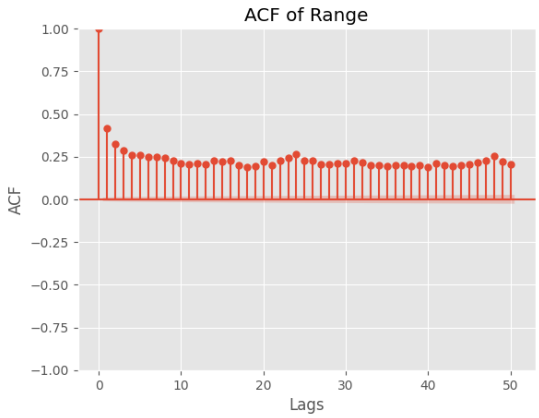
Figure B.3: Autocorrelation and partial autocorrelation plots of the hourly last price feature

B.4 Hourly range

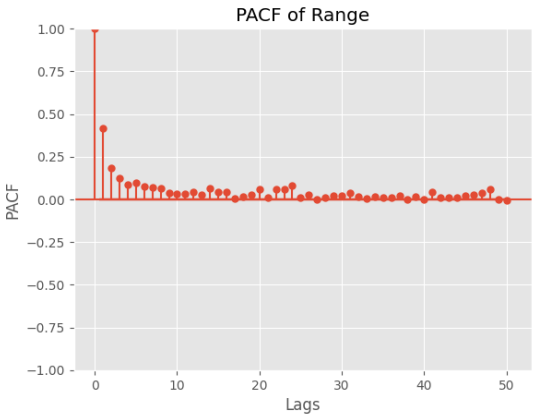
Figure B.4 shows the ACF and PACF plots of the hourly range feature. The ACF plot indicates a high value of autocorrelation for the first two lags, which decreases but still remains higher than the rest for lags 3 and above. The PACF plot shows a significant correlation for the first two lags, with the first one being very high and the second one also much higher than the rest. The T-test in Table B.4 confirms that the first two lags are significant. Based on these observations, we decide to include up to two lags in our model for the hourly range feature.

Table B.4: T-test Results for hourly range

Lag	**p-value**	**Significant**
Lag 1	0.011	Yes
Lag 2	0.032	Yes
Lag 3	0.058	No
Lag 4	0.073	No
Lag 5	0.189	No
Lag 6	0.301	No
Lag 7	0.559	No
Lag 8	0.815	No



(a) Autocorrelation plot of the hourly range feature



(b) Partial autocorrelation plot of the hourly range feature

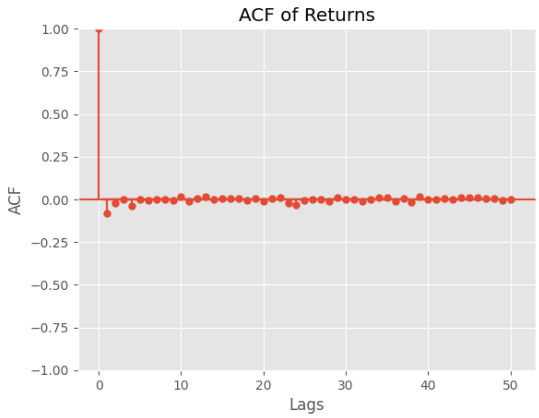
Figure B.4: Autocorrelation and partial autocorrelation plots of the hourly range feature

B.5 Hourly return percentage

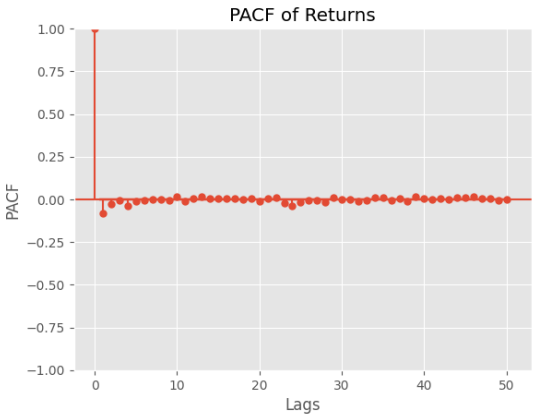
We also use ACF and PACF plots to determine the number of lags to include in our model for the hourly return percentage feature. Figure B.2 shows the ACF and PACF plots for this feature. The ACF plot indicates that there is a significant autocorrelation for only the first lag, while the PACF plot shows a significant correlation for only the first lag as well, with the rest of the lags being almost zero. The T-test in Table B.5 shows only a significant p-value for the first lag. Based on this observation, we only include one lag in our model for the hourly return percentage feature.

Table B.5: T-test Results for hourly return percentage

Lag	**p-value**	**Significant**
Lag 1	0.017	Yes
Lag 2	0.062	No
Lag 3	0.121	No
Lag 4	0.205	No
Lag 5	0.346	No
Lag 6	0.520	No
Lag 7	0.741	No
Lag 8	0.876	No



(a) Autocorrelation plot of the hourly return percentage feature



(b) Partial autocorrelation plot of the hourly return percentage feature

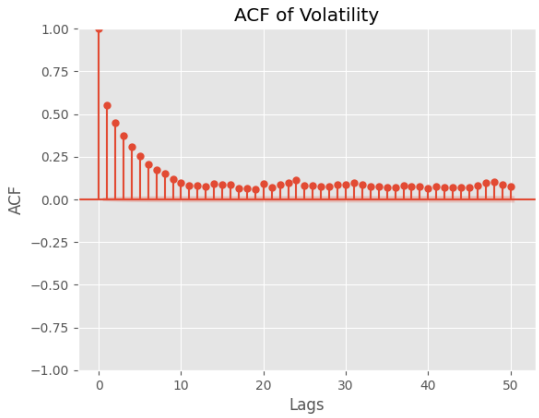
Figure B.5: Autocorrelation and partial autocorrelation plots of the hourly return percentage feature

B.6 Hourly volatility

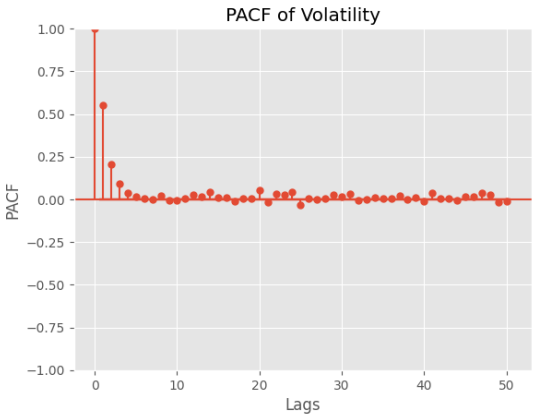
Lastly, for the hourly volatility feature, the ACF plot, shown in Figure B.6(a), indicates a high value of autocorrelation for the first lag, followed by decreasing autocorrelation for the next lags. Similarly, the PACF plot, shown in Figure B.6(b) and the T-test in Table B.6, indicates an observable correlation for the first four lags, followed by almost zero correlation for the rest of the lags. Based on these observations, we decide to include up to four lags in our model for the hourly volatility feature.

Table B.6: T-test Results for hourly volatility

Lag	**p-value**	**Significant**
Lag 1	0.009	Yes
Lag 2	0.015	Yes
Lag 3	0.021	Yes
Lag 4	0.032	Yes
Lag 5	0.183	No
Lag 6	0.399	No
Lag 7	0.558	No
Lag 8	0.717	No



(a) Autocorrelation plot of the hourly volatility feature



(b) Partial autocorrelation plot of the hourly volatility feature

Figure B.6: Autocorrelation and partial autocorrelation plots of the hourly volatility feature

Appendix C

Correlation

The correlation matrix in Figure C.1 displays the pairwise correlation coefficients between all the features we use in the analysis. The correlation matrix reveals interesting insights into their relationships. About two-thirds of the features have a correlation lower than 0.5 or higher than -0.5, indicating weak linear relationships between them. However, 33.77% of the features have a correlation higher than 0.9 or lower than -0.9. It is important to note that some of these highly correlated features are actually lagged versions of a given feature. For example, the high feature has four lagged versions included, which are highly correlated with the original feature. Therefore, we need to be cautious when interpreting the correlation matrix and consider the possibility of multicollinearity among the features.

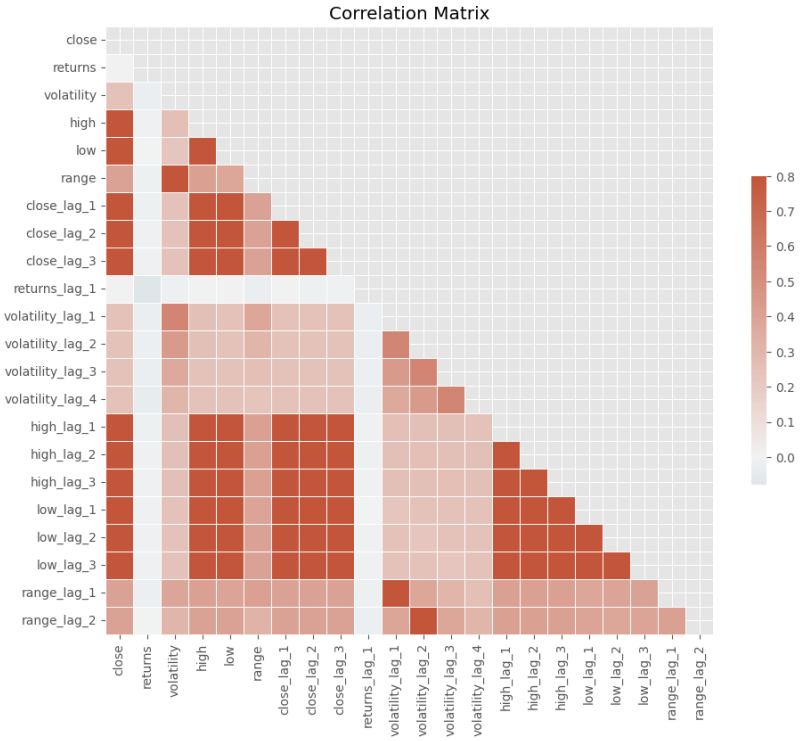


Figure C.1: Correlation matrix of the features

Appendix D

Hyperparameters

Table D.1: Hyperparameters for the Tuned Methods

RF	SVR	LSTM
Depth $= 1 \sim 6$	C $\in (10^{-1}, 10^{4})$	Units $\in \{16, 32, 64, 128, 256\}$
#Trees $\in (100, 600)$	$\gamma =$ scale	Activation $\in$ {relu, tanh, sigmoid}
#Features in each split	Kernel	Learning rate init
$\in (1, 12)$	$\in$ {sigmoid, poly, rbf}	$\in \{0.001, 0.01, 0.1, 0.5\}$
	$\epsilon \in (10^{-1}, 10^{-5})$	Batch size
		$\in$ {16, 32, 64, 128, 256, 512}
		Epochs $\in \{20, 40, 60, 80, 100\}$

Note: The table outlines the hyperparameters that we fine-tune in each machine learning technique for the models we optimize.

Table D.1 details the set of hyperparameters and their corresponding possible values utilized for adjusting each machine learning model. We choose our hyperparameters based on a balance between computational limits and performance.

D.1 RF hyperparameters

For the RF model, we have chosen to fine-tune three hyperparameters: depth, number of trees, and the number of features in each split.

- **Depth**: The depth of the tree determines the number of splits the tree has, and the model can become overfitted if the depth is too high. Therefore, we limit the depth to a range of 1 to 6.
- **Trees**: The number of trees determines the number of decision trees used to build the model. A higher number of trees can lead to better performance, but it also increases the computation time. We chose to fine-tune the number of trees in the range of 100 to 600, ensuring a balance between computational resources and model performance.
- **Features per split**: The number of features in each split determines the number of features randomly selected to determine the best split at each node. Choosing a high number of features can lead to better model performance but also requires more computational resources. We chose to fine-tune the number of features in each split in the range of 1 to 12.

D.2 SVR hyperparameters

For SVR, we have utilized four main hyperparameters in our hyperparameter tuning process: C, gamma, kernel, and epsilon.

- **C**: C is the regularization parameter, which controls the trade-off between a smooth decision boundary and classifying training points correctly. A smaller value of C creates a smoother decision boundary, while a larger value of C results in a more complex decision boundary, potentially overfitting the data.
- **Gamma**: The gamma parameter in the radial basis function (RBF) kernel and determines the shape of the decision boundary. In our implementation, we set gamma to 'scale' so that it is calculated as

  $$1/(n_{features} * X.var())$$

  where X is the training data. This means that the value of gamma is adjusted based on the number of features in the data and the variance of the data. A smaller value of gamma results in a smoother decision boundary, while a larger value of gamma can lead to overfitting.
- **Kernel**: The kernel function we use in SVR controls how the decision boundary is calculated. We have experimented with three different kernel functions: sigmoid, polynomial, and radial basis function (RBF). The sigmoid kernel is suitable for linearly non-separable problems, while the polynomial kernel is effective for problems with non-linear decision boundaries. The RBF kernel is suitable for problems where the data points are not separable in a linear space.
- **Epsilon**: This is the margin error, which specifies the distance between the support vector and the decision boundary. A smaller value of epsilon results in a wider margin, allowing for more margin errors, while a larger value of epsilon results in a smaller margin, leading to fewer margin errors.

D.3 LSTM hyperparameters

For the LSTM model, we fine-tune the following hyperparameters:

- **Units**: This determines the number of memory units in the LSTM layer. A higher number of units can lead to better model performance, but also increases the computational cost. We fine-tune the number of units in the range of 16 to 256.
- **Learning rate**: This determines the step size at each iteration while updating the weights of the model. A higher learning rate can lead to faster convergence but can also cause the model to overshoot the optimal solution. We fine-tune the learning rate in the range of 0.001 to 1.0.
- **Activation function**: This function is used to introduce non-linearity into the output of each neuron in the LSTM layer. We fine-tune the activation function with three options: ReLU, tanh, and sigmoid.

- **Epochs**: Epochs determines the number of times the model is trained on the entire training dataset. We fine-tune the number of epochs in the range of 20 to 100.
- **Batch size**: A batch size is a number of training samples we use in one iteration of gradient descent. A higher batch size can lead to a faster convergence, but also requires more memory. We fine-tune the batch size in the range of 16 to 512.

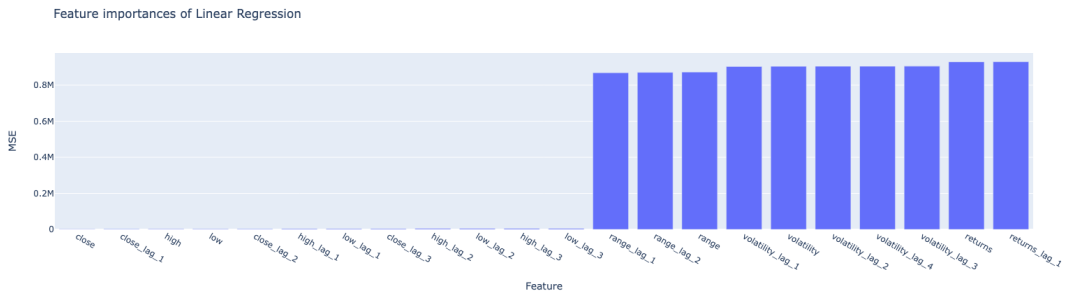
We experiment with different values for each hyperparameter to find the optimal combination that results in the best model performance.

Appendix E

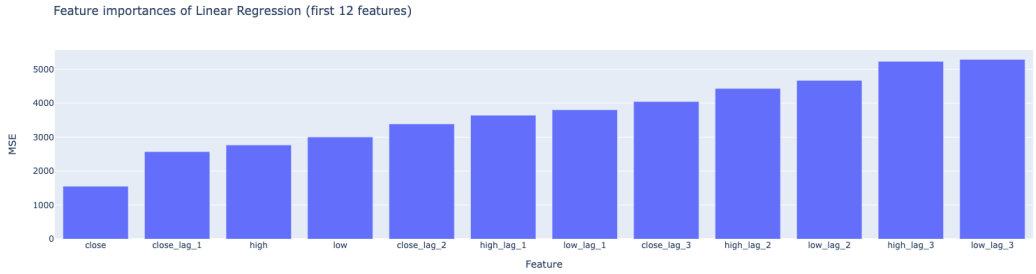
Feature Importance

The figures show the feature importance of each model: Linear Regression, Random Forest, Support Vector Regression, and Long Short-Term Memory. For each model, the feature importance of all features and the first 12 features are plotted. We plot the feature importance of the first 12 features separately to provide more detailed insight into the most relevant features for each model. Focusing on the top 12 features may provide a clearer picture of the most important features for each model.

E.1 Linear Regression



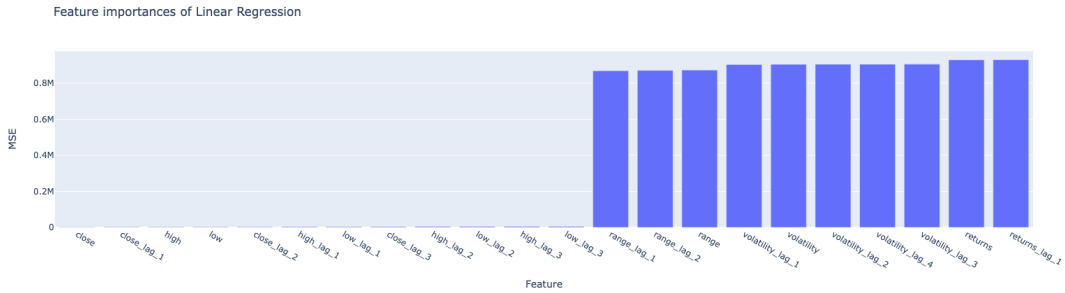
(a) Feature Importance of all features



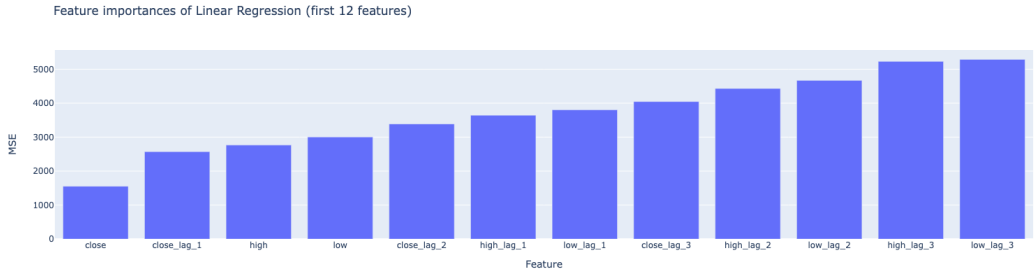
(b) Feature Importance of the first 12 features

Figure E.1: Feature Importance of the Linear Regression model

E.2 Random Forest



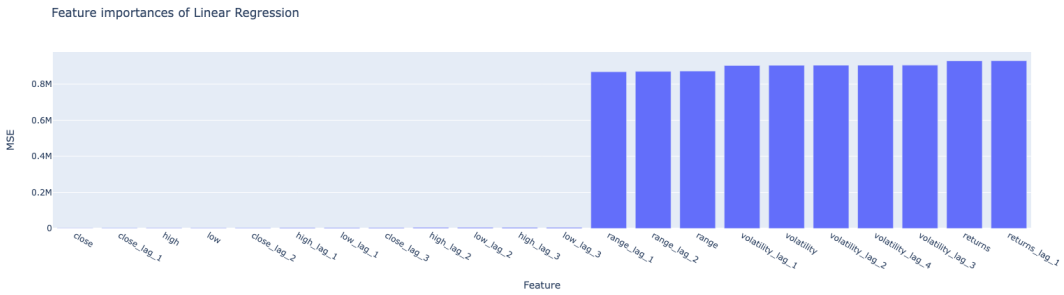
(a) Feature Importance of all features



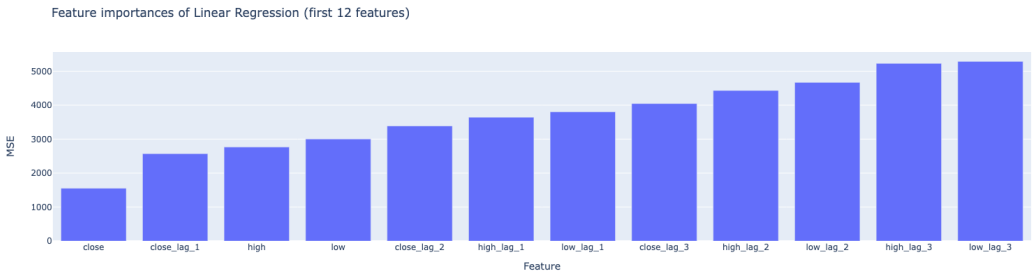
(b) Feature Importance of the first 12 features

Figure E.2: Feature Importance of the Random Forest model

E.3 Support Vector Regression



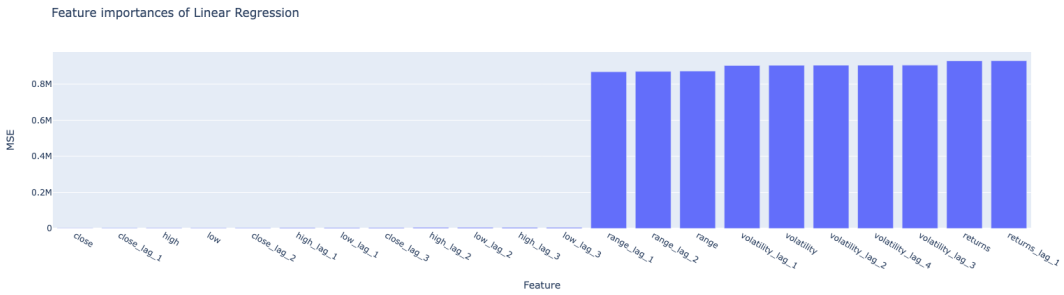
(a) Feature Importance of all features



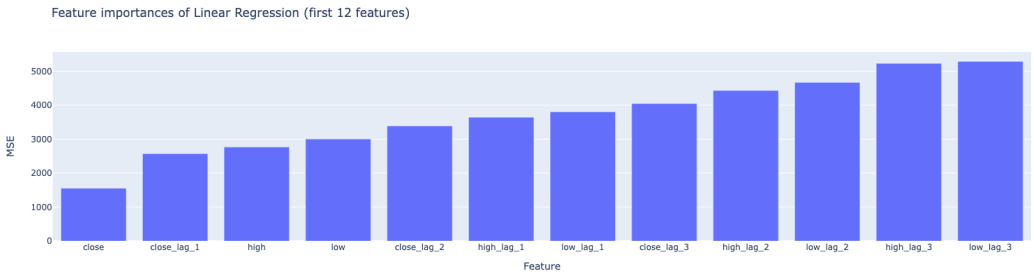
(b) Feature Importance of the first 12 features

Figure E.3: Feature Importance of the Support Vector Regression model

E.4 Long Short-Term Memory



(a) Feature Importance of all features



(b) Feature Importance of the first 12 features

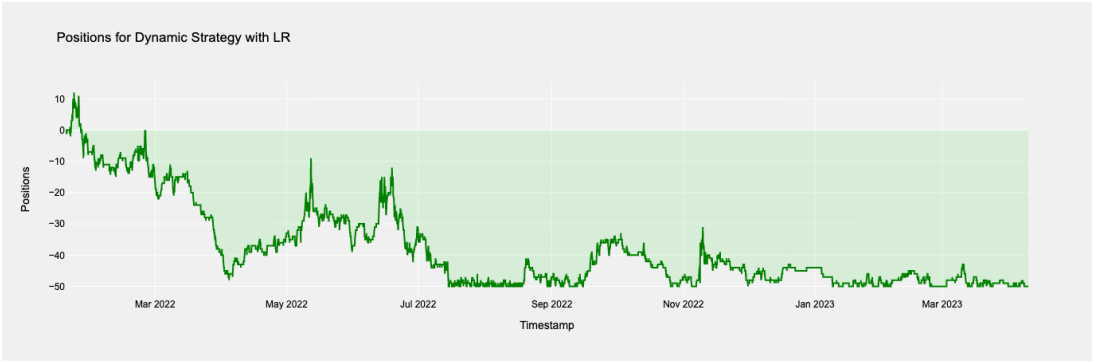
Figure E.4: Feature Importance of the Long Short-Term Memory model

Appendix F

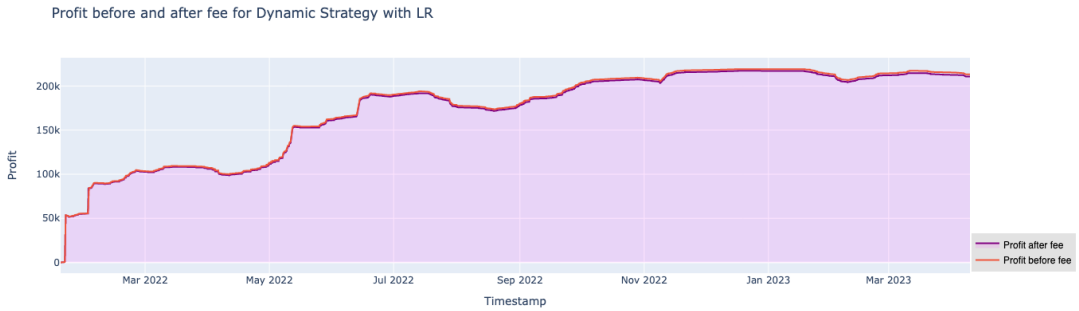
Trading results of the dynamic strategy using the ML models

F.1 Linear Regression Dynamic Trading Results

(a) Buy and Sell orders



(b) Positions

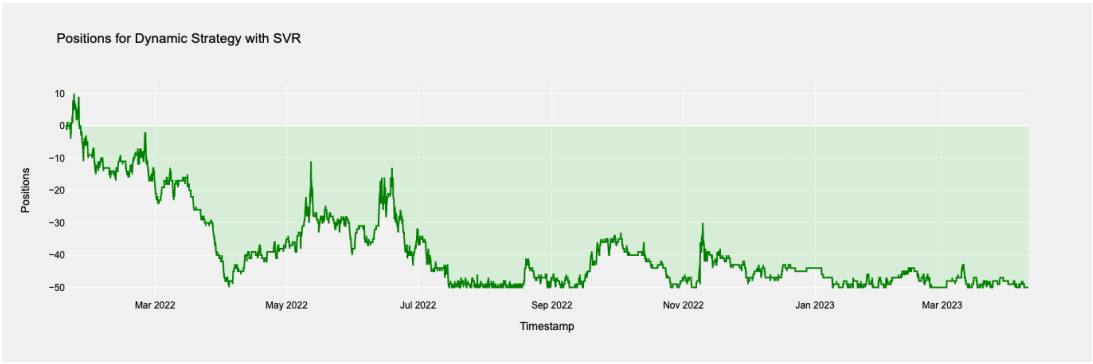


(c) Feature Importance of the first 12 features

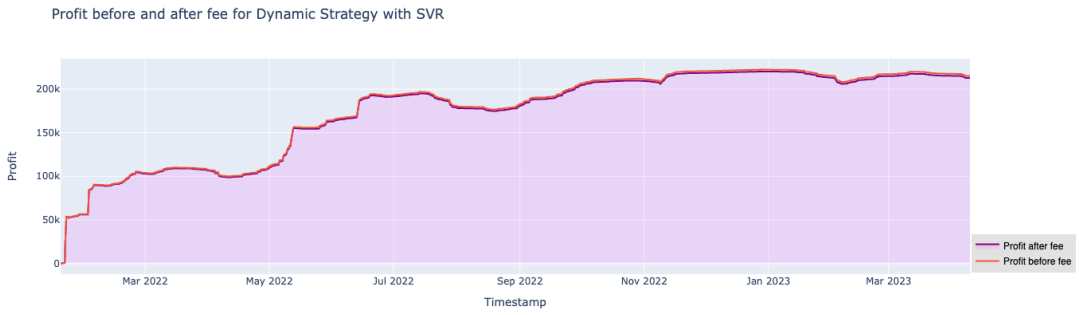
Figure F.1: Profit before and after trading fee

F.2 Support Vector Machine Dynamic Trading Results

(a) Buy and Sell orders



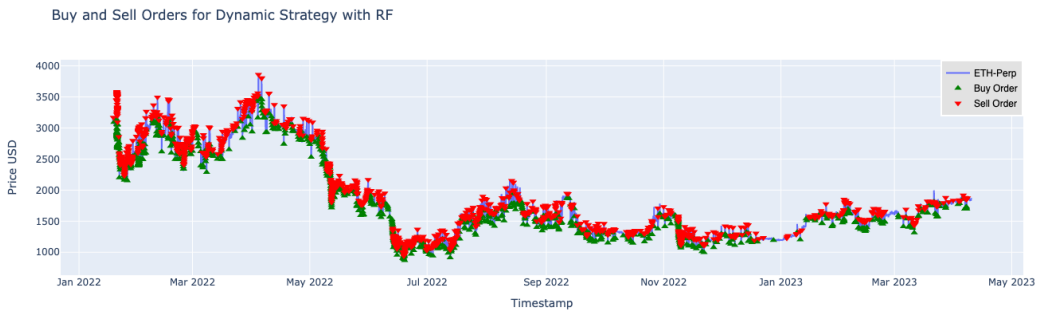
(b) Positions



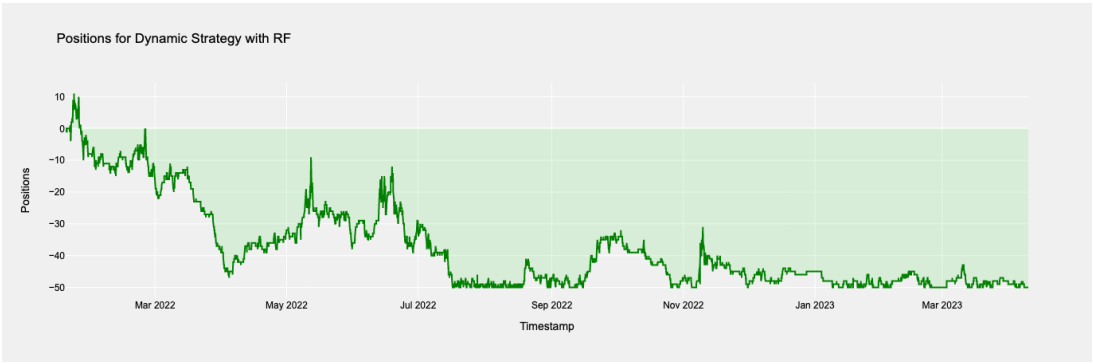
(c) Feature Importance of the first 12 features

Figure F.2: Profit before and after trading fee

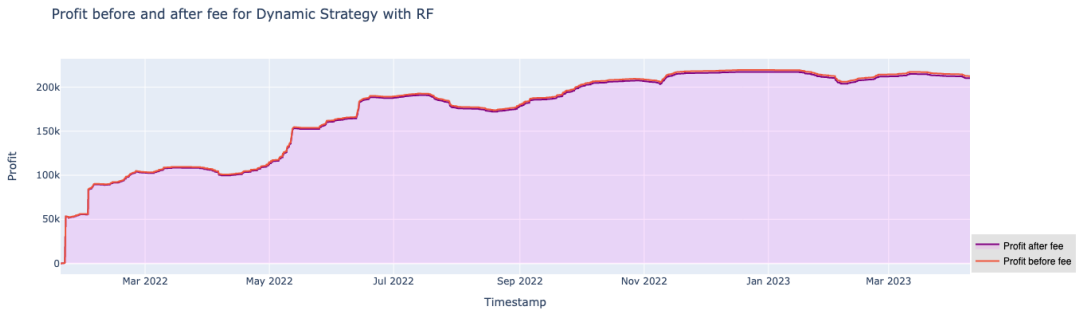
F.3 Random Forest Dynamic Trading Results



(a) Buy and Sell orders



(b) Positions

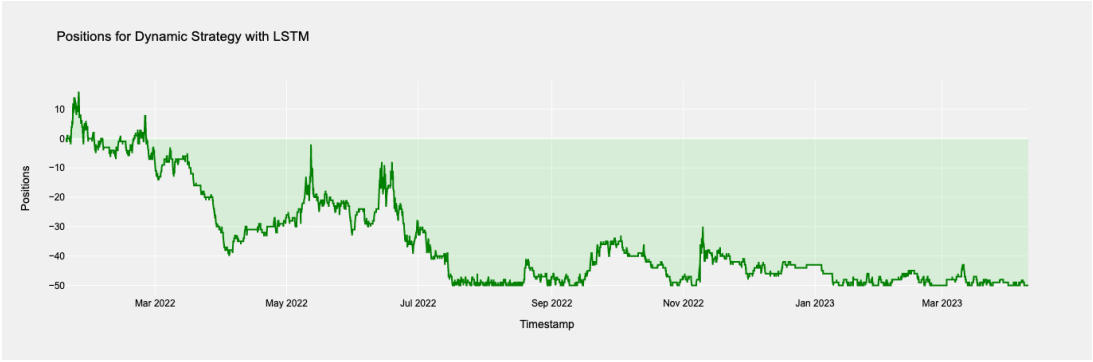


(c) Feature Importance of the first 12 features

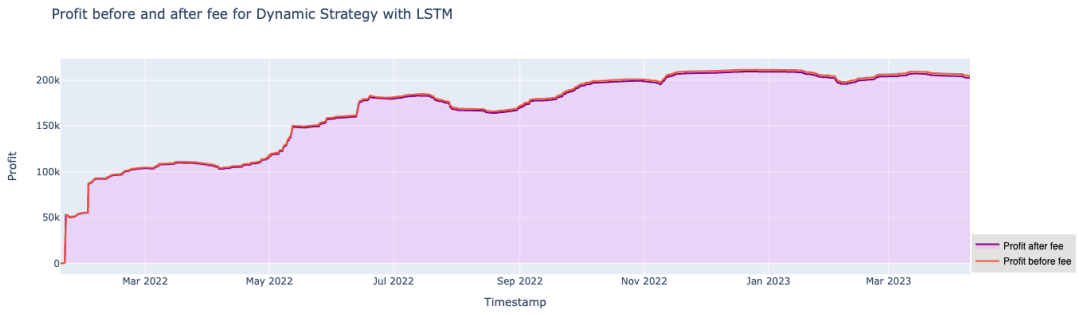
Figure F.3: Profit before and after trading fee

F.4 LSTM Dynamic Trading Results

(a) Buy and Sell orders



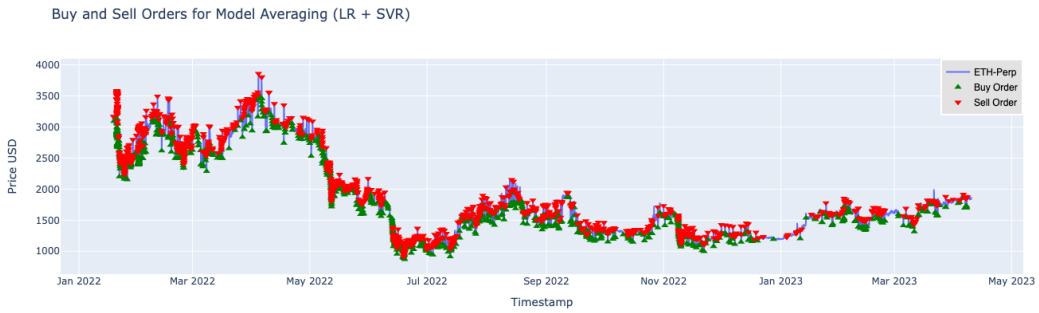
(b) Positions



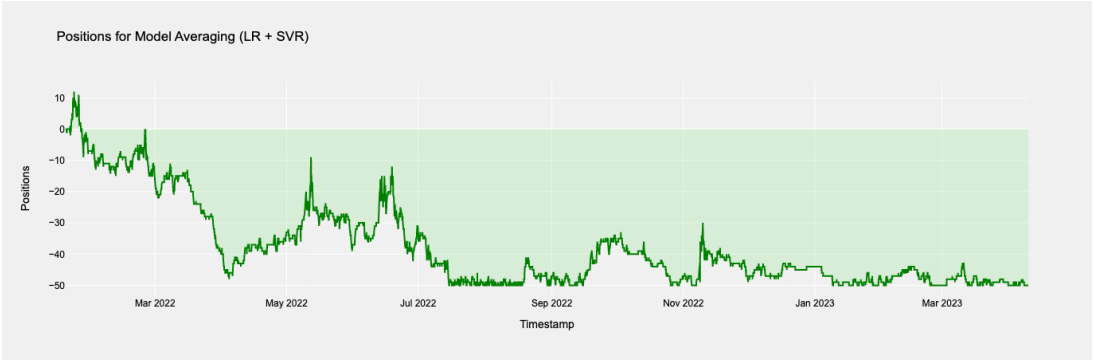
(c) Feature Importance of the first 12 features

Figure F.4: Profit before and after trading fee

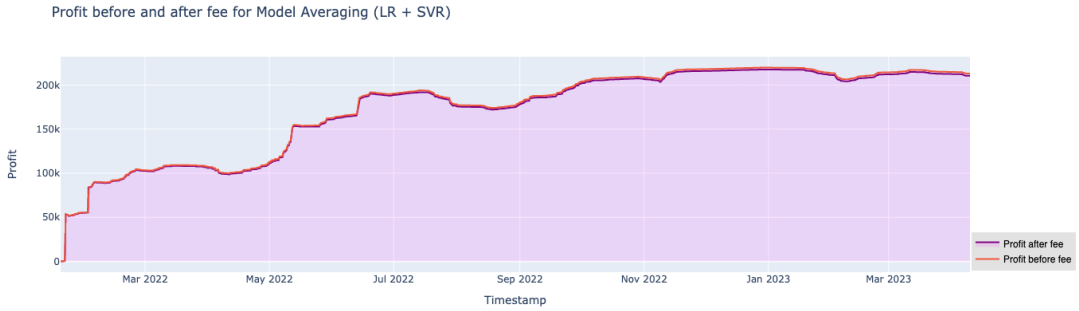
F.5 Model Averaging (LR + SVR) Dynamic Trading Results



(a) Buy and Sell orders



(b) Positions



(c) Feature Importance of the first 12 features

Figure F.5: Profit before and after trading fee