Regression Analysis on Cholesky factors via GA-SVR model: On Forecasting the Covariance Matrix of a Cryptocurrency portfolio

Fernando Victoria Valpuesta (VMGS2)[1]

MSc Machine Learning

Professor Philip Treleaven
Anastasia Bugaenko, SymbioticaAI

September 2022

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Degree in Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

This research investigates a forecasting model for the covariance matrix of a portfolio constructed with a range of cryptocurrencies. The model presented in this Thesis is a combination of two Machine Learning models: Support Vector Regression and Genetic Algorithm. The first algorithm is chosen to capture the linear and the possible nonlinear relationships between the assets while the second one oversees the hyperparameter optimization task, thus seeking to have a better and more accurate understanding of risk and co-movements of the assets. This is essential in Portfolio Optimization and Asset Allocation, and a good estimate of the covariance of assets is at the core of risk management. This study can help Portfolio Managers better understand the amount of risk taken, adjust return to risk ratio and discover new investment insights.

The research motivation is to investigate the possible nonlinear relationships among cryptocurrencies and how these movements affect the overall volatility of a portfolio. In practice, portfolio risk management and correlation analysis is an important topic that every Portfolio Manager takes into account when investing. However, in some cases there is nonlinear relationship among assets, which current correlation and covariance analysis do not take into account, resulting in non-accurate risk management models. For this reason, it is believed that further research should be carried out in the possible nonlinearities in the dynamics of a portfolio. More specifically, cryptocurrencies present a new type of asset that does not behave exactly like the stock market, so this work also contributes to give a better understanding of the dynamics of the cryptocurrency market.

The Thesis is structured in the following way:

1. **Data and Model Selection**: The first part consists in a data exploration analysis, which seeks to find the returns distributions, price-volume relation and other characteristics of the data set. Several tests are performed to see how is the data cor-

related, and according to this, different Kernel functions are included in the Support Vector Regression model. Finally, the data is separated in different sets for training, validating and testing the model.

2. **Model implementation**: The Support Vector Regression (SVR) is the model in charge of making predictions of the covariance matrix, as it is concluded in the first section that it is the best technique for this task. For this reason, the implementation begins with the SVR. Some modifications to traditional SVR are done in order to combine it with Genetic Algorithms for Hyperparameter Optimization. Finally, the benchmark used in all experiments is presented.

3. **Testing and Results**: This Chapter starts with an introduction to the dataset used for the experiments and an explanation on Cross-Validation. It then presents the experiments, the results obtained in all of them and the overall model performances. Finally, an analysis of the results and a summary are presented.

The Thesis is believed to present original contributions to science and practical Machine Learning, which are:

1. Statistical Analysis of cryptocurrencies price, return and volume distributions. The data exploration part gives better insights of the dynamics of cryptocurrencies.

2. Combination of Machine Learning models to form the Support Vector Regression - Genetic Algorithm model. Combined methods tend to perform better than single models because they combine the best of each model for better estimations. In this case, the SVR is in charge of forecasting the covariance matrix, and the GA oversees the optimization of the SVR parameters. Furthermore, the GA dynamically optimizes the SVR to adapt to the particular market conditions.

3. Comparative Regression Analysis on the covariance matrix with different Support Vector Regression (SVR) algorithms. The study presents a range of different Kernel SVRs with different parameters and settings, and investigates how these affect the forecasting performance. To the best of our knowledge, this is the first comparative analysis of SVRs forecasting performance in a DeFi portfolio.

In order to have access to the GitHub containing the code for the project, please contact the author.

Impact Statement

This research presents a novel Machine Learning model for covariance matrix forecasting for Portfolio Allocation. Capturing nonlinear relationships between financial assets is a challenging task that is essential for Asset Management firms to account for all types of risks taken within a portfolio. Portfolio Managers (PM) can benefit from this technique to adjust the Return/Risk ratio (Sharpe) depending on their risk appetite, provide more considerable diversification on their investments, and include cryptocurrencies in their portfolios, which is a new financial asset. Furthermore, this research is also beneficial for Crypto Hedge Funds as it investigates digital asset dynamics. It is believed that with this model, the high volatility that cryptocurrencies suffer can be better understood and forecasted, which is essential for risk management.

Acknowledgements

I would like to thanks my colleagues at Symbiotica who helped me during the writing of this thesis by providing feedback on the report, discussing ideas, and making it a fun experience[1]. Especially my supervisor Ana Bugaenko for her consistent support and guidance during the running of this project. She continuously provided encouragement and was always willing and enthusiastic to assist in any way she could.

Furthermore, I have to thank my family for always encouraging me in the pursuit of further education. They have always supported me in every step in my career. I owe you this degree. Literally.

> 'We can only see a short distance ahead, but we can see plenty there that needs to be done."
>
> — Alan Turing

[1] As fun as writing a thesis can get.

Contents

Chapter 1

Introduction

The objective of this Chapter is to introduce the reader to the research that has been carried out. It starts by a motivation on the topic being researched and why it is important to investigate. Then, the objectives of this work are presented in addition to the experiments carried out. Finally, the Scientific Contributions are presented, as well as a brief outline of the thesis.

1.1 Motivations

Market participants and financial institutions that trade on the market want to profit by taking the least risk possible. Some might be more aggressive in their investments and take larger risks, but no market participant hesitates when a less risky option for the same amount of return is present. Option B is the best. In order to make an investment decision in the present, one must forecast the future, and the more accurate the forecast is, the better the decision can be..

Portfolio Managers oversee the risks in a portfolio and changing assetasset allocation depending on future predictions. Current methods used in the industry to forecast portfolios' covariance matrices are based on linear models. That is, exclusively linear relationships are taken into account. What if a nonlinear model best describes the covariance matrix? Are we missing out on the true Option B with the least risk for the same return?

The purpose of this thesis is to contribute to the existing literature within the field of machine learning in finance, more precisely covariance matrix predictions, by examining a

portfolio of cryptocurrencies. A comparison between linear and nonlinear machine learning models are carried out, and a novel approach based on the fusion of two different models is proposed for this task, which leads to the second question that we try to answer: Does our hybrid model outperforms single models?

Finally, this thesis also aims to shed light upon cryptocurrencies' time series so their high volatility can be better understood and forecasted. With the recent rise in investment interest in digital assets, it is crucial to understand and have better insights into the dynamics of these assets, especially for risk management purposes

1.2 Research Objectives

The research objectives of this thesis are twofold. These can be separated into the finance related and in the machine learning related objectives.

The finance related objective is to explore the nonlinearities in a portfolio, more specifically analysing the covariance matrix of the portfolio. We want to explore if the dynamics of cryptocurrencies and their co-movements can be better explained by a nonlinear model, as opposed to a linear one. This is done with the goal of helping Portfolio Managers better forecast the risk taken in their investments, and have a better understanding on the performances of their portfolios.

The second objective is to analyse different machine learning regression models, and see if a hybrid model based on Genetic algorithms and Support Vector Regression outperforms other models. This comparison is done to have a better understanding on different versions of Support Vector Regression and how machine learning algorithms can be combined together to outperform the single version.

1.3 Research Experiments

The experiments of this thesis can be divided into three components. Each one of this presents a different empirical study involving the data, models and how the latter were tested on the data.

- **Data and Model Selection**: This presents a Time Series Analysis on the cryptocurrencies that looks specifically into the stationarity, autocorrelation, correlation and returns distributions of the series. Based on the results obtained and referring to the literature, the predictive model is selected.

- **Model Implementation**: As the selected model is a Support Vector Regression, the main components of how this model was developed is presented. In addition to this, the optimization part referring to Genetic Algorithms is also explained, stating the libraries and programming languages used for the purpose.

- **Testing and Results**: This chapter presents the tests performed on the models and how their results varied depending on each experiment. A total of three experiments are carried out, and a final analysis of these is presented.

1.4 Scientific Contributions

This research contributes to the existing literature in a number of ways:

1. Detailed Time Series Analysis on the portfolio assets. This thesis explores properties of the time series used, such as stationarity, autocorrelation, correlation and returns distribution. Unlike the conventional finance literature that focuses on traditional assets, this study contributes to the existing finance literature by providing insights into the dynamics of digital assets and the impact of their returns fluctuations on a portfolio.

2. Comparative analysis of Machine Learning regression models. A notable contribution is the analysis of the ML models for the covariance matrix prediction task, where more than five different models are tested in three different tasks with a diverse range of settings.

3. Implementation of a hybrid GA-SVR Machine Learning model for portfolio allocation task. The model selection part introduces the GA-SVR model, which is the first of its kind to be used for covariance matrix prediction of a digital assets portfolio. This method has been existing for a while, but has not been applied to this application domain before.

1.5 Overview of Thesis

The structure of this thesis is structured as follows:

- **Chapter 2 - Background and related work**. The key concepts in the areas of this research are reviewed to present the reader the problems and approaches of this thesis. The chapter begins with a presentation of Portfolio Optimization and Time Series Analysis used in the econometrics literature, including well-known models traditionally used in finance. It then presents the theory of several Machine Learning regression models that can be used as an alternative to the traditional models. The chapter ends with a presentation on linear and nonlinear approaches to covariance and correlation analysis in Portfolio Optimization, thus serving as a literature review on what has and hasn't been done in the intersection of portfolio allocation and machine learning.

- **Chapter 3 - Data and Model Selection**. This Chapter starts with a presentation on how the data was acquired and processed prior to analysing it. Then, the Time Series Analysis (TSA) part is introduced, which presents a study on the stationarity, autocorrelation, correlation and returns distributions of the assets, to better understand how the series behave. The Chapter ends with the model selection part, which leverages the insights obtained from the TSA in order to select appropriate models.

- **Chapter 4 - Model Implementation**. The technology used to carry out the research is reviewed to introduce the reader the technology environment. The theory explained in the previous Chapter is implemented and explained, introducing different techniques and libraries used for the purpose. Finally, the training and testing pipeline is clearly presented.

- **Chapter 5 - Testing and results**. This study is the main empirical block of the research. It starts with a brief presentation of the dataset and how cross-validation was used. Then, the three experiments are carried out, with further explanation on the training and testing process, and the different settings and conditions the algorithms were tested on. Finally, an analysis of the results is presented, with a brief discussion of the key takeaways of the experiments.

- **Chapter 6 - Conclusions and Future Work**. The final Chapter provides an overall conclusion of the thesis with a brief summary of the main points and what

can be learnt from the results and experiments. Finally, future directions of work to be done in this area is presented.

Chapter 2

Background and related work

This Chapter introduces the reader the main theoretical topics relevant to the research. It seeks to present the background of the research and the current approaches being carried out by researchers. Firstly, an introduction to Portfolio Optimization is presented, with an emphasis on Modern Portfolio Theory. Then, Time Series models in the econometrics literature are presented, to have a better understanding on traditional modelling used in finance. The next part presents Time Series models from the Machine Learning literature, introducing Linear Regression, Support Vector Regression and Heuristic Optimization methods. Then, a brief presentation on Decentralized Finance is given before presenting the current approaches to covariance analysis in Portfolio Optimization. The Chapter ends with a brief yet important section on Matrix decomposition.

2.1 Portfolio Optimization problem

Mathematical optimization is a branch of mathematics that deals with the maximization or minimization of some function relative to some set of choices in a certain situation. Portfolio optimization deals with this problem in the context of financial assets, and how to select investments in order to optimize a target function. Firstly, the choice of the target function is a complicated task. The risk appetite of investors varies, as one might allow more risk in the strategies with the expectation of generating a potential profit, or vice versa.

Secondly, the portfolio optimization problem vastly depends on the set of constraints specific to the problem. In general, the constraints $\mathcal{C}$ that are assumed are:

1. All weights are larger than or equal than 0.

2. Sum of all weights equals 1.

Finally, some of the most used notations throughout these optimization problems is presented below:

- μ: Vector containing mean of returns of assets.
- σ: Vector containing standard deviation of returns of assets.
- Σ: Covariance matrix of returns of assets.
- ω: Vector containing the weights of the portfolio.

2.1.1 Modern Portfolio Theory

Modern Portfolio Theory (aslo known as Mean-Variance Analysis) was pioneered by Harry Markowitz in 1952 [61] when he proposed an optimisation problem constituting of the selection of the best asset distribution within a portfolio, in order to maximize the expected return, conditionally on any given amount of risk. The main advantage of this method is the diversification in asset selection, as a higher expected return per risk is achieved, rather than when trading a single asset.

Under the Markowitz's model, the yield of the portfolio is calculated as the Expected Return:

$$E[R] = \sum_{i=1}^{N} w_i E[R_i]$$

and the risk as variance:

$$Var[R] = \sum_{i=1}^{N} w_i^2 \sigma_i^2 + \sum_{i=1}^{N} \sum_{j \neq i}^{N} w_i w_j \sigma_i \sigma_j \rho_{ij}$$

where σ is the Standard Deviation of the sample returns on a given asset, ρ_{ij} is the Correlation Coefficient between two assets returns, and w is the Weight of an asset in the portfolio.

Maximum Return portfolio

A popular type of portfolio is the Maximum Return portfolio. As the name indicates, it gives the *maximum Expected Return* at the chosen level of risk. The problem can be written as follows:

$$\begin{aligned} \text{maximize} \quad & E[R] \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i = 1, \\ & w_i \geq 0. \end{aligned}$$

where $E[R]$ is equal to $w^T.\mu$

Minimum Variance portfolio

This type of portfolio consists in minimizing the Variance for a given expected return. This is particularly interesting for risk-averse investors, and can be written as follows:

$$\begin{aligned} \text{minimize} \quad & Var[R] \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i = 1, \\ & w_i \geq 0. \end{aligned}$$

where $Var[R]$ is equal to $w^T.\Sigma.w$.

Mean-Variance portfolio

MPT is based on the combination of the two above-mentioned portfolios. In theory, an investor does not want to only minimize the risk of the portfolio, as the return of the portfolio might not be large. Similarly, an investor does not want to only maximize the returns, as the risk associated with it increases. There are multiple ways one might optimize a combination of these two, such as the following:

When presented with two functions, where one must be maximized and the other one minimized, one can reformulate the problem as the minimization of the weighted difference between the two functions. The weighting component is applied to control the strategy, such that we restrict risk or increase profit. This can be written as follows:

$$\begin{aligned} \text{minimize} \quad & Var[R] - \lambda E[R] \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i = 1, \\ & w_i \geq 0. \end{aligned}$$

where $Var[R]$ is equal to $w^T.\Sigma.w$ and $E[R]$ equals $\lambda.w^T.\mu$

In general, mean-variance frontier estimation is normally done backward-looking, which means that inputs are in-sample computed. In practice, the optimization of a portfolio requires forward-looking inputs and outputs, in order to do well in the future. As expected returns forecasting is usually a hard task, it is worth looking at the covariance matrix estimation, as it can be estimated somewhat more reliably. For this reason, the following section (Section 2.2), focuses on Time Series Analysis, which is essential when investigating assets performance to compute the covariance matrix.

2.2 Time Series Analysis

In mathematics, time-series models are used to forecast events based on past data points indexed in time. These points are spaced evenly, meaning that it is a discrete series. Time-series analysis is done to obtain significant statistics and other attributes of the data, which is essential in financial applications. When applying a model to predict future values of a series of points, it is called time-series forecasting. For this, there exist different models that represent different stochastic processes, and belong to two main classes: Autoregressive (AR) models and Moving Average (MA) models. These two types depend linearly on previous data in the timeseries, and combined form the Autoregressive Moving Average (ARMA) model. Furthermore, an additional two forecasting methods from the Machine Learning literature are presented: Support Vector Regression and Linear Regression.

2.2.1 Autoregressive Models (AR)

Autoregressive models are a form of representation of a random process used to describe time-varying processes such as financial time series. As previously mentioned, the output depends linearly on the previous values of the series plus a stochastic term. This model is also called $AR(p)$, where p indicates the order of the model, and in mathematical terms it can be represented as follows:

$$X_t = c + \sum_{i=1}^{p} \varphi X_{t-i} + \varepsilon_t$$

where c is a constant, $\varphi_1, \varphi_2, ..., \varphi_p$ are the parameters of the model, and ε_t is a random signal with the same intensity at different frequencies, named white noise. For a more in-depth study of this method refer to [100], and for an empirical study on stock market prediction using this method, refer to [89].

2.2.2 Moving Average Models (MA)

In Time Series Analysis, a Moving Average is a computation carried out to analyse data points of a time series by generating a series of averages of different sub-samples of the full time series. It is commonly used to smooth out short-term fluctuations and underline longer-term trends in the time series. In finance, it is used as a stock indicator (heavily used in technical analysis) because it helps mitigate the impact of random short-term fluctuations on stock prices.There are several variations of this method, such as cumulative MA or weighted-forms MA, but only the simple MA technique is described below.

In quantitative finance, the Simple Moving Average (SMA) is the unweighted mean of the previous m datapoints of a time series. For example, let $p_1, p_2, ..., p_k$ be the price series of a financial asset. The last m point's mean (eg: last m seconds, minutes or days) is computed as follows:

$$SMA_k = \frac{p_{k-m+1} + p_{k-m+1} ... + p_k}{m} = \frac{1}{m} \sum_{i=k-m+1}^{k} p_i$$

When computing the next value for SMA_k (denoted SMA'_k) using the same length of datapoints, the datapoints are shifted by 1, and the SMA'_k for the next value is:

$$SMA'_k = \frac{1}{m} \sum_{i=k-m+2}^{k+1} p_i = SMA_k + \frac{1}{m}(p_{k+1} - p_{k-m+1})$$

Examples of how Moving Average methods can be used for Time Series Analysis and in financial applications can be found in [93] and [42].

2.2.3 Autoregressive Moving Average

Autoregressive Moving Average models are used in the statistical analysis of time series to describe stationary stochastic processes, by combining an $AR(p)$ and a $MA(q)$ model. It was first introduced by Peter Whittle in [95].

The ARMA model is a technique to understand and sometimes forecast future values of time series. The first component of the model, the Autoregressive part, deals with regressing the variable on the time series own values (lagged version of it). On the other hand, the Moving Average part, deals with the modelling of the error term, by linearly combining the error terms between them at several times in the past. This model can be computed using the Box-Jenkins method, which is explained in [60]. The mathematical expression of the $ARMA$ model, which is noted as $ARMA(p, q)$, is as follows:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

Where p refers to the autoregressive term and q moving average terms. The error term ε is generally assumed to follow a Gaussian distribution such that: $\varepsilon_t \ N(0, \sigma^2)$, where σ^2 is the variance. Finally, in order to chose the appropiate values of p and q, the Autocorrelation Function and the Partial Autocorrelation Function can be plotted, as explained in Section 3.3.2.

2.3 Linear Regression

Linear Regression is a Machine Learning algorithm based on Supervised Learning that can be used for prediction. It is a linear method to model the relationship between several variables (explanatory) and a scalar response. The simplest case of Linear Regression is when there is only one explanatory variable, which is called Simple Linear Regression. Given a dataset $\{y_i, x_{i1}, x_{i2}, ...x_{ip}\}_{i=1}^{n}$, linear regression assumes a linear relationship between the

regressors vector $\mathbf{x}$ and the dependent variable y. Mathematically, it can be represented as:

$$\mathbf{y}_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i, \text{i} = 1, 2, ..., \text{n}$$

In matrix form, this can be computed as $\mathbf{y} = X\boldsymbol{\beta} + \varepsilon$, where:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_p \end{bmatrix}$$

In this research, Linear regression will use as the exploratory variables past values of the scalar response, thus trying to find a linear relationship between the past values and the future value of the forecasted variable, which will be later specified.

In general, Linear Regression has had applications in a broad range of areas ranging from Finance [10] to epidemiology [16]. It is a simple and attractive method because the representation is simple, and it is accessible to non-technical people, which adds a great deal in businesses. A more in-depth explanation on how Linear Regression can be estimated and further extensions to the original model can be seen in [79] and [64].

2.4 Support Vector Machines - Classification and Regression

Recently, Support Vector Machines (SVM) were successfully used for classification tasks in a broad range of areas[77], [47], [58], [20], ranging from medicine to horse racing predictions. As the focus of this research lies on covariance forecasting, and therefore continuous space values, an extension of Support Vector Machines called Support Vector Regression (SVR) is implemented. However, the main principle of the classification and regression methods remains the same: Learn a function that describes a dataset, and use this function on unseen data to estimate an outcome.

This section is divided in two parts. The first one introduces Support Vector Machines

(SVMs from now on) and clearly differentiates them in two categories: Linear and Nonlinear SVMs. The second part presents Support Vector Regressions, which is the regression extension to SVMs, and is the core algorithm of this research.

2.4.1 Support Vector Machines (SVM)

SVMs classification algorithm was introduced in [92] and derives from Statistical Learning theory (SLT), where the idea of finding an Optimal Separating Hyperplane (OSH) in the input space was discussed. Given a set of training data, each labeled as one of the two classes, a SVM learns a model that designates previously-not-seen data to one class or the other. By maximizing the width of the gap across classes, the number of hyperplanes in the input space is finite, and due to finitness, there must be an optimal hyperplane.

Linear-Support Vector Machines (L-SVM)

Let $S = \{(x_i, y_i)\}_{i=1}^{m} \in \mathbb{R}$ x $\{-1, 1\}$ be a training set. A hyperplane can be defined as a set:

$$H_{\mathbf{w},b} = \mathbf{x} \in \mathbb{R}^d : f(x) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0 = 0$$

parameterized by $\boldsymbol{\beta} \in \mathbb{R}^d$, which refers to a unit vector where $||\boldsymbol{\beta}|| = 1$, and $\beta_0 \in \mathbb{R}$ refers to the distance to the space's origin. The class y_i of the data x_i is designated depending on its position with respect to $H_{\mathbf{w},b}$. In this case, the classification is done according to:

$$y_i = sign(f(\mathbf{x}_i)) = sign(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)$$

This means that $f(x)$ computes the distance of the input data to $H_{\mathbf{w},b}$, and depending on the sign of it (which indicates to what side of the plane the point is), it classifies it as one class or the other.

Linear Separable Input Space

Linear Separable Input Space refers to the input data that can be separated by a linear function of the form:

$$f(x) = \mathbf{x}_i^T \boldsymbol{\beta} + \beta_0$$

where $\forall i$: $y_i f(x_i) > 0$. As there can be more than 1 function that separates the two classes, the SVM looks for the OSH, which maximizes the margin between the training data points from both classes (Figure 2.1 shows a graphical representation of this).This becomes an optimization problem that can be denoted in mathematical form as follows:

$$\max_{\beta,\beta_0,||\beta||=1} M$$

such that:

$$y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M, \text{ i = 1,...,N}$$

where M refers to the distance from the data to the separating line, which is maximized so the OSH is found among the Separating Hyperplanes (SH).This can be further written as:

$$\frac{1}{||\beta||} y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M$$

such that:

$$y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq M||\beta||, \text{ i = 1,...,N}$$

$$y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1, \text{ i = 1,...,N}$$

One can affirm without loss of generality that: $||\beta|| = \frac{1}{M}$, because for every β, β_0 that satisfies the above inequalities, all positively scaled multiples satisfy them as well. Thus, the problem can be written as:

$$\min_{\beta,\beta_0} ||\beta||,$$

such that:

$$y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq 1, \text{ i = 1,...,N}$$

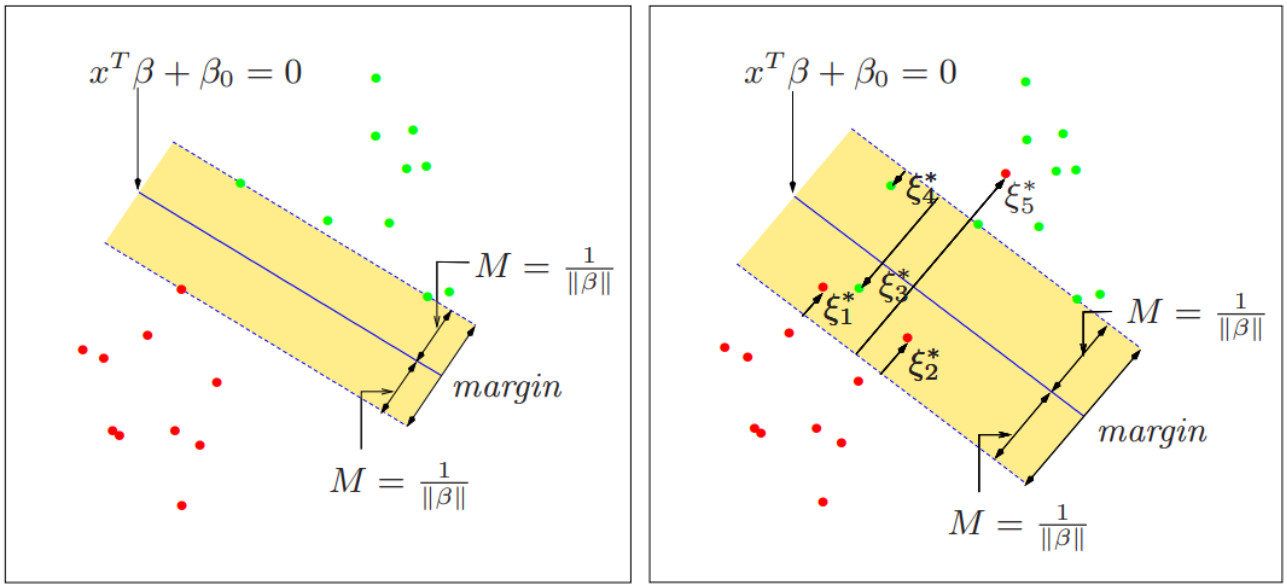


Figure 2.1: Graphical representation of L-SVM in the linear separable case (left) and non-linear separable case (right). Image obtained from [43].

Non-linear Separable Input Space

Figure 2.1 showed that in some cases, the data is not strictly separable, thus the above definition does not solve the classification problem. On the right image on Figure 2.1, some outliers are present, which make impossible to strictly linearly separate the data. In order to tackle this issue, some constraints must be relaxed, in order to allow some misclassification. The problem can be reformulated as:

$$y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \forall i : \xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq c$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2, ..., \xi_N)$ are slacks variables that indicate the relative distance of $\mathbf{x}_i$ to the hyperplane. One can differentiate three cases of ξ_i: Greater than 1, which indicates that it is on the wrong side of the hyperplane (ξ_5 in Figure 2.1), between 0 and 1, which indicates that the data is in the margin zone (ξ_1 in Figure 2.1), or equal to 0, which indicate data lying on the correct side outside the margin. Furthermore, c is defined by the user and is a measure of how much misclassification is allowed.

Rearranging the equations from before, a convex optimization problem is obtained:

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N} \xi_i$$

such that:

$$y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) \geq 1, \text{ i = 1,...,N}$$

$$\xi_i \geq 0, \text{ i = 1,...,N}$$

In this case, the second term in the minimization part controls the amount of misclassifications allowed wrt. their respective classes. The greater the value of C is, the smaller the margin is, so less points are misclassified. On the other hand, the smaller the value of C is, the bigger the margin is, so more points are misclassified.

Convex optimization problems have a characteristic that the local optima must also be the global minima, which is convenient when solving the problem. In this case, Lagrange multipliers can be used to reformulate the problem, which can be re-written as:

$$L = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i(y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) - (1 - \xi_i) - \sum_{i=1}^{N} \mu_i\xi_i$$

where α_i and μ_i are the Lagrange multipliers that need to be maximized, and $\boldsymbol{\beta}$, β_0 and ξ_i are minimized. This can be further expanded as:

$$\boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i = C - \mu_i, \forall i$$

where α_i, ξ_i and $\mu_i \geq 0\ \forall i$. The dual form of this problem can be derived with the help of the previous equations, and is as follows:

$$\max_{0\leq\alpha_i\leq C} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

such that:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Now, in order to find α_i^*, the optimal solutions, KKT Conditions (Karush-Kuhn-Tucker) are proposed:

$$\alpha_i(y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) - (1 - \xi_i)) = 0, \text{ i = 1,...,N}$$

$$\mu_i\xi_i = 0, \text{ i = 1,...,N}$$

$$y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \geq 0, \text{ i = 1,...,N}$$

The KKT Conditions are suited for quadratic programming (QP), and as the problem is a convex optimization problem with linear constraints, it is also equal to the global optimum.

It is considered a Support Vector to all solutions $\alpha_i^* > 0$ that satisfy the first and third KKT conditions. These Support Vectors facilitate the solution of the original convex optimization problem as β can be found as follows:

$$\boldsymbol{\beta}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i$$

In order to find a solution for the last variable, β_0^*, we take all training points that obey: $\xi_i^* = 0$ as well as: $0 < \alpha_i^* < C$. We find that these points are the ones that are exactly on the margin, and thus, the classification can be re-written as follows:

$$y_i = sign(f^*(x')) = sign(\mathbf{x}^T\boldsymbol{\beta}^* + \beta_0^*)$$

Non-Linear-Support Vector Machines (NL-SVM)

The cases that have been covered so far assumed that the input space could be separated by a linear function, which is not always the case in real-world problems. As such, the decision function that was assumed linear, must be changed to the non-linear space. In order to do this, Φ is introduced. This is a feature mapping function that is able to map data from the non-linear separable space into a higher dimensional feature space $\mathcal{H}$, where the data can be linearly separated.

In the BL-SVM case, the dual optimization problem remains the same except the $x_i^T x_j$ terms, which are changed for a kernel function as follows:

$$\max_{0 \leq \alpha_i \leq C} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle$$

such that:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

The Kernel function $\mathcal{K}$, with $\mathcal{K}(\mathbf{x}, \mathbf{x'}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x'}) \rangle$ is introduced to avoid computing $\Phi : \mathbb{R}^d \to \mathcal{H}$. This is the well-known Kernel Trick [45]. This method avoids calculating the dot product in higher dimensions by applying the kernel function on the data and then computing the dot product. One of the constraints that the kernel must obey is that it must be positive-semidefinite, which ensures that it is possible to divide the datapoints in the higher dimensional space. Examples of well-known kernel functions are:

Linear Kernel: $\mathcal{K}(\mathbf{x}, \mathbf{x'}) = \langle \mathbf{x}, \mathbf{x'} \rangle$

Polynomial Kernel: $\mathcal{K}(\mathbf{x}, \mathbf{x'}) = \langle \mathbf{x}, \mathbf{x'} \rangle^p, p \in \mathbb{N}$

Radial Basis Function (RBF) Kernel: $\mathcal{K}(\mathbf{x}, \mathbf{x'}) = exp(-\gamma ||\mathbf{x} - \mathbf{x'}||^2), \gamma \in \mathbb{R}, \gamma > 0$

The linear kernel above corresponds to the linear SVM explained earlier, and the complexity of the algorithm depends on the kernel used and the input space dimensions. For a more in-depth explanation, please refer to [43].

2.4.2 Support Vector Regression (SVR)

Support Vector Regression was first introduced in [85] as a regression extension to the SVM. The idea is very similar to the SVM one: Finding an Optimal Separating Hyperplane (OSH). However, as it is regression and no longer classification, the SVR algorithm seeks to find an OHS that best represents all training datapoints while obeying several

constraints. In order to transform the algorithm to regression, the minimization problem is changed to:

$$\min_{\beta,\beta_0} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i$$

such that:

$$\xi_i \geq 0, \forall i$$

$$y_i(\Phi(\mathbf{x}_i)^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \forall i$$

When solving this optimization problem, there are only two decisive values of ξ_i, which are when it is equal to 0 (referring to no violations), or the biggest allowed value for violations by trespassing the margin on the opposite side is decisive, where $\xi_i = 1 - y_i(\Phi(x_i)^T\boldsymbol{\beta} + \beta_0)$. In this case, the constraint is simplified as: $\xi_i = max(0, 1 - y_i(\Phi(x_i)^T\boldsymbol{\beta} + \beta_0))$ and the problem can be reformulated as:

$$\min_{\beta,\beta_0} \frac{1}{N} max(0, 1 - y_i(\Phi(x_i)^T\boldsymbol{\beta} + \beta_0) + \lambda||\boldsymbol{\beta}||^2$$

where $\lambda = \frac{1}{2NC}$. This equation means that for all datapoints located on the right-hand side of the hyperplane, it returns 0.

When defining the SVM with the Hinge Loss:

$$\mathcal{L}_h(w, \tau) = max(0, 1 - w\tau)$$

One can change the variables w and τ for y_i and $\Phi(x_i)^T\boldsymbol{\beta} + \beta_0$ and rearrange the minimization problem as:

$$\min_{\beta,\beta_0} \frac{1}{N}\mathcal{L}_h(y_i, \Phi(x_i)^T\boldsymbol{\beta} + \beta_0) + \lambda||\boldsymbol{\beta}||^2$$

Furthermore, another loss function is introduced in [85]:

$$\mathcal{L}_\epsilon(w, \tau) = max(0, |w - \tau| - \epsilon), \epsilon \ \in \mathbb{R}, \epsilon > 0$$

This is called the ϵ-insensitive loss function, and can be graphically seen in Figure 2.2.

It reduces the chances of overfitting the data when the value of ϵ is chosen correctly.

Lastly, the SVR problem can be formulated as follows:

$$\min_{\beta,\beta_0} \frac{1}{N}\mathcal{L}_\epsilon(y_i, \Phi(x_i)^T\boldsymbol{\beta} + \beta_0) + \lambda||\boldsymbol{\beta}||^2$$

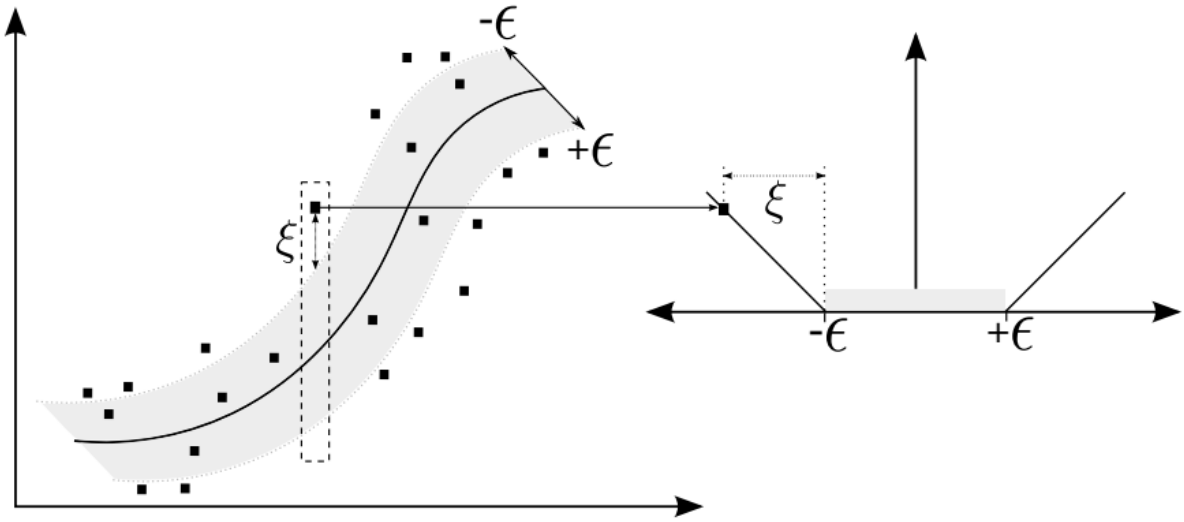


Figure 2.2: Graphical representation of a Support Vector Regression with soft margin loss setting. Image obtained from [96].

2.5 Hyperparameter Optimization

As it has been seen above, the Support Vector Regression, and the vast majority of Machine Learning (ML) algorithms, require some sort of user-defined hyperparameters. This hyperparameters values are used to control the learning process of the ML algorithm, and must be chosen carefully in order to optimally solve the problem. Hyperparameter Optimization consists in finding a combination of hyperparameter values that minimizes the loss function given some data.

There are multiple approaches to Hyperparameter Optimization, ranging from more traditional ways such as Grid search to more sophisticated ones such as Evolutionary Optimization. The latter was first applied to SVM hyperparameter tuning [36], and have proven to be effective in multiple other domains [59], [44], [35].

The most popular technique for Evolutionary Algorithms is Genetic Algorithms (GA). These are commonly used for creating high-quality solutions to optimization problems [63], and have previously been used with Support Vector Regression in order to optimize its hyperparameters [98], [13]. Some of the benefits of using GA's is that they use an objective function, so derivatives might not exist and it would still work (as opposed to Gradient methods), they are well suited for multi-criteria optimization (Gradient methods are mono-criteria), they can handle any 'cost' function with complex constraints and rules, and are robust in finding near-optimum solutions. For these reasons, they have been chosen as the optimisers in addition to the innovative aspect of combining GA's for SVR hyperparameter optimization in the context of crypto covariance matrix forecasting.

2.5.1 Genetic Algorithms

Genetic algorithms are a series of heuristics motivated by nature that fits into the larger evolutionary algorithms category (EA). They are vastly used in optimization and search problems to generate a high-quality solution when no "good" deterministic method exists. The main advantage of Genetic Algorithms (GA) is that all details of the problem don't need to be known nor specified. A "fitness function" is used to rank possible solutions. Then, an evolutionary procedure is followed to devise another possible solution, which is evaluated and ranked. The mutation function is in charge of adding some noise to a candidate to find a better solution. The idea is to merge two possible good solutions hoping to come up with a better solution to the problem. There are multiple methods to choose what members will be used to construct the next generation.

As seen, the search for a better solution is not random, as it uses some previous knowledge of "good" solutions to do an "oriented random search". As one can imagine, GA's do not guarantee the best solution, but in cases where it is not trivial to find an optimal solution, GA's tend to outperform other methods [34], [94].

The two central steps in GA's are:

- **Mutate and Reproduce**: This step consists in describing the problem in a way that solutions can be effortlessly coded and modified automatically. This is called "genetic representation", and is a way of encoding genetic representations (such as appearance, physical qualities, colours) in an evolvable and expressive way.

- **Fitness Function**: This step consists in defining a function that will be in charge of evaluating and ranking solutions. This way, it will guide simulations towards optimal solutions by favoring the best-performing solutions.

The pseudo-code for the classic Genetic Algorithm is:

Algorithm 1 Genetic Algorithm (GA)

0: $\beta \leftarrow$ rate of **crossover**
0: $\alpha \leftarrow$ rate of **mutation**
0: Generations $\leftarrow 0$
0: **while** Termination criterion is not satisfied **do**
0: Generations = Generations + 1
0: **Select** good chromosomes
0: Do **crossover** with Probability β
0: Do **mutation** with Probability α
0: **Evaluate** new population
0: **end while**
0: =0

Below is a list of the Genetic operators introduced in Algorithm 1 and their characteristics:

- **Select**: This operator deals with the selection process of excellent chromosomes to reproduce, based on the fitness function. This function is used to rank solutions, and the ones that score the highest have a higher probability of yield offspring in the next generation, chosen via techniques such as the tournament method. These solutions are then put in a 'mating pool', and are called parents. Next, a pair of chromosomes are chosen to crossover, and they then blend with the rest of chromosomes, creating a new population.

- **Crossover**: This operator is done with probability β in order to substitute genes between 2 chromosomes. Given two chromosomes $C_1 = \{c_{11}, c_{12}, ..., c_{1n}\}$ and $C_2 = \{c_{21}, c_{22}, ..., c_{2n}\}$, chose a number (integer) in the range $0 \leq k \leq n$, C_3, C_4 are offspring of the crossover operation between C_1 and C_2:

$$C_3 = \{c_i | \text{ if i} \leq k, c_i \in C_1, \text{else } c_i \in C_2\}$$

$$C_4 = \{c_i | \text{ if i} \leq k, c_i \in C_2, \text{else } c_i \in C_1\}$$

- **Mutation**: This operator decides if a chromosome should mutate to the next generation and it is right after the crossover operation. Given $C_1 = \{c_{11}, c_{12}, ..., c_{1n}\}$, chose a number (integer) in the range $0 \leq k \leq n$, such that C_3 is a mutation of C_1, and is defined as:

$$C_3 = \{c_i | \text{ if i} \neq k, \text{ then } c_i = c_{1i}, \text{else } c_i = \text{ random}(c_{1i}) \ \}$$

  Then, the population is substituted by the offspring, and a new population is formed. This cycle is repeated until the stop condition is met.

2.6 Decentralized Finance

Decentralised Finance (DeFi) is a financial ecosystem built on blockchain technology. One of the main characteristics of this system is that, as opposed to Centralized Finance (CeFi), there is no need for an intermediary entity like a bank, exchange or brokerage when doing an operation. That is, the users are the ones in charge of lending/borrowing funds from others, trading cryptocurrencies, or investing in other financial instruments without a financial entity doing it for them.

DeFi offers several benefits with respect to CeFi (traditional finance). One of them is the transparency of the products available on the market and how easy it is for users to see the rules by which these function. The second benefit would be the accessibility, in other words, how easy it is to create products that operate in this ecosystem. Only a connection to the internet and DeFi-assets knowledge are required to deploy a product in the DeFi space. Thirdly, the control DeFi gives to the users means that investors have full custody of their assets, and no one else can trade them without their consent.

The recent surge of interest of investors in DeFi products [4], such as cryptocurrencies, has sparked our interest in carrying out an analysis on their returns series dynamics and how they correlate with each other. As cryptocurrencies are a reasonably new product, no in-depth Time Series Analysis has been made on these assets, thus creating a gap in the literature. For these reasons, one of this research's focus is on analysing the properties

of the digital assets returns, which are of great interest to financial institutions that want to include cryptocurrencies in their portfolio, such as Asset Managers. Furthermore, the predictive analysis carried out in Chapter 5 studies the cryptocurrencies' covariance within a portfolio and tries to predict how the assets move with respect to each other, thus being beneficial for risk management practices, such as forecasting portfolio risk.

2.7 Covariance and correlation analysis for Portfolio Optimization

Portfolio Optimization is the process of choosing a distribution of assets that maximizes or minimizes an objective. In general, we tend to maximize returns and minimize financial risk. Covariance matrices of financial returns are essential in numerous areas of finance, like risk management and asset allocation. In the classical Markowitz's optimization problem, the matrix seeks to provide a way for the structure of a portfolio such that risk (standard deviation) is minimized for a particular amount of expected return. It is, thus, an essential task in portfolio theory to have a reliable estimate of the covariance matrix, which is obtained from the return series of the assets.

However, the return samples (T) have some inevitable errors, which leads to some other errors in the covariance matrix forecast. When the number of assets (N) increases, the source of error becomes larger and larger, to a point where one of the main properties of the matrix, positive definiteness, is lost. This is a long-standing problem in the literature [5], [12], [65].

In the next section, several approaches for estimating covariance matrices will be presented. The section is divided into two parts: linear and nonlinear models. The most frequent way of tackling this forecasting problem has been done with linear models, as it will be shown. However, more recently, nonlinear models have become popular, as it has been seen that the relationship between two asset returns can be nonlinear.

2.7.1 Linear approaches

There is a common consensus that forecasting returns is a difficult task. On the other hand, methods exist that quickly compute the covariance matrix, for example, using a

rolling sample covariance matrix. A popular approach is to use a monthly rolling n-year covariance matrix, where n ranges from five to twenty years, as an estimating method of the future matrix [23], [14], [24]. However, this model assumes that the covariance matrix is constant over time or varies slowly, depending on the n chosen, which might not represent the actual matrix. Indeed, [101] provides an in-depth study on the dynamics of financial asset returns and how they exhibit heteroskedasticity, indicating that the covariance matrix is not constant.

The traditional approach (sample covariance matrix [81]) is rarely used because it lacks structure. When the number of stocks (N) equals the number of historical returns (T), the number of parameters to estimate is equal to the size of the data, which is problematic. Firstly, the inverse does not exist because the matrix is not full-rank. If it exists, the expected value of the inverse is biased towards the theoretical inverse. For this reason, a popular approach in the literature to estimate the covariance matrix of a portfolio is introduced in [53]. This technique is called shrinkage estimation, which is helpful in decreasing the estimation error of the sample matrix. This flexible method imposes some structure when the estimation problem becomes significantly large. The method shrinks the unbiased (but very variable) sample covariance matrix towards a more biased but less variable single-index model matrix, thus improving the estimating accuracy. One of the main advantages of this technique is that the matrix is invertible and well-conditioned, which is useful when computing the inverse of the matrix.

Factor models have been widely used in mathematical finance to describe assets returns as a linear combination of factor returns weighted by the exposure to that factor [6], [51], [75]. The most popular factor model is the Capital Asset Pricing Model (CAPM) introduced in [29], which describes the relationship between systematic risk and the return of assets. The variable "Beta" in the model oversees capturing the stock's risk (relative volatility of a stock with respect to the market). The Fama-French three-factor model [28] is an asset pricing model built on top of the CAPM and includes size risk and value risk factors to the market risk factor in CAPM. However, the estimation errors of the covariance matrix in factor models with respect to the dimensionality of the problem are poorly understood. In [30], this issue is studied, and it is concluded that the impact of dimensionality in the covariance matrix forecasting is severe in factor models, thus encouraging to take care of this or not to use factor models when large dimensions. For

further comparison of different factor models used in the literature, [3] proposes an in-depth analysis.

As previously mentioned, the sample covariance matrix estimation is not an ideal technique to calculate this matrix, partly because it suffers from the curse of dimensionality. Furthermore, with a small number of financial assets, [70] showed that portfolios that used the BEKK-GARCH model for covariance matrix estimation outperformed the sample covariance matrix. When estimating large covariance matrices, [101] focuses on recent multivariate GARCH models, such as GO-GARCH and DCC-GARCH. Generalized Orthogonal GARCH (GO-GARCH) [90] model represents a multivariate generalization of a univariate GARCH model. This model uses a GARCH(1,1) to model the independent factors used to compute the vector of random disturbances ϵ combined with a matrix that links the unobserved components with the observed variables. The Dynamic Conditional Correlation GARCH (DCC-GARCH) model decomposes the covariance matrix into the standard deviation and correlation matrix. The elements of the first matrix are modelled as a univariate GARCH(1,1) process, and the latter matrix is created with Pearson's coefficient to measure the linear correlation between assets. They conclude that their multivariate GARCH models perform 50% better in terms of accuracy when forecasting the covariance matrix, with DCC-GARCH being marginally better than GO-GARCH.

2.7.2 Nonlinear approaches

As previously explained, the asset allocation problem has been mainly tackled by linear models. While linear models have proven to be effective tools for analysing portfolios, the linear assumption on the relationship between different factors, such as asset returns, or volatilities, is quite limiting. Numerous empirical analyses have indicated that financial time series can be subject to nonlinear dynamics. Indeed, whether it is equities, commodities or foreign exchange, there is strong evidence that this type of relationship between different factors exists.

For example, in [62], the nonlinear and chaotic nature of energy futures is studied. The daily futures returns from 1990 to 2005 for Natural Gas, Unleaded Gasoline and Light Crude Oil are considered. After several statistical tests, they conclude that nonlinearity in the returns cannot be rejected. Using genetic algorithms (GA's) for each future price process, they estimate a potential motion equation, which offers a lower forecasting error than well-established stochastic models. Similarly, in [56], Kian-Ping Lim et al. use dif-

ferent nonlinearity tests in returns time series of emerging stocks markets. After removing the linear correlations between the different returns, all tests indicate that the remaining time series contains predictable nonlinear correlations. They finally encouraged to explore more sophisticated nonlinear time series models to capture this dynamic.

The correlation of various assets in a portfolio represents the level of relationship between their return movements. More recently, the nonlinear correlations have been a focus of interest for many researchers, who try to explain better what drives price movements and how covariance matrices evolve with time. In [32], a Support Vector Regression (SVR) model is proposed for the dynamic modelling and estimating of covariance matrices of the most heavily traded currency pairs in the forex market. Using the daily low and high prices of these currency pairs, the model can forecast more accurately than the benchmark model. It also shows that this method is particularly useful in turbulent periods when forecasting matters most. In general, due to the ability to capture the dynamic and nonlinear behaviour of financial time series, SVR has shown superior results in volatility forecasting, as seen in [68], [17], and [76]. Further research in SVR for financial time series forecasting is encouraged in the direction of exploring different kernel functions and hyperparameters.

Tree-based algorithms have also been used in the context of covariance forecasting. In [9], Random Forest Regression (RFR), Gradient Boosting Regression Trees (GBRT) and Extreme Gradient Boosting (EGB) with a tree learner is applied to the forex market. The major currency pairs' daily low and high prices are used for this matter. It is shown that the Machine Learning tree-based models outperform the Dynamic Conditional Correlation model. Furthermore, they conclude that the best performing model is the GBRT, and they encourage further research in the selection of different loss functions and the analysis of other variants to the random forest and gradient boosting.

As previously presented, Principal Component Analysis (PCA) has been vastly used in covariance matrix analysis. An extension of PCA is Kernel Principal Component Analysis (KPCA), which leverages techniques of kernel methods to introduce nonlinear components. A direct KPCA approach to Portfolio Allocation is presented in [67], where the Gaussian and Polynomial Kernels are tested to introduce different degrees of nonlinearity to the covariance matrix. They test this method in daily data from seven different markets worldwide for a period of 15 years. They conclude that in some cases, the kernel method

performed better than their linear benchmark, but it accomplished a higher error in other cases. They indicate that using kernels boosts the complexity of the model and does not always achieve better performance. They encourage to explore different kernel functions that better fit the dynamics of the data.

Artificial Neural Networks (ANN) have made disruptive advances in many areas, including financial time series forecasting [80]. In terms of correlation and covariance matrices studies, [31] proposes an end-to-end model based on Convolutional Neural Network(CNN) and Convolutional LSTM (Long Short-Term Memory) for covariance matrix forecasting. The model is able to learn nonlinear correlations between the assets by focusing on local structures and spatiotemporal correlations. One of the major advantages of this model is that it does not assume any distributional or structural assumptions about the datasets.

As seen previously, models such as ARIMA are well suited for capturing linear correlations between time series. On the other hand, ANN has proven to be effective when describing nonlinear relationships. A natural next step is to merge two models to capture all types of co-movements between time series. A hybrid ARIMA and neural network model is proposed in [102] that takes advantage of the unique strengths of each of the techniques in linear and nonlinear modelling with the goal of forecasting exchange rate data. They decided to use these models because there has been theoretical and empirical evidence that dissimilar models will have lower generalisation variance or error [69], [39]. The proposed technique consists of two steps: First, the ARIMA model is used to analyse the linear part of the problem, and then, the residuals are modelled using the ANNs. The logic behind this is that the ARIMA will capture the linearities in the data, and the residuals will contain the nonlinearities, which are left to the neural network to model. A similar hybrid model directly used in portfolio optimisation is introduced in [19], where the task is to estimate the price correlation of two assets. The same approach as the previous example is taken, but in this case, a Recurrent Neural Network, in this case, an LSTM, is implemented, as they have proven good performance in sequential data. The results they present are far superior to other financial models. The MSE, RMSE and MAE are computed, and the ARIMA-LSTM model outperforms the best-performing financial model (Constant Correlation model) by a large amount. They finally encourage to try this model in datasets that include black swans, such as the 2008 crisis.

As mentioned in Section 2.5.1, Genetic Algorithms are useful when searching for a solution to an optimisation problem. Indeed it has been seen that they perform an "oriented random search" that leads to a "good" solution to a problem, where all the details of the problem do not even have to be known nor specified. On the other hand, we have seen previously that Support Vector Regression papers use different hyperparameters hoping to capture the dynamics of financial time series accurately. In some cases, these hyperparameters, like the kernel selection, are not studied, analysed or optimised, and researchers hope that the hardcoded hyperparameters perform well.

As Genetic Algorithms provide a method to overcome the downfall of Support Vector Regression, there has been multiple research proposing the combination of GAs and SVRs in the context of financial time series. In [99], Least Square Support Vector Regression (LSSVR) with Genetic Algorithm is introduced. The reformulation of SVR to LSSVR is used due to the higher stability and speed while training the algorithm, partly because LSSVR uses least square loss function instead of a ϵ-intensive loss function. The GA-LSSVR model uses the GA part to dynamically optimise several parameters of the LSSVR by implementing an evolutionary process with a randomly generated initial population of chromosomes, and the regression part then forecasts using two candidate values. This way, the model simultaneously computes the optimal parameters for the optimisation of the LSSVR. The Mean Squared Error is used as the fitness function. This model is applied to the estimation of the beta in the Capital Asset Pricing Model (CAPM), which is the most important factor in measuring risk. As this parameter is not stable, the estimation of it is a difficult task. They conclude that their model obtains better predictive accuracy than an Artificial Neural Network method.

Another example of Support Vector Regression combined with Genetic Algorithms is found in [76]. In this case, they argue that SVR's performance vastly depends on the hyperparameters, and they propose to combine SVR and GA to overcome this dependency. Their novel approach is used to forecast volatility and the evolutionary algorithm dynamically refines the kernel type and two other constants related to this. When the optimised values are computed, they are used to predict volatility. The evolutionary process begins with an initial population of chromosomes generated randomly. Each chromosome consists of two parts, the first one that is a single integer (0, 1 or 2) indicating the type of kernel(Linear, Polynomial or Gaussian) and the other part that indicates the values of

the parameters (eg: C indicating the influence of higher or lower order polynomials and d indicating the degree of the polynomial when using Polynomial Kernel). In each iteration, indicating a generation, the selective method used was the Boltzmann selection method, among others from the GA literature. The paper concludes that their Machine Learning model succeeds in forecasting the volatility of several datasets and outperforms other models such as GARCH(1,1) and the SVR model with brute force counterpart.

Overall, the literature shows that Support Vector Regression performs well when forecasting time series and captures well the nonlinearities. SVRs provide a flexible framework that allows the user to input the amount of error acceptable in the model while the appropriate hyperplane fits the data. However, the hyperparameters contribute vastly to the performance of the models, and as we have seen, many researchers encourage further exploration of hyperparameter-tuning in future work. To overcome this problem, evolutionary algorithms such as Genetic Algorithms have been proposed in the literature. Support Vector Regression methods, in combination with Genetic Algorithms (GA), have presented interesting results that are worth further exploring. The flexibility introduced by GAs when looking for an optimal solution is one of the main advantages of this method, where all details of the problem do not need to be specified. Several financial applications of SVR combined with GA's have been presented in-depth, such as volatility forecasting and beta estimation in the CAPM model. Other applications such as bankruptcy prediction, portfolio optimization with classification and intelligent trading can be found in [40], [103] and [97].

2.8 Matrix decomposition

In mathematics, more precisely in linear algebra, matrix decomposition refers to the process of factorizing matrices in products of matrices. It is a way of reducing a matrix into its constituent parts, therefore it simplifies the operations done on complex matrices by applying them directly to the decomposed element.

In this research, the main goal is to do a regression analysis on the covariance matrix of a portfolio in order to predict future values of the matrix. It is thus beneficial to be able to decompose the original portfolio's covariance matrix into a simplified one, as computing power is saved by operating on the decomposed version.

There exist different methods for decomposing matrices, and can be separated into different groups:

- Solving systems of linear equations related methods:
  - LU Decomposition
  - Rank factorization Decomposition
  - Cholesky Decomposition
- Eigenvalues-related decomposition:
  - Eigen Decomposition
  - Schur Decomposition
  - Singular-Value Decomposition

2.8.1 Cholesky Decomposition

Cholesky decomposition is a factorization of a Positive Definite (PD) matrix into the product of a lower triangle matrix and its conjugate transpose. In the case that the matrix contains only real numbers, like in our case, the factorization is simply the product of a lower triangle matrix and its transpose. This method was selected for the purpose of this research for several reasons. Firstly, the covariance matrix of a portfolio is always square and usually Positive Definite (like in this case), which are two of the constraints of Cholesky method. Furthermore, it is approximately twice faster than other methods such as LU Decomposition for solving systems of linear equations [71], which is a huge advantage considering that covariance matrices can be very large.

The Cholesky algorithm can be thought as a modified version of Gaussian elimination, and it's as follows:

For i = 1 to n:

$$\mathbf{A}^{(1)} = \mathbf{A}$$

Taking into account that $\mathbf{I}_{i-1}$ is the Identity Matrix of dimensions $= i - 1$, at step i:

$$\mathbf{A}^{(i)} = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & a_{i,i} & \mathbf{b}_i^* \\ 0 & \mathbf{b}_i & \mathbf{B}^{(i)} \end{pmatrix}$$

The definition of $\mathbf{L}_i$ is defined as:

$$\mathbf{L}_i = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & \sqrt{a_{i,i}} & 0 \\ 0 & \frac{1}{\sqrt{a_{i,i}}}\mathbf{b}_i & \mathbf{I}_{n-i} \end{pmatrix}$$

Now, $\mathbf{A}^{(i)}$ can be re-written as $\mathbf{A}^{(i)} = \mathbf{L}_i \mathbf{A}^{(i+1)} \mathbf{L}_i^*$ because $a_{i,i} > 0$ since $\mathbf{A}^{(i)}$ is PD and $\mathbf{A}^{(i+1)}$ is defined as:

$$\mathbf{A}^{(i+1)} = \begin{pmatrix} \mathbf{I}_{i-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbf{B}^{(i)} - \frac{1}{a_{i,i}}\mathbf{b}_i \mathbf{b}_i^* \end{pmatrix}$$

After the n steps have been completed, the matrix $\mathbf{A}^{(n+1)} = \mathbf{I}$ and $\mathbf{L}$ is computed as follows:

$$\mathbf{L} = \mathbf{L}_1 \mathbf{L}_2, ..., \mathbf{L}_n$$

Chapter 3

Data and Model Selection

This study explores the covariance matrix of a portfolio of cryptocurrencies and how price movements in one asset correlate with other assets. The study takes into account daily data from each asset while proposing a variety of time series models. In order to explore the covariance matrix, we start by analysing the data, from the data acquisition to the pre-processing. Once the data is cleaned and manipulable, we proceed to have a better understanding of the returns, log returns and their distributions, with particular attention to how well a Gaussian Distribution describes the returns and if there are any fat tails in the plots. Several graphs are presented, such as Quantile-Quantile plots, Box-plots and Empirical Cumulative Distribution Function (ECDF) plots. After analysing each asset individually, we analyse pairs of assets to see if they have linear and nonlinear correlations. Several correlation tests are performed, such as Pearson, Spearman, Distance, and Maximal Information Coefficient (MIC), to see how asset prices move with respect to each other. Finally, the model selection is discussed and presented, and the model workflow is explained.

3.1 Introduction

Data exploration and analysis are crucial when selecting a model that tries to capture the dynamics of the time series data. Financial time series analysis is concerned with the theory and practice of asset valuation over time. This area is highly empirical, but like every other science, theory shapes the basis for making inferences. One of the most distinguishable features of financial time series, which is not observed in other time series analysis, is the presence of an element of randomness in theory and practice, for example, volatility.

For this reason, statistical methods are fundamental in financial time series analysis.

In this section, we explore the price/return series and interactions between assets and choose the model based on this. One of the main factors when deciding what model best describes the data is the nonlinearities between the assets and the data distribution, which will be seen in the following sections.

3.2 Data

The analysis is conducted in a universe of different cryptocurrencies with their daily data for the time interval between 1st September 2014 and 1st September 2022. The cryptocurrencies are all traded against the US Dollar, as it is the most traded currency.

This section is subdivided into two areas: Data Acquisition and Data Pre-processing. The first one presents how the data was obtained and from what source. The other one shows how the data was manipulated to enhance the analysis's performance. Overall, data acquisition and cleaning is a crucial first step in any research that uses data to analyse something, as it ensures the data is reliable and ready to use for one's research.

3.2.1 Data Acquisition

The Python library used to fetch the data from Yahoo's publicly available APIs is 'yfinance' [1]. This library offers a threaded and Pythonic way to download market data and contains over 9000 cryptocurrency pairs. The reasons for using this particular library are its impressive range of data, no fees to access data, quick and easy to set up, and its popularity.

The market data downloaded from the 'yfinance' library for the vast cryptocurrency universe consists of the most prominent cryptocurrency pairs by market capitalisation (or market cap). This is, the total value of all the coins that have been mined, which is calculated by multiplying the number of coins that are circulating by the market price of a single coin. This variable can be observed in Figure 3.1 on the 5th column of each cryptocurrency.

The daily data market data were retrieved from Yahoo Finance by developing a Python script to automate the process of crypto-specific data retrieval. The function in charge of

Symbol	Name	Price (Intraday)	Change	% Change	Market Cap	Volume in Currency (Since 0:00 UTC)	Volume in Currency (24Hr)	Total Volume All Currencies (24Hr)	Circulating Supply
BTC-USD	Bitcoin USD	23,005.88	-323.96	-1.39%	439.652B	28.318B	28.318B	28.318B	19.11M
ETH-USD	Ethereum USD	1,628.95	-43.75	-2.62%	198.407B	20.111B	20.111B	20.111B	121.8M
USDT-USD	Tether USD	1.0002	-0.0000	-0.0013%	66.319B	55.495B	55.495B	55.495B	66.308B
USDC-USD	USD Coin USD	1.0003	+0.0004	+0.0433%	54.480B	8.488B	8.488B	8.488B	54.463B
BNB-USD	Binance Coin USD	285.19	-0.96	-0.33%	46.012B	1.511B	1.511B	1.511B	161.337M
XRP-USD	XRP USD	0.3731	-0.0058	-1.5222%	18.038B	1.217B	1.217B	1.217B	48.343B
BUSD-USD	Binance USD USD	1.0002	+0.0008	+0.0829%	17.925B	6.783B	6.783B	6.783B	17.921B
ADA-USD	Cardano USD	0.5016	-0.0141	-2.7275%	16.931B	702.386M	702.386M	702.386M	33.753B
SOL-USD	Solana USD	41.45	-1.01	-2.39%	14.364B	1.147B	1.147B	1.147B	346.52M
DOGE-USD	Dogecoin USD	0.0674	-0.0016	-2.3683%	8.938B	403.214M	403.214M	403.214M	132.671B
DOT-USD	Polkadot USD	8.02	-0.44	-5.22%	8.844B	666.363M	666.363M	666.363M	1.103B
HEX-USD	HEX USD	0.0463	-0.0022	-4.5185%	8.029B	8.747M	8.747M	8.747M	173.411B
DAI-USD	Dai USD	1.0005	+0.0011	+0.1120%	7.425B	877.871M	877.871M	877.871M	7.422B

Figure 3.1: List of first 13 cryptocurrencies by market cap as per 'Yahoo.Finance.com' on 2nd October 2022 at 17:00 BST. Rows represent different cryptocurrencies traded against the US Dollar, and columns include the name, intraday price, change, percentage change, market cap, different volumes and the circulating supply.

fetching the data takes as input a list of desired indices names (called Symbols as per Yahoo Finance), retrieves the information, and for each pair, it returns the OHLC and volume daily data. As the script is run every time we run the entire program, the data fetched is up to the previous day that one is running the program, thus having more data than downloading it once and storing it in the local drive of a computer. The resulting data structure after fetching the data is a list of N elements, indicating the number of cryptocurrencies, where each element consists of a DataFrame consisting of an index and six columns: Date (YYY-MM-DD format), Open, High, Low, Close and Volume.

3.2.2 Data Pre-Processing

The market data pre-processing stage starts by looking at the data and spotting if there exist Null/NaN values or any other outlier that might result in inaccuracies in the experiments. As the Python library used is reliable and efficient, only two NaN values were spotted for a cryptocurrency (AXS-USD), which was overcome by filling the space with the next price in time.

Figure 3.2: Plot showing Bitcoin's closing price in US Dollar with respect to time for 9 years, including Moving Averages of 10, 20 and 50 days window sizes.

The next step was to include other indicators in the data, such as different window Moving Averages. As previously mentioned, MA's are used to smooth out short-term fluctuations and highlight longer-term trends and are useful to analyse data. In this case, the $10, 20, 50$ day window MA's were computed, as seen in 3.2. In general, not much data pre-processing was carried out because the Python library 'yfinance' cleaned it beforehand.

3.3 Time Series Analysis

Time Series analysis (TSA) studies the attributes of the response variable with respect to time, the independent variable. In TSA, data points are usually recorded in consistent intervals (e.g., nanoseconds, minutes, days) rather than recording data points at random times. In the context of finance and econometrics, Time Series Analysis is concerned with the theory and practice of asset valuation over time. This area is highly empirical, but like every other science, theory shapes the basis for making inferences. One of the most distinguishable features of Financial Time Series, which is not observed in other TSA, is the presence of an element of randomness in theory and the practice, for example, volatility. For this reason, statistical methods are of great importance in Financial TSA.

The objective of analysing this data is to understand how the dynamics of the data work, what factors affect certain variables at particular points of time and have better

insights into the consequences and insights of the features presented in the dataset. In this study, we focus on the stationarity, autocorrelation and correlation of the data, which will be explained in the following sections. These characteristics are believed to be of great interest when building a portfolio of assets, as they help to identify how assets behave with respect to each other and diversify the portfolio when needed.

3.3.1 Stationarity

Stationary models are an essential class of stochastic models used to describe time series [41]. This model assumes that the process remains in statistical equilibrium with probabilistic properties that do not change over time, with a constant mean and variance. This is of particular interest in quantitative finance, as at the core of every quantitative investment idea is the assumption that there are patterns in the market that have been seen in the past and that could be seen in the future. Furthermore, most of the available methods in econometrics were created based on assuming stationarity.

However, it is well known that stock prices are not stationary [54], as they do not fluctuate around a constant mean. This is one of the main reasons why stock prices are hard to predict. To tackle the nonstationarity problem, researchers model *returns* instead of *prices*, which, given a time series $p_T = (p_1, p_2, ..., p_t)$, can be computed as follows:

$$r_t := \frac{p_t - p_{t-1}}{p_{t-1}} x100$$

It is multiplied by 100 to calculate the %. Returns are usually stationary (see Figure 3.3), and to further prove this, there exist different tests that can be carried out on the dataset, that are presented below.

Augmented Dickey-Fuller (ADF) Test

The Augmented Dickey-Fuller test (ADF test) is a unit root test to see if a time series is stationary or not. In statistics, a unit root is a characteristic of a stochastic process that may cause difficulties when making inferences concerning time series, such as causing non-predictable results. The hypothesis for the ADF test is that there is a unit root (Null hypothesis) - which means that the time series is non-stationary, and the alternative hypothesis is usually stationarity or trend-stationarity. The ADF test informs whether or not a null hypothesis can be rejected. In this case, if the test fails to reject the Null hypothesis,

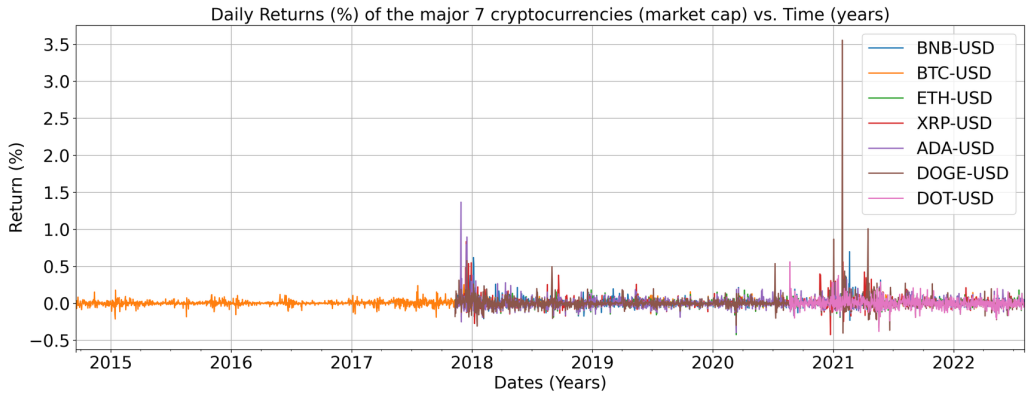


Figure 3.3: Daily returns for the major 7 cryptocurrencies by market capitalisation .

the time series is non-stationary. In order to reject the Null hypothesis, the Test Statistic must be less than the Critical Value, and the p-value must be less than 0.05. For a more mathematical and in-depth explanation of this, refer to [18].

In this research, the ADF test was carried out on every return time series to see if they were stationary. As the universe of assets was large, only a few cryptocurrencies were included in Table 3.1.

Looking at the table, we see that the p-values are smaller than 0.05, and the Test Statistics are all less than the Critical Values. This further confirms that the returns are indeed stationary. Similar results are observed for the rest of the assets, which all reject the Null hypothesis.

Assets(Symbols)	Test statistic	P-Value	C.V. (1 %)	C.V. (5 %)	C.V. (10 %)
BNB-USD	-8.845	1.629 e-14	-3.434	-2.863	-2.568
BTC-USD	-54.664	0.0	-3.433	-2.863	-2.567
ETH-USD	-12.260	9.122 e-23	-3.434	-2.863	-2.568
XRP-USD	-8.836	1.00 e-13	-3.434	-2.863	-2.568
ADA-USD	-7.493	4.439 e-11	-3.434	-2.863	-2.567
DOGE-USD	-7.128	3.602 e-10	-3.433	-2.863	-2.568
DOT-USD	-9.119	3.234 e-15	-3.440	-2.865	-2.569

Table 3.1: Table showing the ADF Test Statistic, P-value, and Critical Values (C.V) at the 1%, 5% and 10% levels for Binance Coin, Bitcoin, Ethereum, Ripple Coin, Cardano, Doge Coin and Polkadot. The numbers are rounded to the third decimal.

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test [52] is also a type of unit root test to see if a time series is stationary or not. Contrary to most unit root tests, like the ADF test, it takes a Null hypothesis that the series is stationary (no unit root) and an Alternate hypothesis that the series is non-stationary (has unit root). In this case, the p-value indicated by the test is the probability score based on which one can decide whether to reject this hypothesis or not. If the p-value is greater than 0.05, one cannot reject the Null hypothesis, and the time series is stationary. Similarly, if the Test statistic is smaller than the Critical Value, the Null hypothesis cannot be rejected, and the time series is stationary. For a more mathematical explanation of this test, refer to [52].

Assets(Symbols)	Test statistic	P-Value	C.V. (1 %)	C.V. (5 %)	C.V. (10 %)
BNB-USD	0.189	0.101	0.347	0.463	0.739
BTC-USD	0.162	0.102	0.347	0.463	0.739
ETH-USD	0.179	0.114	0.347	0.463	0.739
XRP-USD	0.105	0.111	0.347	0.463	0.739
ADA-USD	0.190	0.123	0.347	0.463	0.739
DOGE-USD	0.195	0.102	0.347	0.463	0.739
DOT-USD	0.199	0.104	0.347	0.463	0.739

Table 3.2: Table showing the KPSS Test Statistic, P-value, and Critical Values (C.V) at the 1%, 5% and 10% levels for Binance Coin, Bitcoin, Ethereum, Ripple Coin, Cardano, Doge Coin and Polkadot. The numbers are rounded to the third decimal.

In this study, the KPSS test was carried out on every return time series, in order to see if they were stationary. As the universe of assets was large, only a few cryptocurrencies were included in Table 3.2, similar to the ADF test above.

One can conclude that the return of the assets presented in Table 3.2 are indeed stationary, as they all have a greater p-value than 0.05 and the Test statistic are lower than the Critical Value. This second test also agrees with the previous one, both indicating that the returns are stationary.

3.3.2 Autocorrelation

In financial time series, autocorrelation analysis is essential to measure how much of an impact do past prices have on a security on its future prices and to check for randomness. In the context of portfolio allocation, it can help Portfolio Managers diversify their portfolios and take less risky investments. Analysing autocorrelation is a critical step when doing Data Exploration of time series. It is a mathematical description of the resemblance between a time series and a lagged version of itself over consecutive time intervals. It measures the relationship between the current value of a time series and its past values. The level of resemblance is a value in the range $[-1, 1]$, where -1 means a perfect negative correlation, implying that an increase in a time series results in a decrease in the other. On the other hand, an autocorrelation of 1 means a perfect positive correlation, where an increase in a time series leads to an increase in the other. When the value is 0, it is not resemblant at all.

There are several ways to estimate the autocorrelation of a time series, but in this case, the Autocorrelation function (ACF) and the Partial Autocorrelation function (PACF) are used. These assume stationarity in the data, which we have shown is the case in the present data. The ACF and PACF methods are explained below, and several exciting plots are analysed:

Autocorrelation Function (ACF)

The autocorrelation is usually measured calculating the Pearson correlation between values at different times as a function of the time lag. This test plays an important role for Moving Average (MA) models, as it indicates what "q" term to use in MA(q). A simple procedure for this would be:

1. Estimate the sample mean:
$$\overline{y} = \frac{\sum_{t=1}^{T} y_t}{T}$$
2. Calculate the sample autocorrelation:
$$\rho_i = \frac{\sum_{t=i+1}^{T}(y_t - \overline{y})(y_{t-j} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}$$
3. Estimate variance in order to compute confidence intervals(optional).

In this study, we consider all asset's returns and compute the ACF with the help of the library 'statsmodel' [78], which computes it automatically. We show some of the Autocorrelation plots in Figures 3.4, 3.5, 3.6, 3.7, and later we analyse them.

Partial Autocorrelation Function (PACF)

The Partial Autocorrelation Function (PACF) measures the relationship between a stationary time series and its lagged version, with the effect of a set of controlling stochastic variables removed. It is beneficial in autoregressive modelling in order to find the "p" value in an Autoregressive Model (AR(p)). Given a time series y_t, $\phi_{\mathbf{j,j}}$ denotes the partial autocorrelation of lag $\mathbf{j}$, and can be described as the autocorrelation between y_t and y_{t+j} but without the linear dependence of y_t on y_{t+1} through y_{t+j-1}. In other words, the partial autocorrelation is the autocorrelation between y_t and y_{t+j} that is not taken into account by the lags 1 to $j-1$:

$$\phi_{1,1} = \mathbf{corr}(y_{t+1}, y_t), \text{ for j} = 1$$
$$\phi_{j,j} = \mathbf{corr}(y_{t+j} - \overline{y}_{t+j}, y_t - \overline{y}_t), \text{ for j} \geq 2$$

where $\overline{y}_{t+j}$ and $\overline{y}_t$ are linear combinations (as in this research the time series is stationary, $\overline{y}_{t+j}$ and $\overline{y}_t$ are equal) of $y_{t+1}, y_{t+2}, ..., y_{t+j-1}$ that minimize the MSE (Mean Squared Error) of y_{t+j} and y_t.

A simple procedure for the calculation of the partial autocorrelation of a stationary time series can be done using Durbin-Levinson Algorithm [26]:

$$\phi_{n,n} = \frac{\rho(n) - \sum_{j=1}^{n-1} \phi_{n-1,k}\rho(n-j)}{1 - \sum_{j=1}^{n-1}(\phi_{n-1,k}\rho(k)}$$

where:

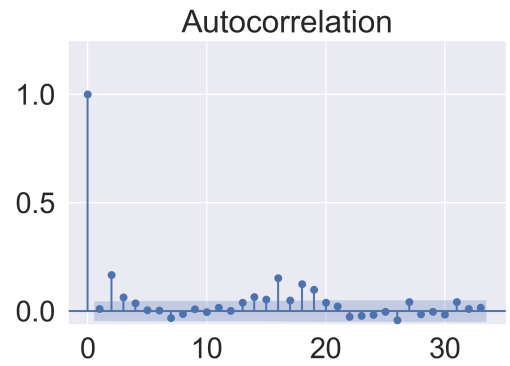
$$\phi_{n,k} = \phi_{n-1,k} - \phi_{n,n}\phi_{n-1,n-k}, \text{ for } 1 \leq \text{k} \leq \text{n - 1}$$

and:

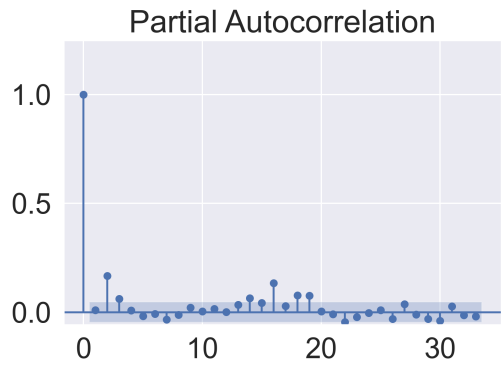
$$\rho(n), \text{ is the autocorrelation presented above.}$$

More details about this method can be found in [72], [25] and [82].

For the purpose of this research, these two methods (ACF and PACF) were carried out on all returns series. As previously mentioned, autocorrelation analysis is essential for any time series prediction task, including portfolio allocation. The plots presented in Figures 3.4, 3.5, 3.6 and 3.7 indicate the correlation coefficient on the vertical axis, with respect to the lag value on the horizontal axis. Additionally, the blue area depicts the 95% confidence interval and means that anything within this area is statistically close to zero, and anything that lies outside is non-zero.



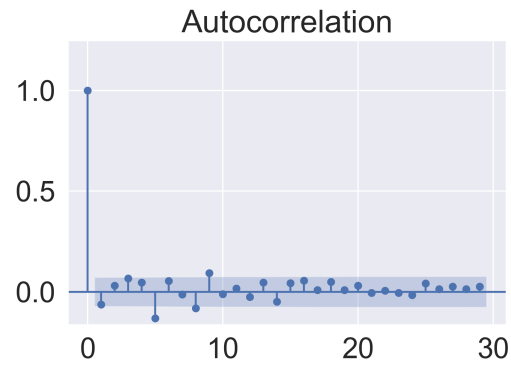
(a) Autocorrelation Function (ACF) for ADA-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.



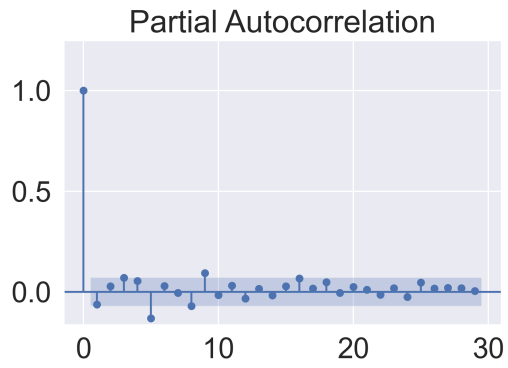
(b) Partial Autocorrelation Function (PACF) for ADA-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.

Figure 3.4: ACF (left) and PACF(right) plots for ADA-USD.

Figure 3.4 shows the autocorrelation (left) and partial autocorrelation (right) for Cardano traded against the US Dollar. As there are several autocorrelations that are outside the confidence interval, this time series is non-random. It can be noted in the ACF plot (left) that there are four points that are significantly outside of the confidence interval, which are at lags 2, 16, 18 and 19, that indicate an autocorrelation coefficient of less than 0.25 or less. By looking at the PACF, we can see that at the same lags, there are significant correlations. As previously mentioned, these numbers describe the order of the $AR(p)$ and $MA(q)$ models.



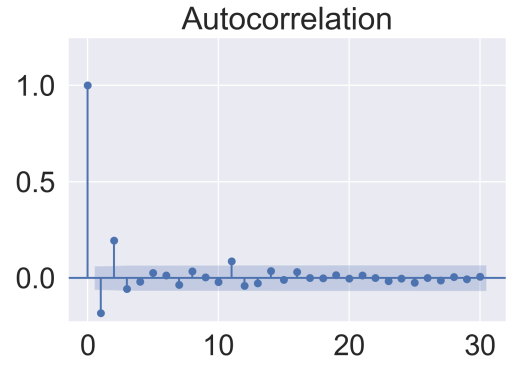
(a) Autocorrelation Function (ACF) for DOT-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.



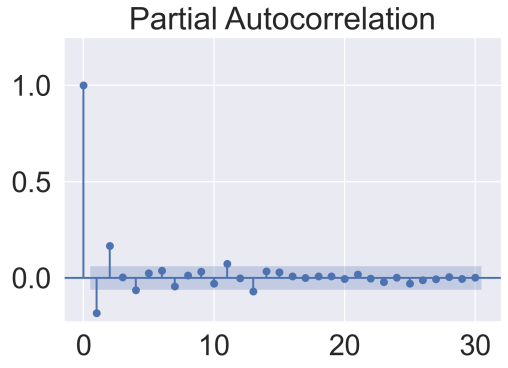
(b) Partial Autocorrelation Function (PACF) for DOT-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.

Figure 3.5: ACF (left) and PACF(right) plots for DOT-USD.

Figure 3.5 shows the ACF and PACF plots for DOT-USD. These graphs indicate that the lag at value 5 is the one that best describes the stationary time series. However, these are around -0.20, which indicates a low negative autocorrelation. Figure 3.6 shows a significant autocorrelation in the two first lags, followed by autocorrelations that are not significant. From this, we can infer that there is a Moving Average term in the data. The number of significant correlations indicates the order of the moving average term, which in this case is two.



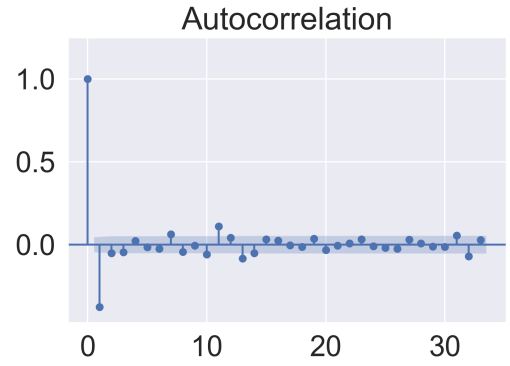
(a) Autocorrelation Function (ACF) for HEX-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.



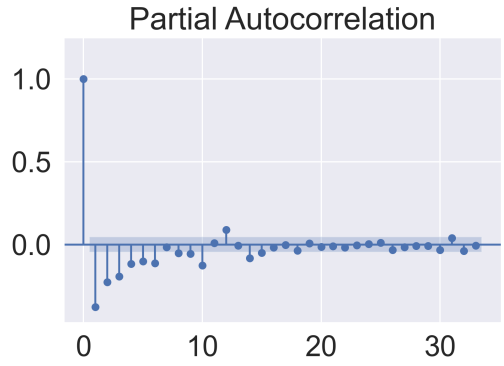
(b) Partial Autocorrelation Function (PACF) for HEX-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.

Figure 3.6: ACF (left) and PACF(right) plots for HEX-USD.

Finally, in Figure 3.7, we see a pattern that differs significantly from the three others. At lag 1 (ACF plot), we see an autocorrelation of around -0.4, and then no significant autocorrelation. In the PACF plot, we see a geometric decay. This pattern indicates that the best model to fit the data would be a $MA(1)$. These plots are done on the USDT-USD coin, which is a stablecoin that is pegged to the US Dollar (1:1 relationship). Taking this into account, the negative autocorrelation at lag 1 means that when the USDT increases in price at time t, the next value at time $t+1$ will decrease, which is exactly the behaviour of a stablecoin, as it must not deviate much from the 1:1 relationship to the US Dollar.



(a) Autocorrelation Function (ACF) for USDT-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.



(b) Partial Autocorrelation Function (PACF) for USDT-USD. Lags are plotted on the horizontal axis, and correlations on the vertical axis.

Figure 3.7: ACF (left) and PACF(right) plots for USDT-USD.

Overall, several plots have been presented and analysed. These plots were selected based on the different patterns that were shown in each one of them, seeking to have a broader understanding of what can be inferred from the ACF/PACF graphs. The plots didn't show a considerably large autocorrelation (more than 50%) in any of the assets, indicating that their lagged versions do not describe the time series accurately. However, this analysis has been insightful in discovering the lack of autocorrelation and the different patterns that can be present.

3.3.3 Correlation

As previously explained, one of the basic principles of Modern Portfolio Theory (MPT) is diversification, which aims to minimize risk and maximize return. This is achieved by selecting particular assets that do not appear to be strongly correlated. However, Portfolio

Allocation methods have normally measured the correlation between assets using Pearson's Correlation, which is only sensitive to linear correlations, while several other methods tend to be more robust for non-linear correlations.

In this section, several methods to measure correlation are presented and analysed, which are:

1. **Pearson's correlation** (linear)
2. **Spearman correlation** (linear and nonlinear)
3. **Distance correlation** (linear and nonlinear)
4. **Maximum Information Coefficient** (linear and nonlinear)

The aim of this is to show the degree of correlation between different assets and see how strongly related each asset is as per each method. Particular attention is paid to the nonlinear correlations, as Machine Learning models have proven efficient in capturing this type of dynamics. A correlation matrix with 30 cryptocurrencies will be presented and analyzed for each method.

Pearson correlation

In statistics, Pearson's Correlation is a way of measuring the linear correlation between two datasets. It can be measured as the covariance of two variable divided by the product of their standard deviations. The result is the Pearson's Correlation Coefficient, which lies in the $[-1, 1]$ range. Mathematically, it can be calculated as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

where:

- r_{xy}: Pearson's Correlation Coefficient
- n: sample size
- x_i and y_i: individual samples

- $\overline{x}$ and $\overline{y}$: sample means for all x and y values

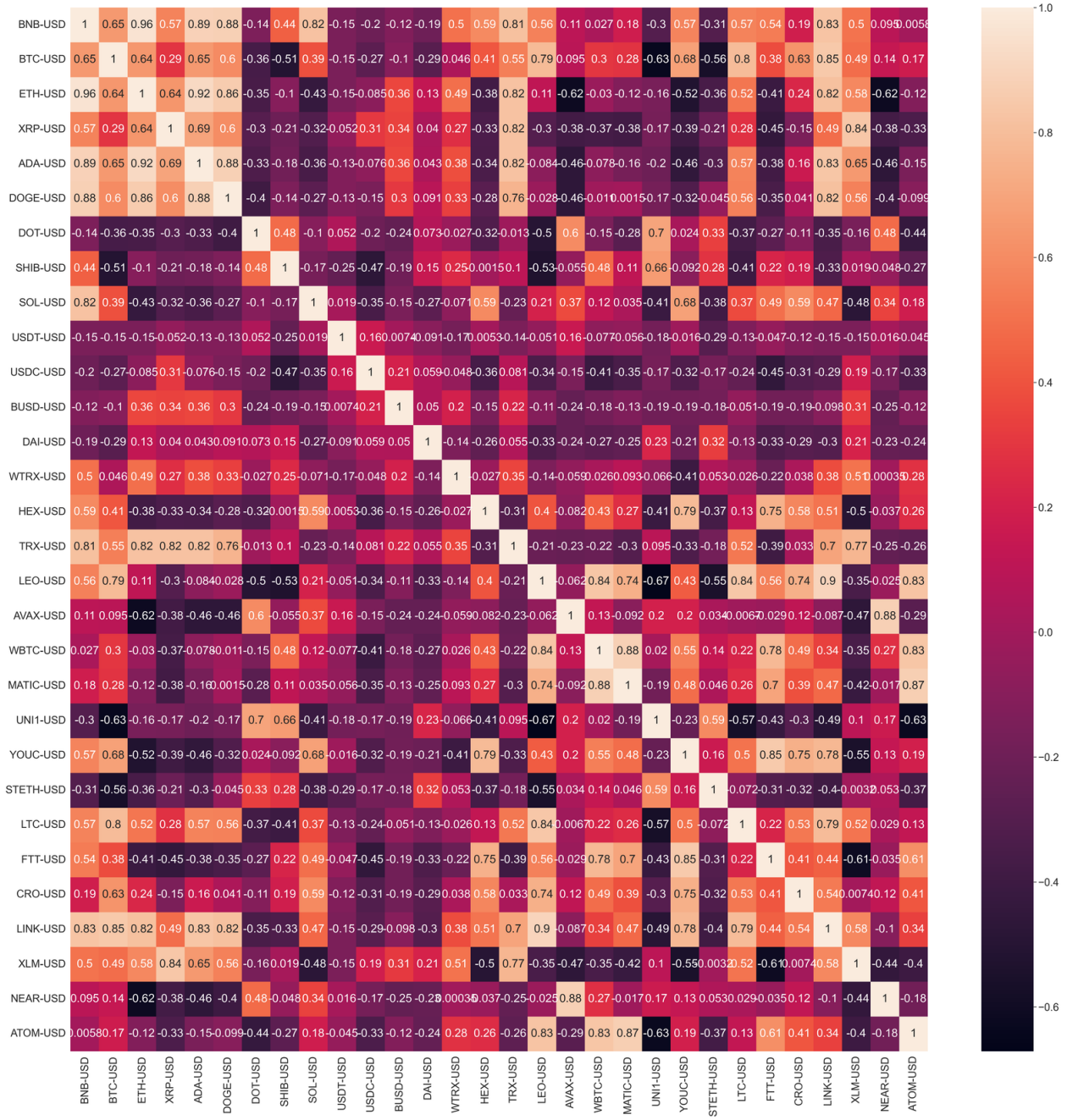


Figure 3.8: Correlation matrix computed with Pearson's Correlation on 30 different crypto assets.

Figure 3.8 shows the Pearson's Correlation matrix of 30 assets of a portfolio. The colours indicate the degree of correlation, ranging from -1 (black) to $+1$ (white). We see that the diagonal is 1 for every entry, which indicates that each asset is perfectly correlated to itself (obviously). The most negatively correlated instruments are LEO-USD and UNI1-USD in this matrix, with a coefficient of -0.67. Taking the diagonal values out, the most positively correlated instruments are ETH-USD and BNB-USD with a coefficient of 0.96, indicating that the returns of these two assets move nearly perfectly in tandem (only linear correlations are taken into account). This matrix has 420 different pairs (as there are 30 assets, the matrix has 900 entries from which we must take into account half of it because the Lower triangle is equal to the Upper triangle, and further subtract 30, which represents the autocorrelations). Of these 420 pairs, exactly half of it has a positive correlation, and the other half is negative, which is convenient for the diversification of a portfolio.

Spearman correlation

In statistics, Spearman's correlation measures the statistical dependence among the rankings of two variables. The coefficient indicates how well a monotonic function can describe the relationship between two variables. This type of measurement is, in fact, equal to Pearson's correlation between the rank values of the two variables; however, Spearman's correlation evaluates linear and nonlinear monotonic relationships. In mathematical terms, Spearman's correlation can be formulated as:

$$r_{s_{xy}} = \frac{cov(Rank(X), Rank(Y)}{\sigma_{R(X)}\sigma_{R(Y))}}$$

where:

- $r_{s_{xy}}$: Spearmans Correlation Coefficient
- n: sample size
- $Rank(X)$ and $Rank(Y)$: individual rank of the variables X and Y
- $\sigma_{R(X)}$ and $\sigma_{R(Y)}$: standard deviations of the rank of X and Y

In the special case where all ranks are distinct integers, the Spearmans correlation can be computed as follows:

$$r_{s_{xy}} = 1 - \frac{6\sum_{i=1}^{n}(d_i^2)}{n(n^2-1)}$$

where:

- d_i: difference between Rank of X_i and Y_i
- n: number of observations

Figure 3.9 shows the Spearman's Correlation matrix of the 30 same assets that were shown in 3.8. Overall, there is a similar pattern in both matrices, with an increase in positively correlated assets of 16. As this type of correlation considers nonlinearities, this might indicate that some of the nonlinear correlations among assets are positive. Comparing the most negative and positive correlated assets from the Pearson's matrix (LEO-USD with UNI1-USD, and ETH-USD with BNB-USD, -0.67 and 0.96, respectively), we see that for LEO-USD and UNI1-USD, the Spearman's correlation indicates -0.77, an increment of 0.1 in negative correlation. For the ETH-USD and BNB-USD correlation, Spearman's coefficient is 0.74, which indicates a decrease of 0.22 with respect to Pearson's. Moreover, when looking at the extreme values, we see that the most negatively correlated assets are FTT-USD with XLM-USD, YOUC-USD with ADA-USD and UNI1-USD with LEO-USD, with coefficients of -0.77. On the other hand, the most positively correlated instruments are ADA-USD with ETH-USD, with a coefficient of 0.96.

An interesting sign change between Pearson's and Spearman's Correlation happens with SOL with DOT and SHIB, respectively. The assets were negatively correlated in the first matrix, with a coefficient of -0.1 between SOL and DOT and -0.17 for SOL and SHIB. When computing the coefficient with Spearman's method, these assets have coefficients of 0.18 and 0.18, respectively, indicating a positive correlation. This finding is interesting as, in the first case, a Portfolio Manager(PM) might want to hedge risk by holding SOL-USD and DOT-USD (or SHIB-USD). On the other hand, a more "aggressive" PM that computes the correlation with Spearman's method might want to buy these combinations of assets to profit from their similar behaviour because it is believed one of the coins will rally.

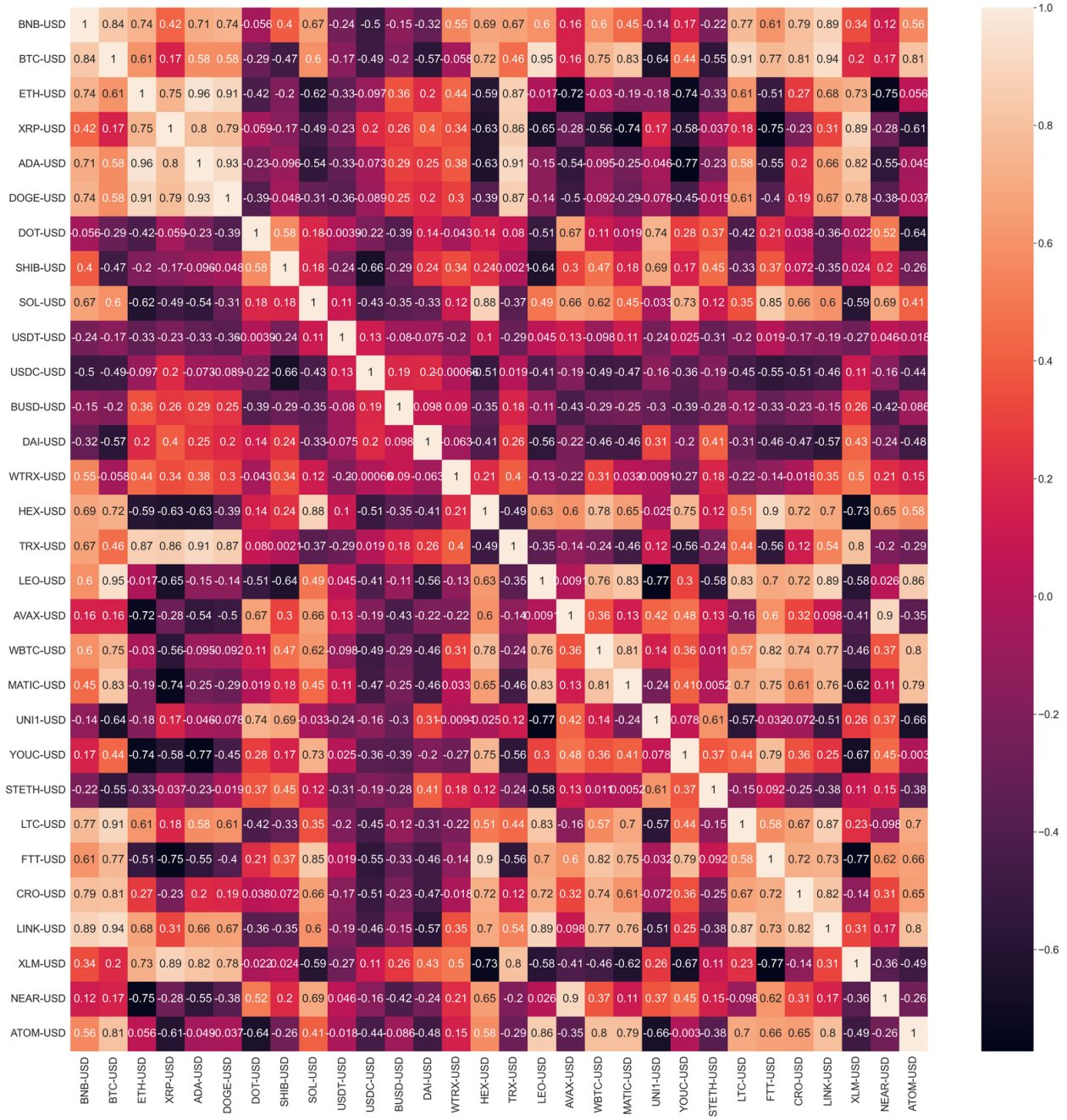


Figure 3.9: Correlation matrix computed with Spearman's Correlation on 30 different crypto assets.

Distance correlation

In statistics, Distance Correlation [88] is a way of measuring the association strength of two random variables, whether it is linear or nonlinear associations. This method also works on multiple dimensions and uses distances between observations as part of its calculations. Mathematically, if two-time series are defined as X_k and Y_k for k = 1, 2, .., n, the Distance Correlation can be formulated as:

1. Compute the nxn distance matrices $(a_{j,k})$ and $(b_{j,k})$ as:

$$a_{j,k} = ||X_j - X_k|| \quad \text{For j, k = 1, 2, 3, ..., n}$$

$$b_{j,k} = ||Y_j - Y_k|| \quad \text{For j, k = 1, 2, 3, ..., n}$$

2. Compute the doubly centered distances as:

$$A_{j,k} = a_{j,k} - \overline{a}_{j.} - \overline{a}_{.k} + \overline{a}_{..} \quad \text{For j, k = 1, 2, 3, ..., n}$$

$$B_{j,k} = b_{j,k} - \overline{b}_{j.} - \overline{b}_{.k} + \overline{b}_{..} \quad \text{For j, k = 1, 2, 3, ..., n}$$

   Where $\overline{a}_{j.}$ and $\overline{a}_{.k}$ are the j-th row and k-th column mean, and $\overline{a}_{.}$. is the overall mean of the X-sample matrix. This same notations apply to b.

3. Using the previous values, the sample covariance can be computed as:

$$dCov_n^2(X,Y) = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}B_{j,k}$$

4. Finally, the Distance Correlation can be computed as the fraction of the covariance by the product of the standard deviations:

$$dCor(X,Y) = \frac{dCov^2(X,Y)}{\sqrt{dVar(X)dVar(Y)}}$$

   Where $dVar(X) = dCov_n^2(X,X)$. Same notation for Y.

In this research, the distance correlation between assets was computed using this procedure, with the help of the library SciPy [48]. This library is used broadly in technical computing, as it includes modules for "optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering".

Figure 3.10 shows the Distance Matrix of the 30 crypto assets. This matrix shows similar results to Pearson's correlation matrix, indicating that there are nearly no nonlinear correlations among the 30 first cryptocurrencies. In general, the assets that have strong negative or positive correlations conserve similar extreme values in all three matrices, indicating that the three first methods do not differ greatly from each other.

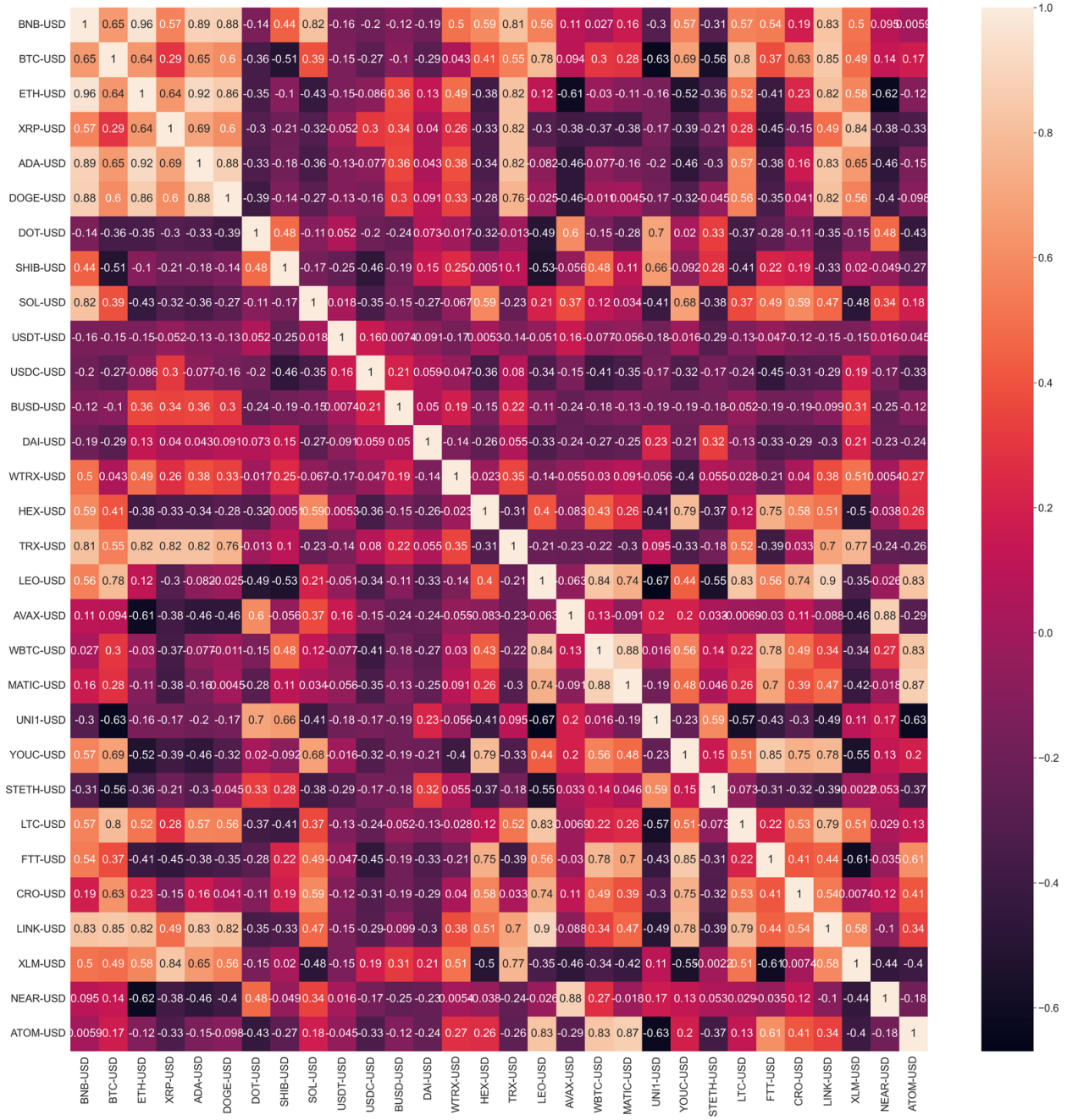


Figure 3.10: Correlation matrix computed with Distance Correlation on 30 different crypto assets.

Maximum Information Coefficient (MIC)

Maximum Information Coefficient (MIC) is a way of measuring the linear and nonlinear association strength between two variables based on information theory and was developed in 2011 [74]. In that same year, [86] referred to the MIC as "the correlation for the 21st Century". If two time series are defined as X_i and Y_i for i = 1, 2, .., n, the MIC can be computed (for discrete variables) as follows:

1. Compute entropy to measure uncertainty of each distribution as:

$$\mathcal{H}(X_i) = -\sum_{i=1}^{k} p(X_i) log_2 p(X_i)$$

$$\mathcal{H}(X_i, Y_i) = -\sum_{X_i}\sum_{Y_i} p(X_i, x_j) log_2 p(X_i, X_j)$$

2. Compute the entropy for X, Y and X, Y using the above formula ($\mathcal{H}(X)$, $\mathcal{H}(Y)$, $\mathcal{H}(X, Y)$) and calculate the Mutual Information (MI):

$$\mathcal{I}(X_i; Y_j) = \mathcal{H}(X_i) + \mathcal{H}(Y_j) - \mathcal{H}(X_i, Y_j)$$

   Where $\mathcal{I}(X; Y)$ is equal to $\mathcal{I}(Y; X)$. It is worth noting that the Mutual Information (above) reveals the amount of information of one variable described by the other, and isn't dependent on the functional structure of the core association .

3. It's nearly always feasible to create exactly 1 bin for each datapoint (x, y), since the variables X and Y are reals. However, this would suppose that the Maximum Information would be an extremely high number. For this reason, the authors in [74] proposed to take n_x for X and n_y for Y such that:

$$n_x \, n_y \leq N^{0.6}$$

   Where N is the size of the data sample

4. Finally, the MIC can be computed as:

$$MIC(X;Y) = \max_{X,Y_{total}<B} \frac{\mathcal{I}(X;Y)}{log_2(min(X,Y))}$$

Where:

- $\mathcal{I}$: MI as desribed in step 2.
- X and Y: number of bins for each.
- X,Y_{total}: total number of bins.
- B: Number less than the product of the number of bins (usually $N^{0.6}$).

Some of the pros of MIC are:

- Captures linear and nonlinear relationships.
- No assumptions on the variables distributions.
- Deals well with outliers because of the foundations of mutual information.

Some of the cons of MIC are:

- Coefficient is in the range $[0, 1]$, so it does not report if the correlation is negative or positive. 0 indicates that both variables do not share information and 1 means completely identical probability distributions.
- Computationally expensive.
- Despite the popularity of MIC and the great results presented in [74], [83] presents several scenarios where MIC underperforms with respect to Distance Correlation and, in some cases, Pearson's Correlation.

The final correlation matrix is shown in Figure 3.11 and is created using the Maximum Information Coefficient (MIC) explained above. One thing to note is that the MIC is in the range $[0, 1]$, so it does not indicate the sign of the relationship between assets. Focusing on the fundamental values we presented for the other matrices, we note that, in this case, several pairs have a correlation of 1, indicating a perfect correlation. For example, the correlation between WBTC-USD and BTC-USD, FTT-USD with HEX-USD, and LINK-USD with FTT-USD all have a correlation of 1. On the other hand, in the Distance Correlation matrix, these have a correlation of 0.3, 0.75, and 0.44, respectively. Moreover, the most

negatively correlated assets present in Spearman's correlation matrix had a coefficient of -0.77. In contrast, in the MIC matrix, the coefficients were 0.75 (FTT-USD with XLM-USD), 0.8 (YOUC-USD with ADA-USD), and 0.7 (UNI1-USD with LEO-USD). Overall, the MIC matrix indicates similar values as the other matrices for highly positively correlated assets (e.g., BTC-USD with BNB-USD) but differs in the calculation of negatively correlated assets due to the range of values the coefficient is in.

Overall, in the literature, different approaches have been used to compute the covariance matrix in the context of portfolio selection. The classical method shown in most papers is Pearson's correlation [84], [46], [55] because it is the "go-to" way to compute correlation when it is not the main focus of your research. Other methods that put more emphasis on the computation of the correlations and how to best capture the dynamics of asset correlations have seen important benefits in using Spearman's Correlation due to its "computational simplicity", "high efficiency", and "outlier robustness" [11], [21], [7]. Furthermore, distance correlation has also been used in portfolio allocation research due to its ability to "capture fat-tail nature" and "nonlinear dependence" [87], [8], [73]. However, it has been criticised because it is computationally expensive [22].

In general, correlation analysis is essential for any financial time series analysis. Strongly correlated assets, such as ETH and BNB, might be used to predict each other and thus optimise the return of a portfolio. For example, if two assets are historically strongly positively correlated and one of them has started to rally while the other not, one might see this as an indication that the second asset is going to follow the other and have an upward trend, thus taking a position on that asset. On the other hand, negatively correlated assets can be used to predict and offset the risk, thus optimising the risk of a portfolio. For example, if a portfolio is long on ETH and the predictions point toward a downwards trend of this asset, but the position is too big to unload it rapidly, one might prefer to long NEAR or AVAX because they are negatively correlated to ETH, thus offsetting risk.

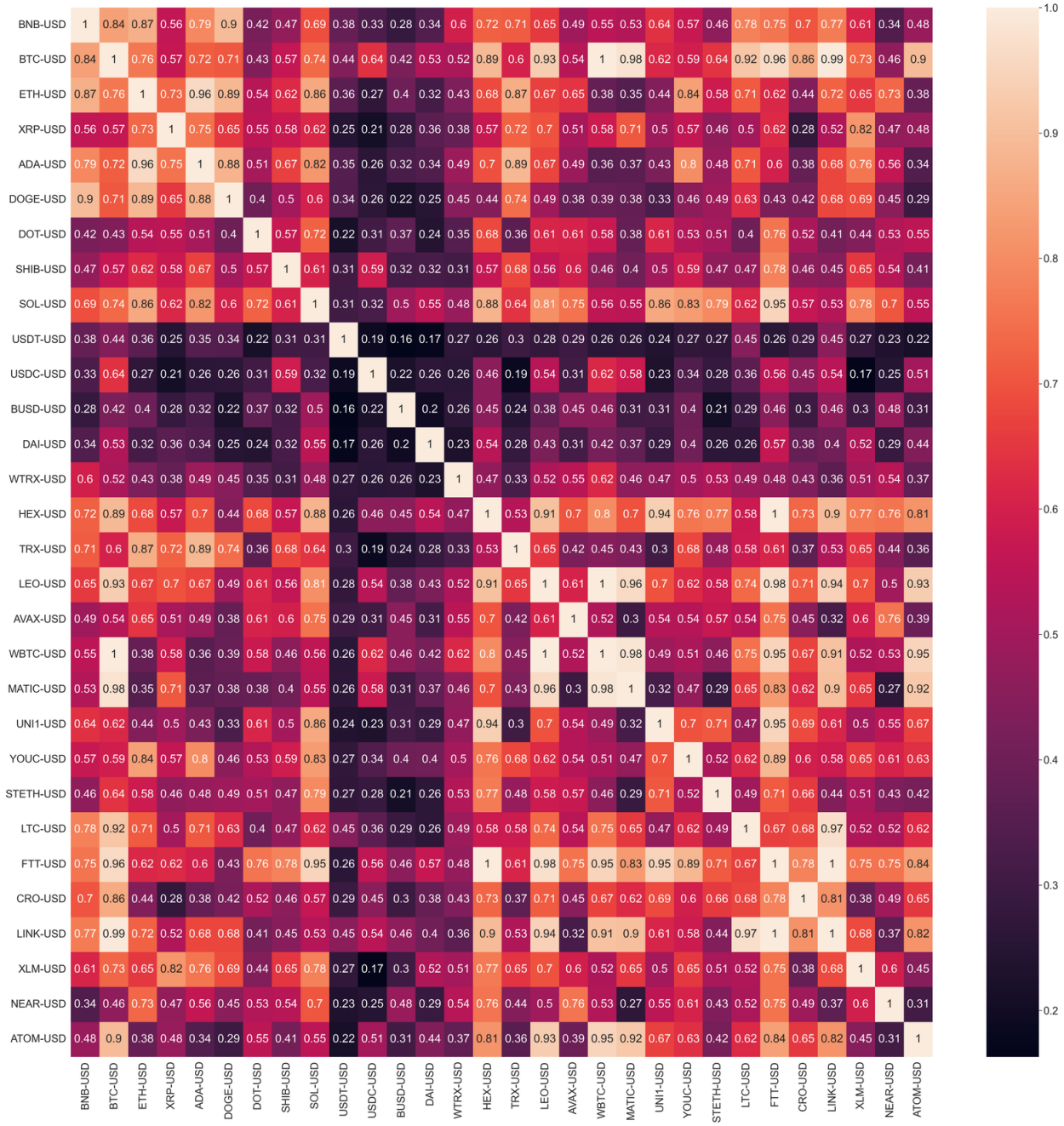


Figure 3.11: Correlation matrix computed with Maximum Information Coefficient (MIC) on 30 different crypto assets.

3.3.4 Do returns follow a Normal Distribution?

As previously stated, it is common to study the returns of financial assets rather than the raw asset prices when examining an asset's time series. This is because returns have more appealing statistical properties for analysis, in addition to a scale-free assessment of the assets' performance.

This section contributes to the Time Series Analysis by studying the returns distribution, which is assumed to be normally distributed in many models and studies. The distribution matters because it gives further information on the performance of the assets overall and how volatile the asset can be, as well as having better insights on outliers and extreme values in the time series. In this case, several plots are analysed, including the histograms, Quantile-Quantile plots, Box plots and Empirical Cumulative Distribution Functions.

Figure 3.12 shows the Daily Returns histogram (bins = 400) of different cryptocurrencies:ETH, XRP, ADA and DOGE. The Kernel Density Estimate is also plotted to give us a better visualisation of the probability density. The rug (dark blue lines sitting on the x-axis) is analogous to a histogram with zero-width bins, and it is shown to further help us visualise the distribution of returns. It can be seen that none of the distributions follows a Normal distribution, as is assumed in many models. In the DOGE-USD plot(bottom right), one can see that the distribution has a fat tail on the right, indicating that it is positively skewed. It can also be noted that the returns have some outliers (around 1% and 3.4%) that are captured thanks to the rug plot. Similar behaviour can be noted in the ADA-USD plot (bottom left), where some positive outliers are captured by the rug plot, and the distribution is positively skewed. The other plots show that the asset distributions are heavy-tailed but not necessarily skewed.

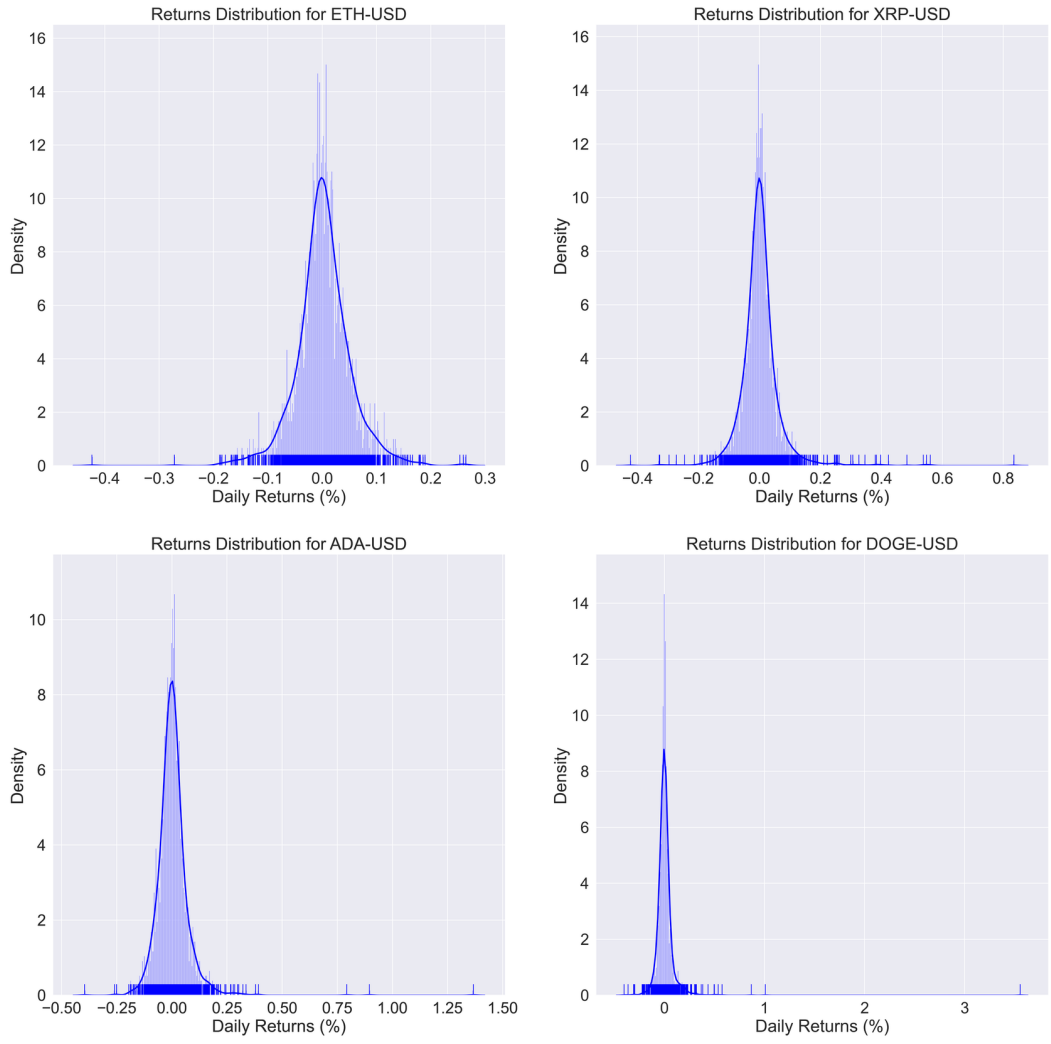


Figure 3.12: Histogram with 400 bins showing the Daily Returns of different crypto assets, where the Kernel Density Estimate (KDE) and rug are also shown.

Figure 3.13 shows the Quantile-Quantile plots for the four previous cryptocurrencies. The quantile-quantile(q-q) plot is a graphical way of determining if two datasets come from a population with a common distribution(in this case, we compare one time series to a normal distribution). Therefore, a q-q plot is trying to answer the question, "How

similar are the quantiles of the dataset compared to the ones of a theoretical probability distribution?" In this case, the theoretical probability distribution is a Gaussian Distribution.

A quantile is a fraction of points below a given value. For example, the 0.8 quantile is the point where 80% of the data falls below, and the rest falls above that value. The red line in the plots is a 45 degrees reference line ($y = x$), and it is plotted to see if the two distributions are linearly related. If they are (a returns time series vs normal distribution), the points (cryptocurrency time series) should lie on the red line (normal distribution).

It can be seen that the four time series distributions do not follow a normal distribution (this is, in fact, true for the remaining cryptocurrencies of the portfolio). The blue lines, representing the assets returns, diverge severely on the edges of the graphs, indicating that these distributions have fat tails on each side. The top left graph (ETH-USD) has a smooth divergence from the normal distribution, as opposed to the bottom right plot (DOGE-USD), where several points are far from the overall graph. This is because the distribution of ETH-USD has fat tails on both sides but remains symmetric, as seen on 3.12. On the other hand, DOGE-USD was heavily positively skewed and had several outliers that the rug plot showed, which is further seen on the q-q plot.

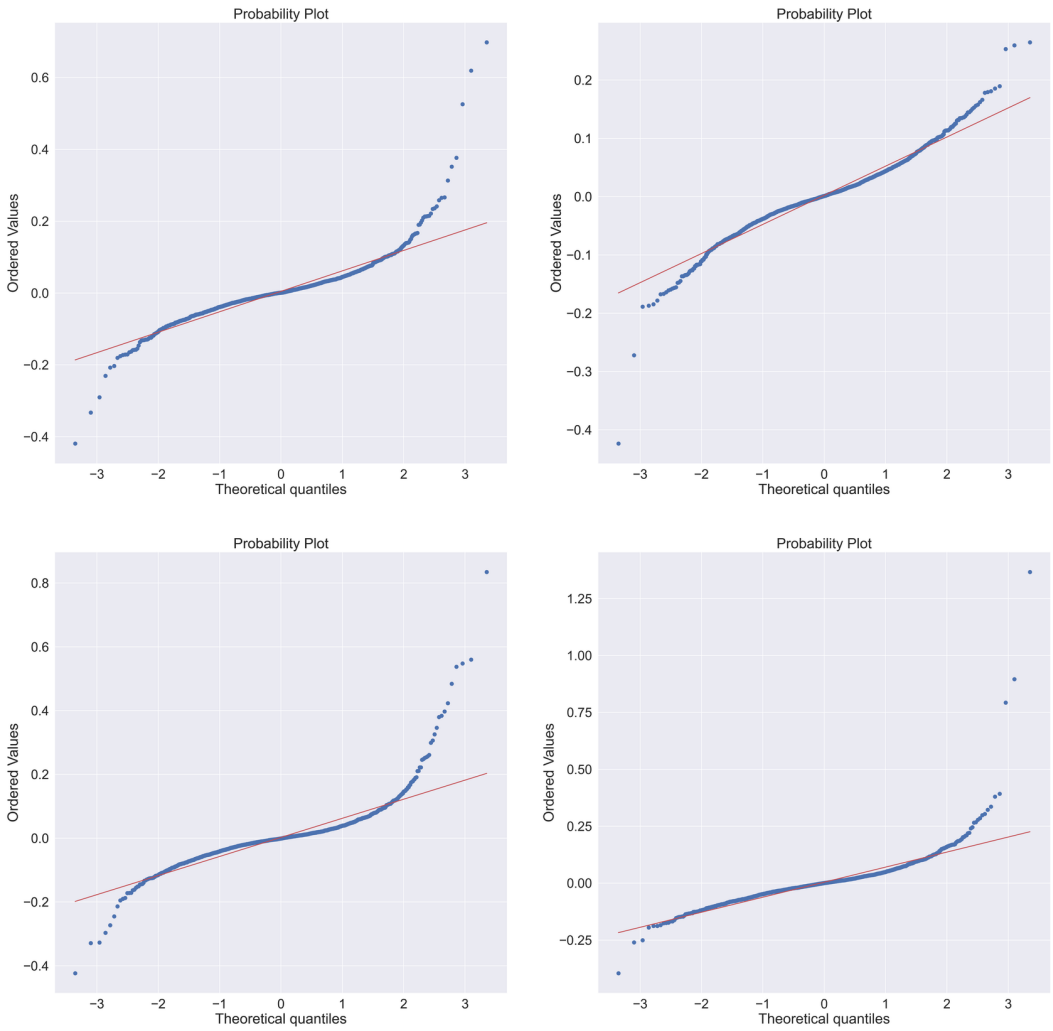


Figure 3.13: Q-Q plots comparing four cryptocurrencies (ETH, XRP, ADA and DOGE) returns distribution (blue dots) to a normal distribution (red line). Top left: ETH- USD, top right: XRP-USD, bottom left: ADA-USD, bottom right: DOGE-USD.

Figure 3.14 shows the Box plots for the BNB, ETH, XRP and ADA assets. This type of plot shows the "five-number summary" of a set of data: minimum, maximum, first and third quartile, and median. The box is drawn from the first quartile to the third one. The line inside the boxes indicates the median, and the horizontal lines outside the

boxes indicate the maximum and minimum values. In this case, as the number of outliers is considerable, they are shown as dots and lie outside the "maximum" and "minimum" values. It is worth noting that a datapoint is considered an outlier when it is beyond three standard deviations from the mean.

These plots clearly show that the number of outliers in the data is large, which means that extreme returns are seen often, whether these are positive or negative returns. It is worth noting that the values on the y-axis differ, as some assets have more extreme returns than others. For example, BNB-USD has several returns above 0.5%, which is not seen in ETH-USD. On the bottom right plot, ADA-USD has the largest outlier, indicating a return of more than 1.30%, which corresponds to ADA-USD's spike on the 28th of November 2017.

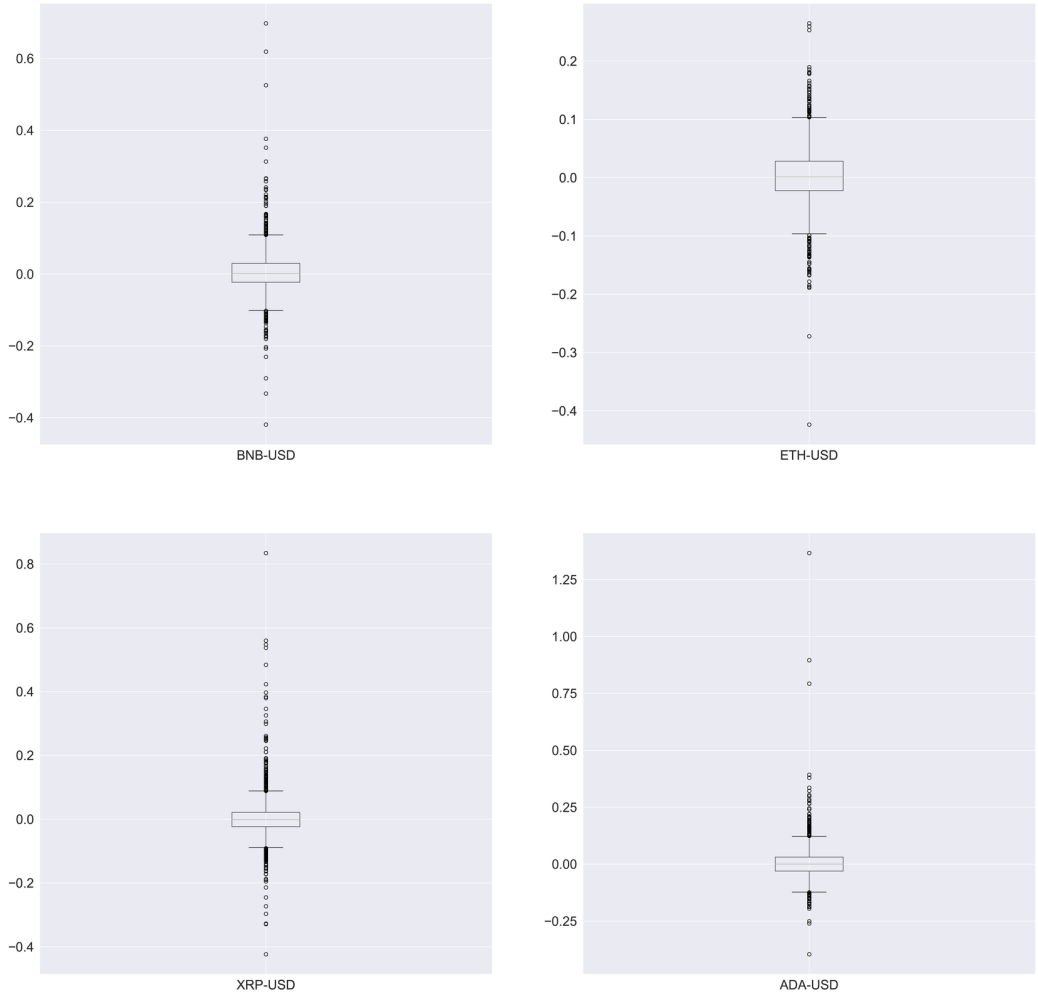


Figure 3.14: Box plots for four cryptocurrencies (BNB, ETH, XRP and ADA) returns. Top left: BNB-USD, top right: ETH-USD, bottom left: XRP-USD, bottom right: ADA-USD.

Table 3.3.4 shows the Mean, Standard Deviation, Kurtosis and Skewness of the 50 cryptocurrencies. We can see that all the cryptocurrencies have a positive mean return, except 'ICP', which has a mean of -0.005938, and in general, the values are very close to 0 for all coins. Focusing on the standard deviation, we see that the numbers change from

pair to pair. The highest standard deviation is achieved by 'YOUC', with an overall value of 1.02, indicating that the volatility of this asset is considerably high compared to other coins such as 'BUSD', with a volatility of 0.0037. Finally, it is particularly interesting to look at the Kurtosis and Skewness properties of the assets. These tell us how heavy the distribution tails are, and measure the symmetry of the distribution, respectively. For the Kurtosis values, a standard normal distribution has a value of 3, and a higher value means that the distribution has a thing 'bell' with a high peak, and a lower value means that the distribution has a broad peak and 'thick' tails. By looking at the table, we see that all distributions have a Kurtosis value ¿ 3, meaning that they are not normally distributed, except for the 'NEAR' asset, which has a value of 3. Furthermore, the Skewness values for a symmetrical distribution oscillate between -0.5 and 0.5, and it is considered moderately skewed' if the value is between -1 and -0.5 (negative) or 0.5 and 1(positive). Looking at the results, we can see that most of the assets have a high positive skewness. However, some popular cryptocurrencies, such as 'ETH', have a value of -0.22, meaning that the returns form a fairly symmetrical distribution. Overall, the assets returns distributions do not match the properties of a normal distribution, which further strengthens our belief that the returns are not normally distributed.

	BNB	ETH	BTC	XRP	ADA	DOGE	DOT	SHIB	SOL
Mean(x 10e-6)	4636	2229	2054	2369	4009	5891	3665	19171	7206
Std.Dev(x 10e-6)	61958	51246	38680	67919	76529	113843	71740	242913	80905
Kurtosis	21.36	5.17	7.08	26.43	75.15	544.49	8.34	201.26	3.31
Skewness	1.90	-0.22	-0.15	2.62	5.10	18.25	1.10	11.97	0.53

	USDT	USDC	BUSD	DAI	WTRX	HEX	TRX	LEO	AVAX
Mean(x 10e-6)	00006	0006	0006	0036	1146	8201	4620	2019	5127
Std.Dev(x 10e-6)	4640	3860	3703	7689	43189	245587	80436	35440	82413
Kurtosis	43.63	28.55	102.3	31.02	5.74	567.32	66.42	58.24	11.29
Skewness	0.99	0.68	0.07	-0.43	0.22	20.89	5.06	4.14	1.44

	WBTC	MATIC	UNI1	YOUC	STETH	LTC	FTT	CRO
Mean(x 10e-6)	2154	8090	3808	54674	2913	2413	3619	3716
Std.Dev(x 10e-6)	40441	89570	82177	1026398	52759	57262	49243	74923
Kurtosis	10.61	10.76	31.29	647.4	2.67	15.71	6.7	94.66
Skewness	-0.018	1.53	3.03	25.02	-0.01	1.33	0.62	5.86

	LINK	XLM	NEAR	ATOM	XMR	BTCB	XCN1	ALGO
Mean(x 10e-6)	4535	2543	5239	3006	1690	2140	4845	0388
Std.Dev(x 10e-6)	73116	65157	83058	71078	55411	54697	68403	69726
Kurtosis	7.21	19.05	3.00	4.46	6.83	21.10	10.41	7.67
Skewness	0.83	2.05	0.54	0.11	-0.06	1.39	0.52	0.33

	ETC	BCH	VET	FLOW	MANA	ICP	SAND	APE3
Mean(x 10e-6)	2521	1175	2555	588	5894	-5938	7934	1274
Std.Dev(x 10e-6)	64738	66836	67261	75014	94269	76274	98664	99573
Kurtosis	7.85	12.13	5.06	6.04	81.44	4.25	17.97	5.96
Skewness	0.77	1.25	0.36	1.20	5.58	0.55	2.33	0.88

	XTZ	FRAX	HBAR	TONCOIN	FIL	TUSD	THETA	AXS
Mean(x 10e-6)	2434	0048	2532	6508	4450	0005	3965	11617
Std.Dev(x 10e-6)	73116	65157	83058	71078	55411	54697	68403	69726
Kurtosis	10.95	35.41	38.42	33.57	20.19	81.99	8.06	7.31
Skewness	0.72	0.62	3.44	4.21	2.39	-0.45	0.96	1.77

Table 3.3: Table of 50 cryptocurrencies and the Mean, Standard Deviation, Kurtosis and Skewness of their daily returns.

Overall, this section analysed the distribution of returns of several cryptocurrencies, and gave us a better understanding of what distributions underlines the asset's returns. The following is a brief summary of each plot shown:

- *Histogram*: The histograms included the Kernel Density Estimate (KDE) and the rug-plot. It showed how the assets returns could be grouped into 400 bins, an estimate of the probability density function, and each of the assets returns as the rug-plot.
- $Q - Q$ *plot*: The Quartile-Quartile plots clearly showed how the tails of the returns distributions differed from the Gaussian one. These plots mainly showed the fat-tails, skewness and kurtosis of the returns.
- *Box plot*: the box plots showed the 'five-number summary', and spotted the extreme values of the distributions. It further supported the idea that returns suffer from extreme values, which can be great for a portfolio when it is positive!
- *Kurtosis and Skewness*: The tables showed the values of the assets for their returns distributions kurtosis and skewness. The majority of the values differed from the standard normal distribution ones, indicating that the daily returns do not follow this type of distribution.

Finally, it is concluded that the returns of the cryptocurrencies present in the portfolio do not follow a normal distribution. As previously said, it is important to note this in order to understand why frameworks such as Markowitz's MPT fail to capture reality. As non-normality can arise from extreme, positive or negative events, this leads to an underestimation of the portfolio's overall risk.

3.4 Predictive Model(s) Selection

As it has been seen in the Time Series Analysis done, the data has some characteristics that are not ideal for modelling via classic econometric models such as Autoregressive (AR), Moving Average (MA) or Autoregressive Moving Average models (ARMA). For this reason, further research on Model Selection must be done.

Due to the non-strong autocorrelation in the time series, fitting any of the models mentioned above would result in poor predictive performance of the models, as they would be

unable to capture the dynamics of the data. As shown, the values of p and q that these models require are not evident, so these types of modelling are discarded.

Furthermore, the correlation analysis indicated a nonlinear correlation component in the overall correlation among the assets. This indicated that for the predictive task, a linear model would not be able to capture the whole co-dynamics of the assets, and a nonlinear method would outperform it (theoretically). As Machine Learning (ML) models normally generalize well on nonlinear data, and a 'cleaned' and pre-processed dataset was available, we decided to shift the model selection to the ML literature.

As previously explained in Section 2.7.2 where nonlinear approaches were studied, the literature shows that Support Vector Regression (SVR) performs well in forecasting tasks and can capture nonlinearities. Furthermore, several distinct researches [76], [32], [27] concluded that SVR outperformed classic econometric/statistical models present in the literature, such as Generalized Autoregressive Conditional Heteroskedasticity (GARCH), as it is able to perform well on non-normally distributed data. For these reasons, a Support Vector Regression (SVR) model was selected for the prediction task, as it suited the characteristics of the data and it had already proven that it outperformed other statistical and econometrics models such as AR, MA, ARMA and GARCH.

As a Hyperparameter Optimization (HO) approach, the literature showed that Genetic Algorithms (GA) combined with SVR had proven outstanding performance. Given that there was a gap in the literature on GA-SVR approaches for covariance matrix forecasting in a cryptocurrency portfolio, GA was selected as the HO algorithm. For this reason, it is believed that this research presents a novel approach to the literature.

Finally, a benchmark was chosen to measure how well the SVR was performing with respect to another model. As one of the goals of this research is to show that the co-movement of asset returns is also explained by a nonlinear factor, it was of interest to choose a purely linear method to compare a nonlinear vs a linear model. For this reason, Linear regression (explained in Section 2.3) was selected, as it is a simple yet very used Machine Learning model able to capture linearities between a scalar response and one or more explanatory variables.

3.5 Summary

Overall, this chapter (Chapter 3) has covered and analysed many properties of the dataset used in this research and finally presented some characteristics of the time series. Furthermore, these characteristics have been at the core of model selection, and two models have been presented for the prediction task (the main model and the benchmark). In this summary, the main ideas and key outcomes of the analysis are presented for each section, and some further comments are indicated:

- **Data**: The data section started presenting the data set used in this project. From the time series length to what cryptocurrencies were included in the portfolio. It is further subdivided into two subsections, the data acquisition and data pre-processing parts. The first one presented the Python libraries used and how it was used to fetch the data, as well as in what format the data was acquired. The second part showed the data pre-processing part to format it for a better analysis. This task includes spotting values that do not correspond to the time series, such as outliers of NaN values and visualising the dataset to understand it better.
- **Time Series Analysis**: Time series Analysis (TSA) was carried out in this section, which is a crucial step in all Machine Learning research that deals with time series. Several characteristics of the series were analysed to choose a model that adjusted to the features of the data.
  - Firstly, the stationarity of returns was examined. As a stationary time series means that the process remains in statistical equilibrium with probabilistic properties over time, this is particularly interesting when making predictions from already-seen patterns. This part was done using two types of tests, ADF and KPSS, which both indicated that all returns-series of the dataset were stationary.
  - Secondly, the autocorrelation of the dataset was measured, to have a quantitative understanding of how much of an impact did past prices had on its own future prices. This was done by applying the ACF and PACF, and analysing the resulting plots. It was concluded that the cryptocurrencies used in this research

do not have large autocorrelation, so no accurate AR/MA/ARMA model could be used to describe the data.

- Thirdly, a correlation analysis between assets was carried out, which indicated what cryptocurrency prices moved in tandem, independently, or in opposite directions. Different types of correlations were measured, which included Pearson's, Spearman's, Distance correlations, as well as the Maximum Information Coefficient. It was shown that some assets prices were strongly positively correlated, and some assets were strongly negatively correlated, which is essential when building a portfolio in order to minimize risk and maximize returns.

- The last part of the TSA consisted of an analysis of the returns distributions. As it is commonly assumed in the econometrics literature that returns follow a normal distribution, it was of importance to analyze their distribution. A series of plots that included Histograms, Kernel Density Estimates, Q-Q plots and Box plots were presented. The conclusion from this section was that the assets returns that we deal with do not follow a normal distribution at all, and have outliers that could be beneficial for boosting portfolio returns, or in the contrary, wipe out all profits from a portfolio.

- **Predictive Model(s) Selection**: This part of the overall analysis consisted in choosing a model that was well-suited for the data. By inferring information from the Time Series Analysis and by looking at the literature on linear and nonlinear methods presented in Sections 2.7.1 and 2.7.2, a Machine Learning model combined with an Evolutionary Algorithm were chosen for the prediction task. Finally, a linear model was chosen as the benchmark in order to compare linear and nonlinear models, and further show that a nonlinear model should be used in the Portfolio Allocation task.

Chapter 4

Model Implementation

Chapter 4 presents the two main models that were implemented in this research (GA-SVR and SVR), comparing them and analysing their key characteristics. It is divided in a brief Introduction, followed by a discussion on the environment and programming language that were used, and ending with a presentation of the models that are used for the experiments.

4.1 Introduction

As previously stated, different models are implemented in this research in order to best adjust to the dataset available and to capture the nonlinear dynamics of the time series. Three Machine Learning algorithms are implemented, which are the GA-SVR, SVR and Linear Regression models. The two first ones consist of the same principle of Support Vector Regression and are explained in-depth, as these are the main models. The last model, Linear Regression, is a linear model implemented to compare the nonlinear techniques to a linear one and thus, serves as a benchmark.

4.2 Programming Language and Environment

The programming Language used is Python [91] in an Anaconda [2] environment, done in Jupyter Notebook [50].

Anaconda is a distribution of several programming languages, including Python, and it is used for scientific computing, such as Machine Learning. Its purpose is to simplify packet management and deployment, and it is used in all three main Operating Systems: Linux,

Windows and macOS. One of the main perks of Anaconda is the Virtual Environment manager that it incorporates. This allows us to separate different applications and avoid problems with different dependencies. For example, in this research, Python 3.6 version was used and included in an environment, as well as the libraries that were used, such as $scikit-learn$ [66]. By using a virtual environment, these packages and their specific versions did not interfere with other projects and versions of different packages installed on the Computer.

Finally, Jupyter Notebook is an open-source web application used for creating and sharing documents that contain code, visualizations and text. It was used for the purpose of this research because as there were many graphs, plots and code to present, Jupyter Notebook was ideal for this task.

The following section indicates how the models were developed in Python in a Jupyter Notebook using an Anaconda environment. It starts by presenting the SVR, followed by the GA-SVR, and ends with the benchmark model.

4.3 Support Vector Regression (SVR)

The theory of Support Vector Regression was explained in Section 2.4.2, where a description of the Machine Learning model was presented. In practice, this model was already implemented in the $scikit-learn$ [66] library so the $SVR()$ class and methods were used for the model.

The implementation is based on a library for Support Vector Machines called *libsvm* [15].The input parameters are as follows:

- **Kernel**: This parameter specifies the type of kernel used in the SVR. In our case, three types of kernel were used: Linear, Polynomial and Radial Basis Function kernel (RBF). The two latter ones are nonlinear kernels, thus the algorithm learns a nonlinear function that is in charge of predicting.

- **Degree**: This parameters is only useful when the selected kernel is the Polynomial one. It indicates the degree of the polynomial kernel, and is ignored by all other

kernels.

- **Gamma**: For nonlinear SVR, this parameter defines the distance of the influence of a single training datapoint. This means that a high value refers to 'close' and a low value refers to 'far'. In other words, Gamma configures the sensitivity to differences in feature vectors. In the case where Gamma is too large, overfitting occurs, and when brought to the limit (infinity), the kernel becomes a unit matrix and it leads to a perfect fit, thus totally overfitting.

- **C**: This parameter can be thought as the penalty parameter of the error term (a regularizer). It tells the SVR till what extend the model can misclassify a training point. The larger it is, the smaller the margin of the SVR is accepted if the decision function improves at regressing correctly.

- **Epsilon**: This refers to the insensitive Hinge loss explained in Section 2.4.2. This parameter acts as a margin of tolerance where no penalty is given to the errors.

As stated in Section 2.4.2, these are user-defined inputs. The values for these will be made clear when describing the experiments.

4.3.1 Training process

The training process of the SVR mentioned above model consists of two parts and is done after creating the SVR with $scikit-learn$. The first one is the covariance matrix computation, and the second is the data formatting and learning process. The first part describes the process of transforming the returns time series into covariance matrices. The second part deals with how the data was structured to feed it to the learning algorithm.

Covariance Matrix

As the original data was price-series, the first step was to compute the returns of each cryptocurrency as explained in previous sections. With the returns series, the sample covariance matrix was computed.

This matrix is a $S - by - S$ matrix $\mathbf{K} = [q_{jk}]$ with entries:

$$q_{jk} = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \overline{x}_j)(x_{ik} - \overline{x}_k)$$

where S is the number of assets in the portfolio, q_{jk}, is an estimate of the covariance between variable j and k of the population generated by the data. In other words, the covariance of assets j and k is the sum of N samples (in our case, this is done with a window of 10) of the return of asset j at time i minus the average return of that asset times the return of asset k at time i minus the average return of that asset.

Finally, this process is done for all the time steps available in the data minus 10 for the window (1742). This gives us the total amount of matrices to decompose and use for training and testing.

Data formatting and learning process

In order to train the model, the $.fit()$ method from the SVR Class was called. This method accepts two array-like matrices that consist of X and Y training sets. The dimensions of these matrices must be [n x m] and [n x 1] dimensional X and Y matrices respectively (n meaning number of samples and m meaning the number of features).

The format of the dataset available for this task can be seen in Figure 4.1 and is further divided into 80% training set and 20% testing set (that is, taking the last 20% of the rows of each matrix). As we can see, the X data is a collection of C matrices, one for each Cholesky element of the Covariance matrix. As the ML model only accepts [n x m] and [n x 1] dimensional X and Y matrices, some modifications had to be done.

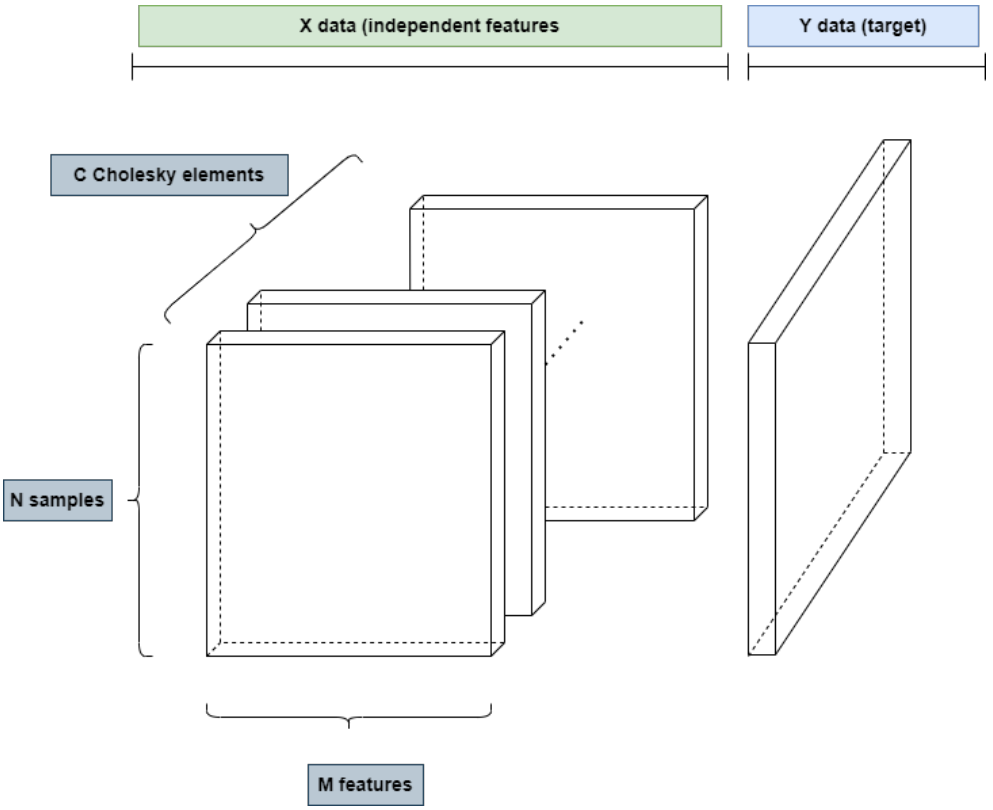


Figure 4.1: Graphical representation of the formatting of the X and Y datasets fed into the models.

The procedure followed for this task was similar to the one presented in [32], where a model is trained for each Cholesky element. Indeed, C SVRs were implemented to predict the $j-$day-ahead Cholesky element in the time series by regressing on the past M Cholesky elements. However, we considered the learning process to be online. At each step, all available data is used to update the predictor; thus, with each step, more data is fed into the model. This way, the algorithm dynamically adapts to new patterns in data, as it is generated as a function of time.

The model regresses on past M Cholesky elements and predicts the $j-$day-ahead Cholesky element. When this is done for all C elements, a list containing the elements of the Lower Triangle Cholesky-decomposed Matrix is obtained. Now, the Cholesky-decomposed matrix is done by constructing an all-zero [$number\text{-}of\text{-}assets$ x $number\text{-}of\text{-}assets$] matrix and introducing the forecasted elements of the list in the lower-triangle part of the matrix, thus having the forecasted Cholesky decomposed matrix. This two-step procedure can be graphically seen in Figure 4.2. Finally, the last step is multiplying the forecasted

Cholesky-decomposed matrix by its transpose to have the full covariance matrix prediction.

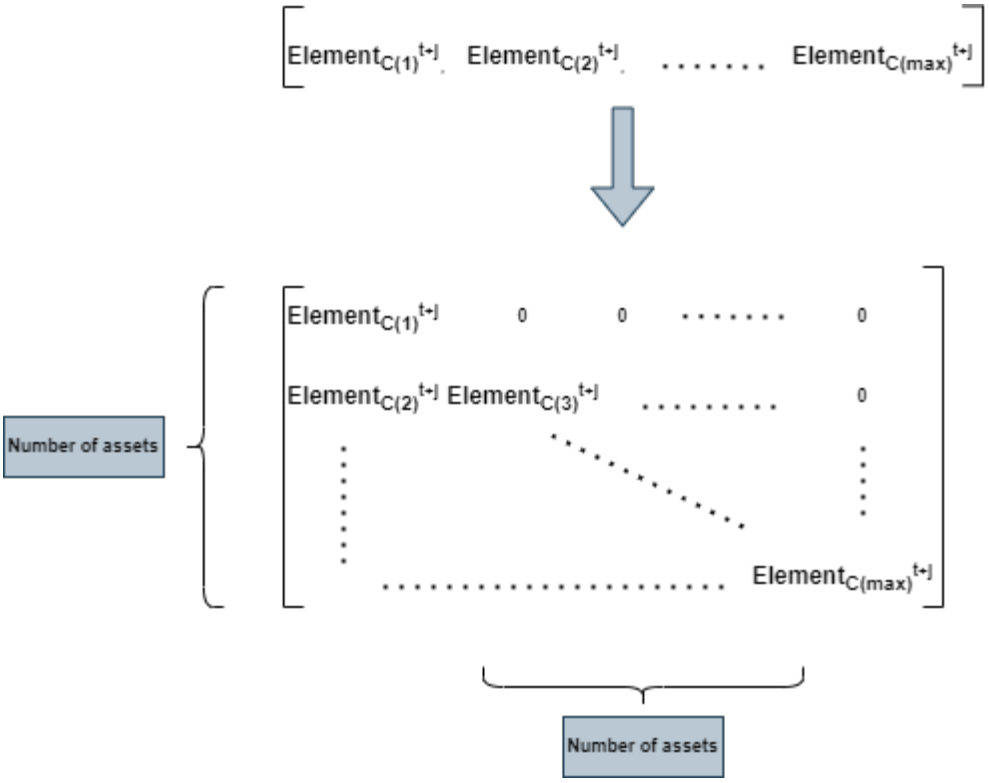


Figure 4.2: Two-step procedure of the generation of the Lower triangle Cholesky-decomposed forecasted matrix from the outputs of the SVR models.

4.4 Genetic Algorithm-Support Vector Regression (GA-SVR)

The GA-SVR is similar to the SVR algorithm, the difference being the addition of Hyperparameter Optimization (the GA part). For this reason, this section will focus on the explanation of the GA optimization and how it was added to the model.

In the proposed GA-SVR model, the parameters C, ϵ and γ are dynamically optimized during the training process by implementing Genetic Algorithms. At the same time, the *Kernel* type is pre-defined by the user in order to be able to select all types of kernels and compare them. This optimization process via GA can be seen in Figure C.18.

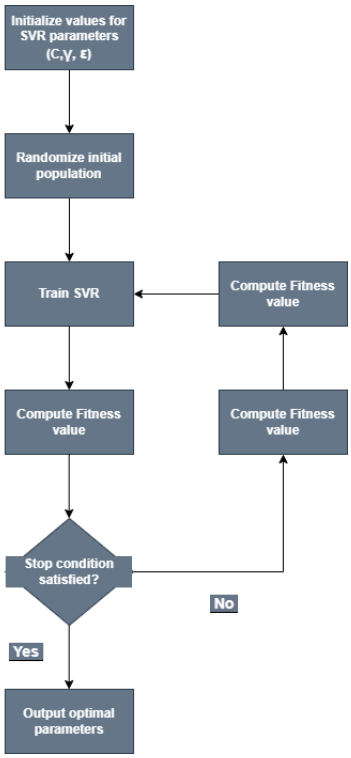


Figure 4.3: GA-SVR hyperparameter optimization procedure via Genetic Algorithm.

The theory of Genetic Algorithms was explained in Section 2.5.1. In this section, the focus will be on the Fitness Function used in order to asses the performance of each chromosome and the implementation part.

4.4.1 Fitness Function

The Fitness Function is in charge of assessing how well the chromosomes of the population perform and ranking them. This function must be defined prior to looking for the optimal values of our Support Vector Regression model. In the time series prediction and Hyperparameter Optimization literature, several functions have been proposed to measure the fitness of chromosomes [57], [49]. In this research, we compute the fitness of chromosomes based on the mixture of three different functions, R^2, $MAPE$ (Mean Absolute Percentage Error) and MSE (Mean Squared Error), defined below:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{(y_i - \overline{y})^2}$$

where the numerator is the sum of squares due to regression (variance explained by the model), and the denominator the total sum of squares(total variance). The y indicates the actual value, $\hat{y}$ the predicted, and $\overline{y}$ the average.

$$MAPE = \frac{1}{n}\sum_{i} |\frac{y_t - \hat{y}_t}{y_t}|$$

where n is the number of data points.

$$MSE = \frac{1}{n}\sum_{i}(y_i - \hat{y}_i)^2$$

where the variables are as above.

The mixture of fitness functions that had to be maximized is as follows:

$$L = R^2 - MAPE - MSE$$

This mixture was chosen because they measure different properties that should be optimized in order to achieve a good model. The R^2 gives an approximation of the relationship between the movements of the dependent variable based on the independent's variable movements. The MAPE measures how accurate the prediction is of the forecasting model, and MSE measures how well the fitting line is compared to the actual data points. However, MSE depends on the scale used. Finally, R^2 must be maximized (positive sign), and the others minimized (negative sign) because a higher R^2 means a better fit, and a lower $MAPE$ and MSE mean low error.

4.4.2 Implementation

DEAP

DEAP (Distributed Evolutionary Algorithms in Python) is an evolutionary computation framework for 'rapid prototyping and testing of ideas' [33]. It has all the data structures and methods needed to efficiently implement the majority of Evolutionary Algorithms, such as Genetic algorithms. This library was selected for the GA part of the research due to the clear documentation provided and its popularity.

The Genetic Algorithm procedure starts by taking the data that must be evaluated and predicted, which are four data sets containing the training and testing X and Y values.

Then, the techniques and methods used for selection, crossover, mutation and evaluation must be defined. For the first three, DEAP provides a series of already-implemented functions, as seen in Figure 4.4. The selected functions appear in orange and are further described below, as well as the definition of the Evaluation function:

- **Selection**: For this part, the $selTournament()$ function is chosen. It is in charge of selecting the best performing individual among 3 randomly chosen individuals.
- **Crossover**: $cxUniform()$ is called from DEAP, which executes a uniform crossover that modifies the 2 sequence individuals, as explained in Section 2.5.1. The attributes that define the chromosomes are interchanged with a probability of 0.25. The returned value is a tuple of 2 individuals.
- **Mutation**: The $mutGaussian()$ function is chosen for this task, which is in charge of applying a Gaussian mutation (with probability 0.1) of mean 2 and standard deviation 1.3 on the input individuals (these values were found to perform well). The function then returns a mutated chromosome.
- **Evaluation**: As previously stated, this function was implemented without using the DEAP library. The inputs to the function were three values, one for each parameter to optimize (C, ϵ and γ). Then, the function would fit the model with the input parameters and output the values for the loss functions described above. Like this, a fitness value can be computed in order to rank the combination of values for the parameters, and spot the best and worst performing ones.

Table 4.1 summarizes all training parameters and the range of values that the GA was allowed to search.

Parameter	Value
Population size	200
Number of Generations	5
$\boldsymbol{C}$	$(1, 2000)$
$\boldsymbol{\epsilon}$	$(0.1, 0.7)$
$\boldsymbol{\gamma}$	$(0.02, 0.22)$
Selection method	*Tournament*
Mutation method	*Gaussian*
Probability of mutation	0.1
Probability of crossover	0.25

Table 4.1: Table containing the training parameters for the GA-SVR model

At the end of this process, the best scoring values for C, ϵ and γ are outputted and passed into the actual SVR model in order to make the prediction. The numbers *population* and *generation* were 200 and 5, which were empirically found to perform well.

Crossover	Mutation	Selection
cxOnePoint()	mutGaussian()	selTournament()
cxTwoPoint()	mutShuffleIndexes()	selRoulette()
cxUniform()	mutFlipBit()	selNSGA2()
cxPartialyMatched()	mutPolynomialBounded()	selNSGA3()
cxUniformPartialyMatched()	mutUniformInt()	selSPEA2()
cxOrdered()	mutESLogNormal()	selRandom()
cxBlend()		selBest()
cxESBlend()		selWorst()
cxESTwoPoint()		selTournamentDCD()
cxSimulatedBinary()		selDoubleTournament()
cxSimulatedBinaryBounded()		selStochasticUniversalSampling()
cxMessyOnePoint()		selLexicase()
		selEpsilonLexicase()
		selAutomaticEpsilonLexicase()

Figure 4.4: List of functions for Crossover, Mutation and Selection offered by the DEAP library. In orange, the final chosen functions.

4.5 Linear Regression

The benchmark model is Linear Regression, and as previously mentioned, it was selected because it is a Machine Learning model and does not capture nonlinear relationships, so it is ideal for the research objectives of this work.

The model was explained in Section 2.3 and was implemented via the $scikit-learn$ library, calling the $Linearregression()$ function. This function uses Ordinary Least Squares to estimate the unknown parameters of the model, this is, minimizing the sum of squares of the differences between the actual and predicted variables. For the training process, the Linear Regression follows the same procedure as the SVR models explained in Section 4.3.1, that is, the covariance matrix creation, and data formatting and learning process.

As the model has already been mentioned and there is not much more done besides calling the above-mentioned function, no further explanation will be given on this model. For a more in-depth explanation, refer to Section 2.3 and [37], [38].

4.6 Summary

The Model Implementation Chapter presented an in-depth explanation of the different predictive models used at the core of this research. It also gave the reader a better understanding of some of the libraries used and how the overall environment was set. It covered the end-to-end training pipeline of the models and how returns series were transformed to covariance matrix predictions. The Chapter was divided into 5 main topics that are summarized and discussed below:

- **Introduction**: The Chapter started with a brief introduction presenting the three ML models. It emphasises the goal of this research, which is developing a model to capture the nonlinear dynamics of the time series.
- **Programming Language and Environment**: This brief section served to give a better understanding of the technology behind this research. It introduced Python, Anaconda and Jupyter Notebook and how Virtual Environments were leveraged.
- **Support Vector Regression (SVR)**: This is the first model presented. Firstly, we introduce $scikit-learn$ library and its in-built function $SVR()$. The parameters

required to call this function are explained, which are 'Kernel', 'Degree', 'Gamma', 'C' and 'Epsilon'. Then, the training process is explained, which consists of a 'Covariance Matrix' and a 'Data formatting and learning process' part. The first one oversees the technique to compute the covariance matrix and its properties. The second one deals with how the data was formatted in order to be inputted into the model properly. Finally, a series of diagrams are included in order to graphically explain the data formatting and the forecasting procedure.

- **Genetic Algorithm-Support Vector Regression (GA-SVR)**: Emphasis on the Genetic Algorithms are given in this part. Firstly, the overview of the GA procedure is explained and graphically shown. Then, the Fitness Function is covered. As this had to be manually done and we had the freedom to select a specific function or a combination of functions, the latter one was chosen, which consisted of a maximisation of the R^2 and a minimisation of the MAPE and MSE metrics. This was done to make sure the GA learnt optimised hyperparameters. The last part introduced the DEAP library, which was at the core of the GA implementation. The functions for Selection, Crossover, Mutation and Evaluation, as well as their characteristics, were covered.

- **Linear Regression**: This brief section served as a further explanation of the Linear Regression, which was described in previous sections. The library used for this model was $scikit-learn$ as it was easy and straightforward.

- **Linear Regression**: This brief section served as a further explanation of the Linear Regression, which was described in previous sections. The library used for this model was $scikit-learn$ as it was easy and straight forward.

Overall, some of the implementations of the models were already incorporated in libraries such as $scikit-learn$ and $DEAP$, as explained. However, the end-to-end pipeline involved manipulating data, formatting it and understanding what the models were actually doing, to then combining GA and SVR in a dynamic way, which was a tricky task. One of the most challenging parts of the implementation was the Genetic Algorithms and learning how to use $DEAP$. As there were many functions and algorithms to choose from, and there was not a clear structure or function on how to create GAs, the implementation was more complex than for the SVR. However, the documentation provided by $DEAP$ helped structure the GA procedure.

Chapter 5

Testing and results

This Chapter deals with the experiments part of the research, where the models explained in earlier sections get tested and analysed. In general, this, combined with Chapter 3 on Time Series Analysis, make the building blocks of the empirical component of this research. The main parts of this Chapter are a brief introduction to the models, a section on the dataset used, an explanation of cross-validation and why it was used, the three main experiments and the final summary and conclusion.

5.1 Introduction

The Machine Learning models that have been explained till now have been three: Support Vector Regression, Genetic Algorithm -Support Vector Regression and Linear Regression. However, as the SVR models had different Kernels and performances, the following models are treated as separate:

- **Linear Support Vector Regression**
- **Polynomial Support Vector Regression**
- **Radial Basis Function Support Vector Regression**
- **Genetic Algorithm - Support Vector Regression (RBF)**
- **Linear Regression**

For the purpose of experimentation, different lags have been compared as well. This is, different lag lengths have been tested on each model in order to have a better understanding

of how the performance changes depending on how many values it regresses on. In order to measure the accuracy and error, the R^2, MAE and MSE are measured.

5.2 Dataset

The portfolio of assets in this research was made of 5 different cryptocurrencies, which were: BNB-USD, ETH-USD, XRP-USD, ADA-USD and DOGE-USD. These were chosen because they had the same days of data publicly available, which spanned from November 2017 to August 2022. In addition to that, they are some of the largest crypto assets by market capitalization.

The format that the time series were fetched was in price series, which had to be transformed to returns series and then to covariance matrices, one for each day. The number of matrices created was 1752 [5x5] matrices. As explained in previous sections, the matrices were decomposed in a list of Cholesky elements and then arranged in order to enable the models to regress on M past elements of the same Cholesky element. Finally, the data was also normalized, so it contained numbers in the range of $[0, 1]$.

5.3 Cross-Validation

Cross-validation was performed in order to validate the models and assess how the results would generalize to another dataset. It is beneficial for other researchers in order to replicate these experiments and expect the same or similar results.

Cross-validation is a resampling method that utilizes distinct chunks of data to train and test the model on various iterations. In our case, K-fold was implemented because it is vastly used in Machine Learning and is easy to understand and easy to implement. As the value for K was chosen to be 5, the procedure is called 5-fold cross-validation, and it is as follows:

1. Shuffle randomly the data
2. Divide data into 5 equal chunks
3. For each chunk of data:

(a) Assign this chunk to be the Test set (1/5)

(b) Rest of data = Train set (4/5)

(c) Train model on the train set

(d) Test model on test set

(e) Record the score + discard model

4. Average results to produce single estimation

The $scikit - learn$ library was used to do K-fold, as it has an already implemented module. In order to call the function, $sklearn.model_selection.KFold()$ was used, where the input is the data and the number of fold to do, in this case, 5.

5.4 Experiment 1

The first experiment consists of forecasting the 1-day-ahead Covariance Matrix of the portfolio. As previously explained, the models implemented to perform this task can be divided into five different ones. 1-day-ahead forecasts are the simplest kind of prediction as the model must estimate what the next number in the series is. In this case, given the past M Cholesky elements corresponding to the covariance of two assets of the M past covariance matrices, the task is to predict the element in the next matrix. In order to measure how accurate each model is, the MSE, MAE and R^2 are measured.

In Appendix A, some plots showing the predictions made by all the models in Experiment 1 are shown, as well as plots containing the errors (MSE, MAE, R-squared) for each model in each Cholesky element.

Table 5.1 gathers the results of the experiments run and the values of the metrics obtained for Experiment 1, where SVR, GA-SVR and LR refer to Support Vector Regression, Genetic Algorithm - Support Vector Regression and Linear Regression:

Model	Model characteristics	Lag	MSE	MAE	R-squared
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	1	0.0791	0.1838	0.7763
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	5	0.0804	0.1859	0.7699
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	15	0.0780	0.1845	0.7802
SVR	**Kernel = Polynomial Degree = 2, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.2749**	**0.3793**	**0.4108**
SVR	**Kernel = Polynomial Degree = 3, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.2186**	**0.3418**	**0.3856**
SVR	**Kernel = Polynomial Degree = 4, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.2666**	**0.3724**	**0.2548**
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon 0.001	1	0.0793	0.1846	0.88912
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon = 0.001	5	0.0799	0.1874	0.9089
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon = 0.001	15	0.0761	0.1848	0.9361
GA-SVR	**Kernel = RBF**	**1**	**0.0750**	**0.1835**	**0.9489**
GA-SVR	**Kernel = RBF**	**5**	**0.0746**	**0.1833**	**0.9623**
GA-SVR	**Kernel = RBF**	**15**	**0.0743**	**0.1832**	**0.9711**
LR		1	0.1846	0.1934	0.7416
LR		5	0.1874	0.1976	0.7689
LR		15	0.1848	0.1984	0.7399

Table 5.1: Table of results for the multiple SVR's, GA-SVR's and LR's for Experiment 1.

In the first experiment, we can see the best results are obtained by the GA-SVR models. The best one is the 15-day lag version (with an R^2 value of 0.9711, MSE of 0.0743 and MAE of 0.1832), followed by the 5-day lag and the 1-day lag versions of GA-SVR. The worst performances are achieved by the Polynomial Kernels SVR, with R^2 values of 0.4108, 0.3856 and 0.2548 for polynomials of degrees 2, 3 and 4. After the three GA-SVR models, the SVRs with RBF kernels are the best performing models, being the 15-day lag, the best performing model, then the 5-days and then the 1-day. The next set of models in the performance ranking is the Linear SVRs, but in this case, the order of highest to lowest performing models is 15-day, 1-day and 5-day lag versions models. Finally, after the Linear SVR and before the Polynomial Kernels SVRs comes the benchmark, Linear Regression. This linear model achieved the highest performance score with the 5-day lag version, followed by the 1-day and then 15-day.

5.5 Experiment 2

The second Experiment has the same layout and settings as the first one. In this case, the prediction is done for 2-days-ahead, which means that at time t, each model regresses on the following M Cholesky elements $C_M = \{C_{t-M}, C_{t-M+1}, ..., C_{t-M+(M-1)}, C_t\}$ and tries to predict C_{t+2}. Table 5.2 shows the MSE, MAE and R-squared values for each of the models, which again serves as a classification of the best and worst models.

In Appendix B, some plots showing the predictions made by all the models in Experiment 1 are shown, as well as plots containing the errors (MSE, MAE, R-squared) for each model in each Cholesky element.

Table 5.2 gathers the results of the experiments run and the values of the metrics obtained for Experiment 2:

Model	Model characteristics	Lag	MSE	MAE	R-squared
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	1	0.1149	0.2323	0.7634
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	5	0.1159	0.2332	0.7544
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	15	0.1135	0.2310	0.7637
SVR	**Kernel = Polynomial Degree = 2, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.2754**	**0.3784**	**0.3877**
SVR	**Kernel = Polynomial Degree = 3, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.2294**	**0.3479**	**0.3745**
SVR	**Kernel = Polynomial Degree = 4, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.2704**	**0.3745**	**0.2323**
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon 0.001	1	0.1156	0.2324	0.8789
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon = 0.001	5	0.1141	0.2324	0.8832
SVR	**Kernel = RBF Gamma = 1, C = 2 Epsilon = 0.001**	**15**	**0.1089**	**0.2283**	**0.9161**
GA-SVR	Kernel = RBF	1	0.0995	0.2278	0.9003
GA-SVR	**Kernel = RBF**	**5**	**0.0092**	**0.2273**	**0.9188**
GA-SVR	**Kernel = RBF**	**15**	**0.0084**	**0.2209**	**0.9367**
LR		1	0.1522	0.2549	0.7132
LR		5	0.1517	0.2634	0.7198
LR		15	0.1548	0.2579	0.7048

Table 5.2: Table of results for the multiple SVR's, GA-SVR's and LR's for Experiment 2.

Experiment 2 consisted of the same principle as Experiment 1 but forecasting two days ahead. In this case, all the GA-SVR versions do not outperform the rest. The best performing model is seen in GA-SVR with lag 15, followed by the lag 5 model. However, the RBF SVR with lag 15 outperforms GA-SVR with lag 1, which indicates that in the 2-days-ahead forecast, the standard hyperparameters used for the RBF SVR are good enough to outperform the lag 1 GA-SVR. After these four models, the next best-performing ones are the other two RBF SVR with lags 5 and 1. Similarly to Experiment 1, the Linear Kernels are next, with lag 15 being the best, then lag one and then lag 5. The Linear Regression (benchmark) occupies the next spot in the ranking, with lag five being the best, then lag one and then lag 15. Lastly, the Polynomial SVRs failed again to capture the dynamics of the data, and they vastly underperformed compared to the rest, with R^2 values as small as 0.2323 and MAE as high as 0.3745. A graphical representation of this can be seen in Appendix B.

5.6 Experiment 3

The third experiment is similar to the two previous ones. In this case, the task is to predict 3-days-ahead, which means that at time t, each model regresses on the following M Cholesky elements $C_M = \{C_{t-M}, C_{t-M+1}, ..., C_{t-M+(M-1)}, C_t\}$ and tries to predict C_{t+3}. Table 5.3 shows the MSE, MAE and R-squared values for each of the models, which again serves as a classification of the best and worst models.

In Appendix C, some plots showing the predictions made by all the models in Experiment 1 are shown, as well as plots containing the errors (MSE, MAE, R-squared) for each model in each Cholesky element.

Table 5.3 gathers the results of the experiments run and the values of the metrics obtained for Experiment 3.

Finally, Experiment 3 shows the models performance when the task of forecasting 3-days-ahead Cholesky elements of the Covariance Matrix of a portfolio is presented. In this case, the GA-SVR model with a lag of 15 days is again the highest performing model, with errors of 0.1193 and 0.2501 (MSE and MAE) and a R^2 value of 0.9204. We see

that in this experiment, the RBF SVR with lag 15 beats the GA-SVR with lag 1 (as in Experiment 2) but fails to outperform lag 5 and 15 GA-SVRs. The same ranking as in Experiment 2 is seen in this case, with the RBF SVRs outperforming the Linear SVRs, which outperformed the Linear Regression (benchmark). As previously mentioned, the last spot in the performance ranking is for the Polynomial SVRs, the one with degree 4 being the worst model in the experiment. Again, the plots for these are seen in Appendix C.

Model	Model characteristics	Lag	MSE	MAE	R-squared
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	1	0.1454	0.2667	0.7436
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	5	0.1454	0.2654	0.7381
SVR	Kernel = Linear Gamma = 1, C = 2 Epsilon = 0.001	15	0.1441	0.2650	0.7432
SVR	**Kernel = Polynomial Degree = 2, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.4567**	**0.5712**	**0.3389**
SVR	**Kernel = Polynomial Degree = 3, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.4691**	**0.5619**	**0.3471**
SVR	**Kernel = Polynomial Degree = 4, Gamma = 1 C = 2, Epsilon = 0.001**	**15**	**0.4781**	**0.5782**	**0.2169**
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon 0.001	1	0.1476	0.2671	0.8412
SVR	Kernel = RBF Gamma = 1, C = 2 Epsilon = 0.001	5	0.1416	0.2624	0.8553
SVR	**Kernel = RBF Gamma = 1, C = 2 Epsilon = 0.001**	**15**	**0.1391**	**0.2611**	**0.9007**
GA-SVR	Kernel = RBF	1	0.1398	0.2617	0.8993
GA-SVR	**Kernel = RBF**	**5**	**0.1276**	**0.2555**	**0.9111**
GA-SVR	**Kernel = RBF**	**15**	**0.1193**	**0.2501**	**0.9204**
LR		1	0.1507	0.2687	0.6921
LR		5	0.1501	0.2684	0.7011
LR		15	0.1512	0.2692	0.6975

Table 5.3: Table of results for the multiple SVR's, GA-SVR's and LR's for Experiment 3.

5.7 Overall models performances

This section compares the models' overall errors, including the three experiments. It is done to have a better understanding of how well the models predict the covariances of the different assets in general and to visually see where the models manage to outperform the benchmark. Following are 12 plots showing the average errors and standard deviations obtained for each model and the benchmark, and a brief comment on each one of them.

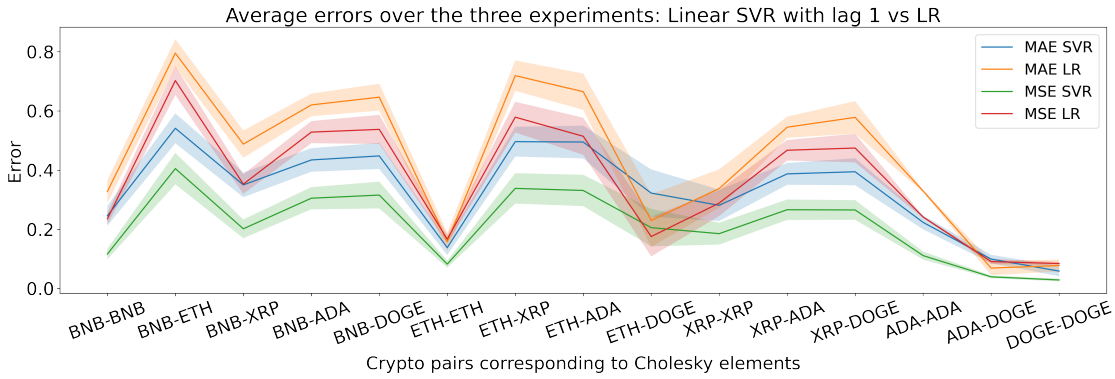


Figure 5.1: Average errors over the 3 experiments for each covariance matrix elements prediction done by the Linear SVR with lag 1.

The first plot shows the errors for Linear SVR with lag one and the benchmark Linear Regression with lag 1. Over the course of the three experiments, the Linear SVR outperformed the benchmark in nearly all covariance predictions, on average. It can be noted that it failed to have a lower error when predicting ETH-DOGE, ADA-DOGE and DOGE-DOGE. It is worth mentioning that some of the lowest errors (MSE and MAE) are obtained for the variances (covariances of the same assets result in variance of that asset) of the five assets. This means that these models perform better when predicting the variance of a single asset rather than the covariance of a pair. As for the standard deviations, the highest was obtained in ETH-DOGE prediction and the lowest in ETH-ETH, meaning that the results for the 1, 2 and 3 days-ahead forecast did not differ greatly. Finally, on average, the models do not differ greatly from each other when forecasting the variance of ETH, which results in a low error from all models.

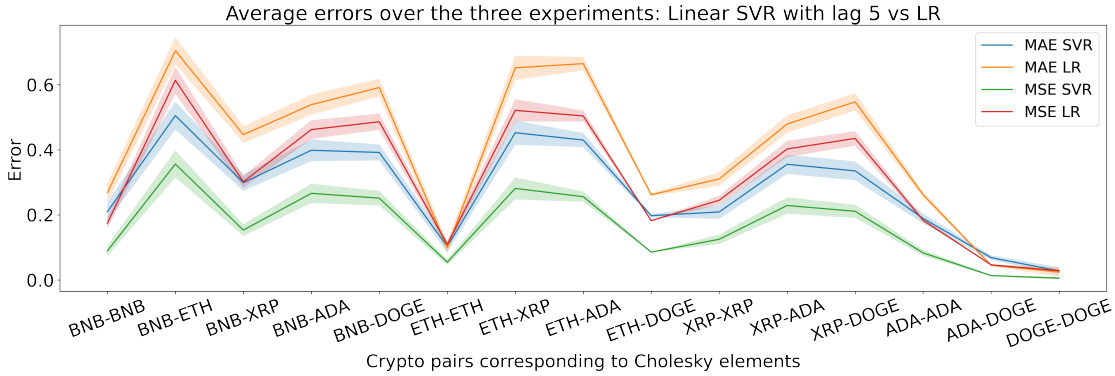


Figure 5.2: Average errors over the 3 experiments for each covariance matrix elements prediction done by the Linear SVR with lag 5.

The second plot shows the errors for the Linear SVR with lag five and the benchmark Linear Regression with lag 5. Overall, the pattern of the errors with respect to the pairs is similar to the previous model with lag 1. However, we do note that the standard deviation of the models decreases considerably, meaning that the prediction errors do not differ as much from one experiment to the other, and the model is thus more stable than the previous one. Another interesting point is that in the previous example, the ETH-DOGE predictions were the highest in standard deviation, whereas by incrementing the lag values to 5, the models obtain the smallest value for standard deviation in the covariance of this pair of assets.

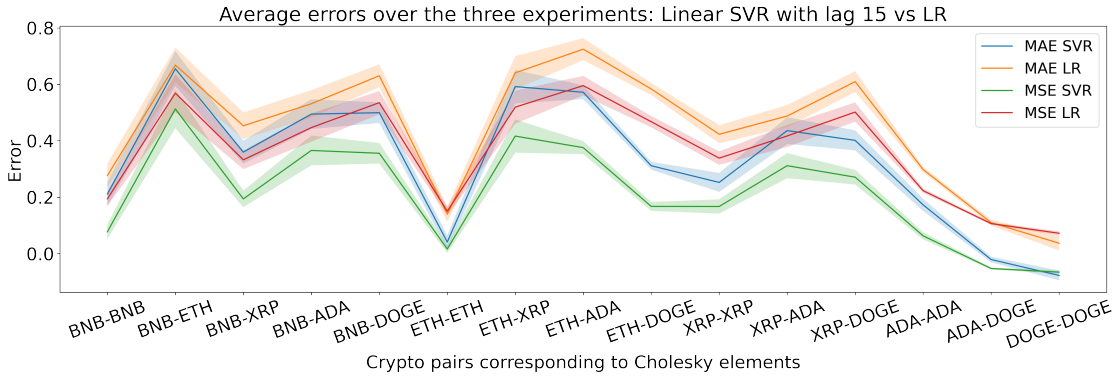


Figure 5.3: Average errors over the 3 experiments for each covariance matrix elements prediction done by the Linear SVR with lag 15.

The above plot shows the MSE and MAE for the Linear SVR lag 15 and the Linear

Regression lag 15. It can be seen that the models behave similarly to the two previous plots, and the shape formed is the same (that is, minimum errors in ETH-ETH covariance and a decrease in errors from XRP-DOGE covariance). Furthermore, the model's standard deviation increases with respect to lag 5 but is not as high as with lag 1.

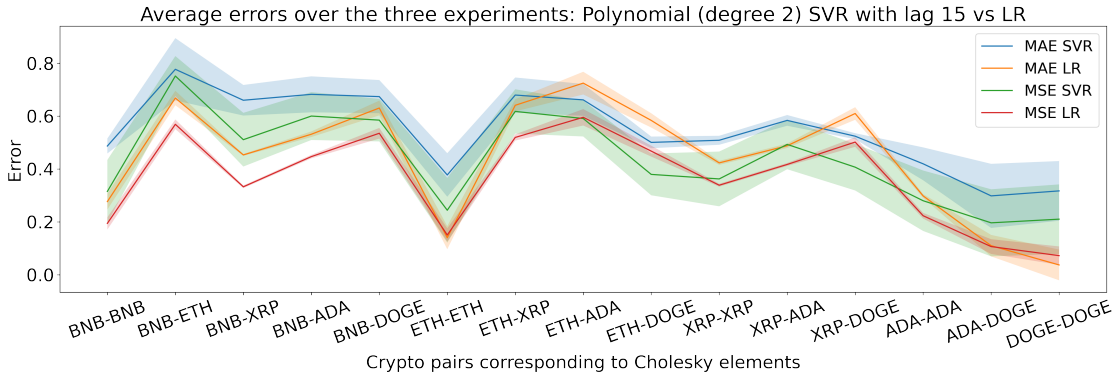


Figure 5.4: Average errors over the 3 experiments for each covariance matrix elements prediction done by the Polynomial (degree 2) Kernel SVR with lag 15.

In this plot, the errors for Polynomial (degree 2) Kernel SVR lag 15 and LR are shown. We can see an overall increase in error with the use of this kernel, which fails to beat the benchmark. As for the standard deviation, we see an increase in the SVR.

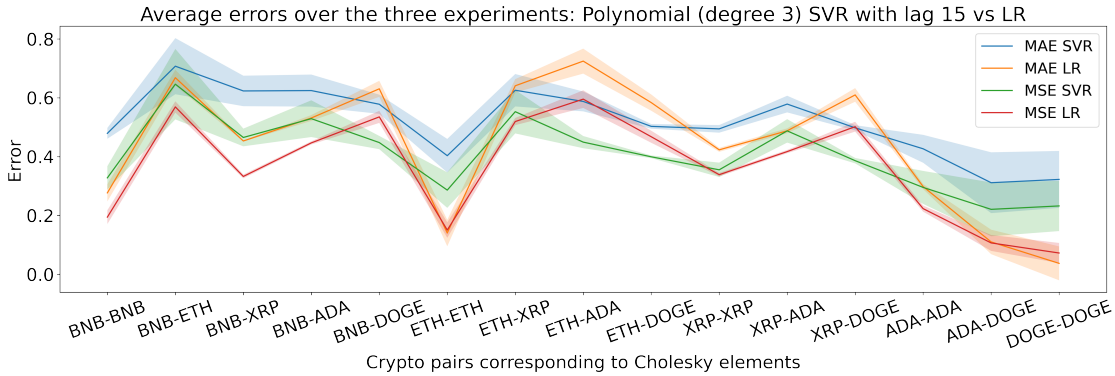


Figure 5.5: Average errors over the 3 experiments for each covariance matrix elements prediction done by the Polynomial (degree 3) Kernel SVR with lag 15.

Similar behaviour is seen in this plot with respect to the previous one. In this case, the

same model but with a polynomial kernel of degree 3 is tested and again fails to accurately predict the covariances of the portfolio.

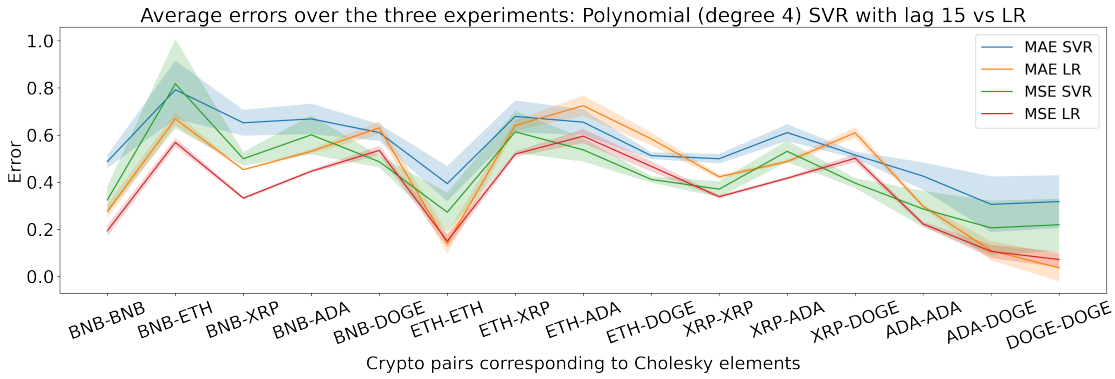


Figure 5.6: Average errors over the 3 experiments for each covariance matrix elements prediction done by the Polynomial (degree 4) Kernel SVR with lag 15.

This plot shows the benchmark's errors and the Polynomial (degree 4) Kernel SVRs. Again, similar errors are obtained with this model, which further proves that the data does not follow a polynomial function of these degrees.

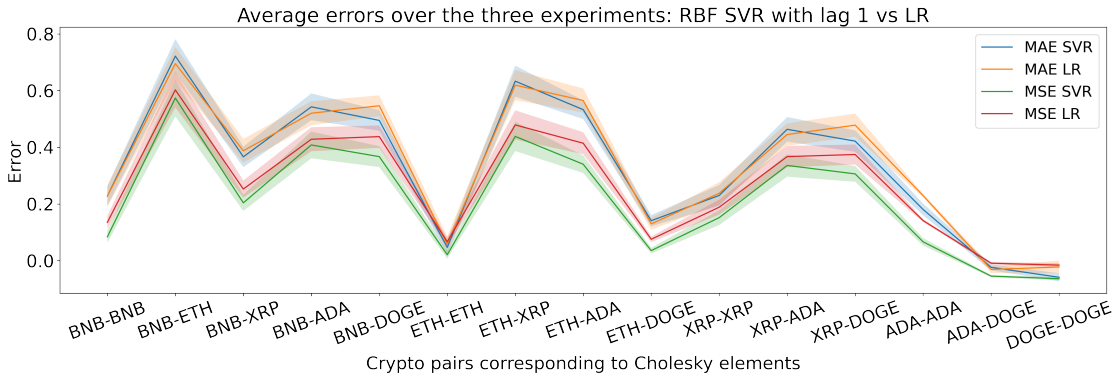


Figure 5.7: Average errors over the 3 experiments for each covariance matrix elements prediction done by the RBF SVR with lag 1.

In this plot, the errors for RBF Kernel SVR lag one and LR lag one are shown. We can see overall superior performance of the RBF model, having a lower MSE error in all covariances predictions overall than the LR. The standard deviation is relatively low compared to the polynomial SVRs, while the MAE errors do not differ greatly between both

models.

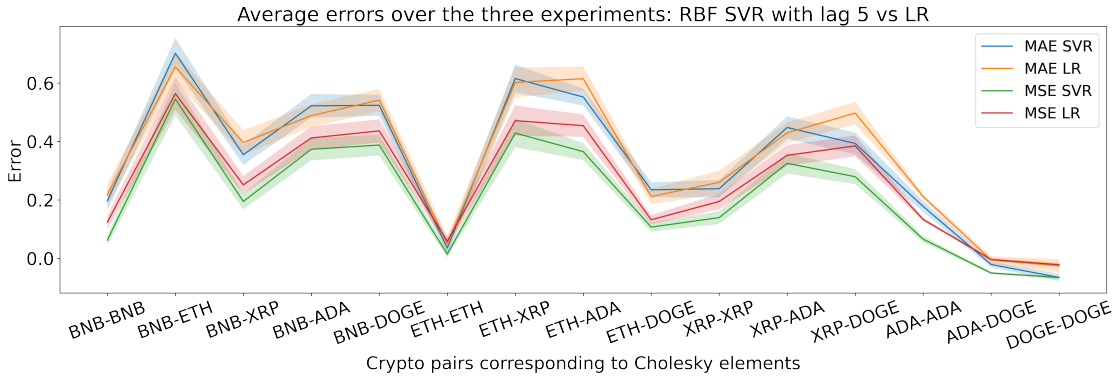


Figure 5.8: Average errors over the 3 experiments for each covariance matrix elements prediction done by the RBF SVR with lag 5.

The plot shows the same model as the previous one, but with an increase in lag values, being 5. In this case, the increase in lag values makes the difference in performance between the benchmark and the RBF SVR greater, and the overall error of the RBF SVR decrease with respect to the previous version. As for the standard deviation, it is slightly less than in the lag 1 model.

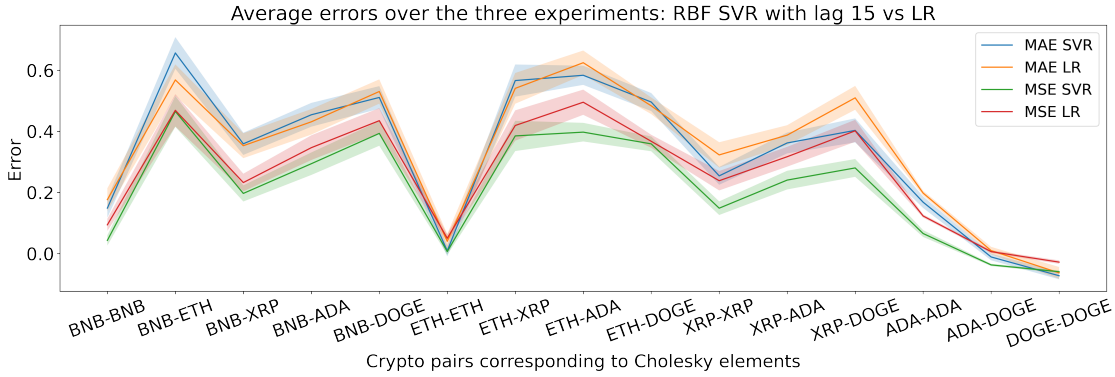


Figure 5.9: Average errors over the 3 experiments for each covariance matrix elements prediction done by the RBF SVR with lag 15.

The last RBF SVR model's errors, with lag 15, is shown above with its corresponding benchmark. It is interesting to see that despite a decrease in overall error from the SVR

model, the model fails to improve (with respect to the previous lag 1 and 5 versions) on forecasting the ETH-DOGE covariance. However, this model performs the best so far.

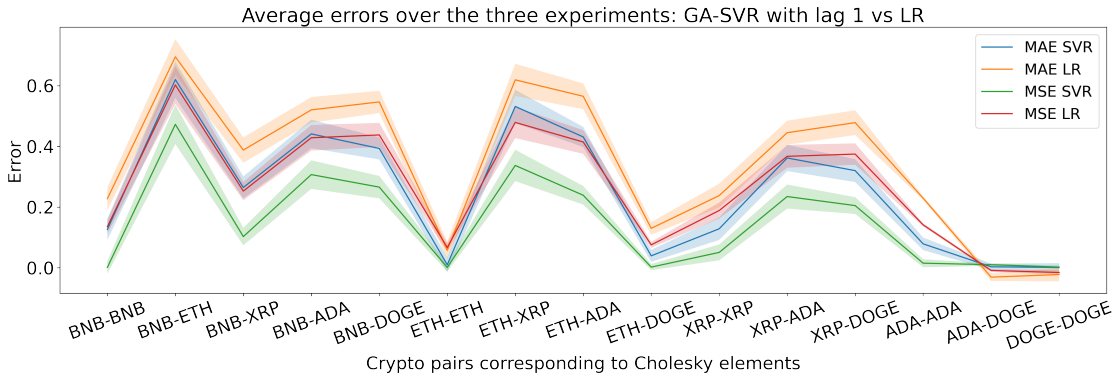


Figure 5.10: Average errors over the 3 experiments for each covariance matrix elements prediction done by the GA-SVR with lag 1.

The GA-SVR with lag one errors are shown in the plot above, as well as the Linear Regression ones. We can see a considerable improvement in the performance of this model. clearly outperforming LR in all forecasts. Furthermore, the standard deviation remains low with respect to previous models.

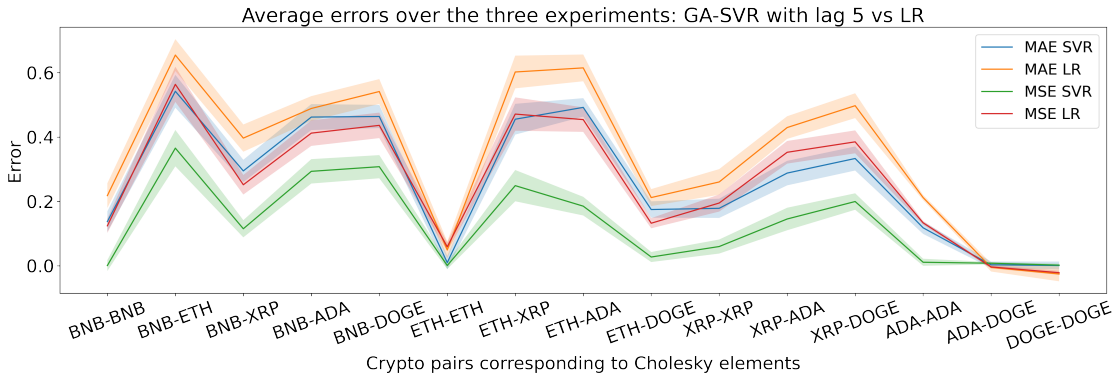


Figure 5.11: Average errors over the 3 experiments for each covariance matrix elements prediction done by the GA-SVR with lag 5.

The GA-SVR with lag five errors are shown in the plot above, as well as the Linear Regression ones. The plot is similar to the previous one, but we can see a considerable improvement in the error forecast of the XRP-ADA pair. In terms of the standard deviation,

it decreases slightly with respect to the previous model with lag 1.

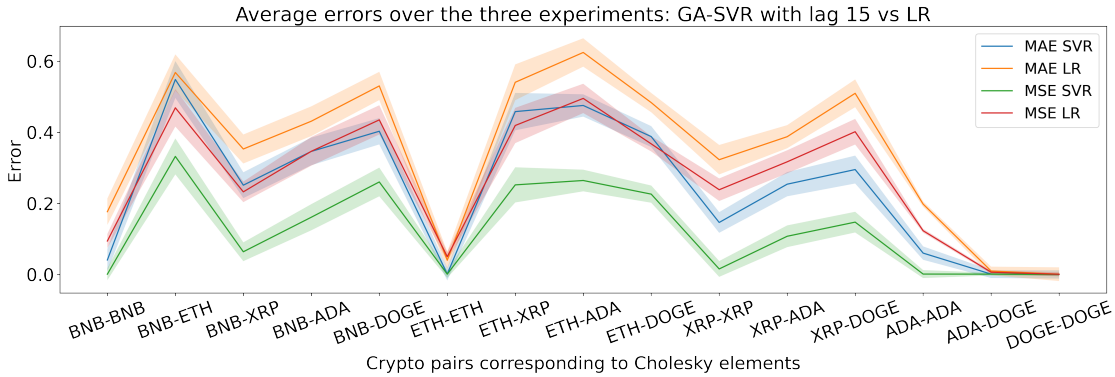


Figure 5.12: Average errors over the 3 experiments for each covariance matrix elements prediction done by the GA-SVR with lag 15.

This last plot contains the MAE and MSE errors for GA-SVR with lag 15 (as well as the benchmark). It is the best performing model in all experiments. It is similar to the previous one, but the addition of the lags makes it even more accurate.

5.8 Analysis of Results

This section will comment on the models' overall performances, comparing them with each other. Furthermore, we are going to analyse how the parameters and lag-values affect the models' performances.

Firstly, the best model in all experiments was the GA-SVR with a lag of 15, performing considerably better than the benchmark. The second best performing model was the lag 5 version of the GA-SVR, scoring slightly higher errors in all experiments than the lag 15 version. The third place in the ranking was disputed between the GA-SVR lag 1 and the RBF SVR lag 15, the latter outperforming the GA-SVR in 2 out of the 3 experiments.

The worst performances were achieved by the polynomial SVRs, which were the only models to score worse than the benchmark. The degree 2, 3 and 4 polynomials were the worst models in all three experiments, indicating that the covariances' dynamics do not

follow these types of functions.

Overall, the Experiments have shown similar patterns in all three cases, with a clear predominance in performance from the GA-SVR models. The lag 15 is the highest performing model in all experiments. Another point to note is that by comparing the R^2 values and the two errors between the different experiments, we can see an increase when the days ahead to predict increase. This is, the errors are greater in Experiment 3 than in Experiment 2 and 1, and errors in Experiment 2 are larger than in Experiment 1. This is because the difficulty in prediction increases when the days ahead to predict increase, as factors that influence the covariance and volatility of assets have more time to impact the portfolio, and covariance and volatility fluctuate more.

Furthermore, an increase in accuracy and decrease in errors have been seen in the nonlinear models (except Polynomial SVRs) when the lag was increased. For example, the RBF SVR and GA-SVR models were ranked in terms of performance, being the best-performing model the lag 15 one, then lag 5 and finally, lag 1. This indicates that the more points the model regresses, the better it is to make predictions, as more patterns can be identified when the series is larger. However, one thing to note is that the training speed increased considerably when the lag increased, thus having a trade-off between training speed and model performance. This trade-off between accuracy and training speed is also heavily noted when training the GA-SVR models because Genetic Algorithms take a long computing time.

Another interesting point previously mentioned is the fact that the prediction error tends to be lower in all experiments when forecasting the variance of an asset (diagonal entries of the covariance matrix), as opposed to the covariance of a pair. This can be due to the fact that covariance depends on two different time series and variance only in one, so the more time series included in one's prediction model, the more source of error in the forecast.

In general, the models can be split into linear and nonlinear models, and one of the objectives of this thesis was to explore the nonlinearities in a portfolio. We saw that the covariance could be predicted with a higher accuracy using nonlinear models (GA-SVR, RBF-SVR) rather than linear ones (Linear SVR, Linear Regression). This shows that the dynamics that govern the co-movements of the assets of this portfolio are better described

by models that capture nonlinear and linear relationships, as opposed to restricting the model to capture the linear relationships. Moreover, this means that by transforming the nonlinear data space into a linear space with the help of a nonlinear kernel, the data points are more easily predicted, as the nonlinearities are abundant in the data. This further proves our belief that the covariance of a portfolio is affected by nonlinear movements in the market, and risk management tools should take this type of risk into account. Further analysis of this research related to risk management is presented in the next Chapter.

5.9 Summary

The Testing and results Chapter has presented the three main experiments in this Thesis, as well as a comparison of the overall performances of the models and the results obtained. The following is a brief summary of the main points of the Chapter, ranging from the dataset used to the experimentation and analysis part.

First, some explanation of the dataset and its manipulation was given. The Chapter started with a section on the dataset, where the Cryptocurrencies used were presented, as well as the period of time that their time series covered. Some further explanation on the data format was also given, such as the elements each model regressed on and how the Covariance Matrix was broken down into elements of an array.

The next section was on Cross-Validation, where the 5-fold Cross Validation method was explained. This part was included in a separate section because it demonstrates the models will perform similarly on unseen datasets, which is crucial for other researchers that want to reproduce the results in this work.

The third section presented Experiments 1, 2 and 3. They each corresponded to a different day-ahead forecast (1, 2 and 3 day-ahead forecasts). Each Experiment included a table containing the models with different parameters and lagged versions, where the errors MSE and MAE were included, as well as the R^2 values. These were the ones used to measure how well each model fits the overall data. Finally, each Experiment section contains a reference to its corresponding Appendix (A, B and C, respectively). In each Appendix, two kinds of plots can be found per model. The first one shows the predictions of the model and

the benchmark compared with the actual data point. The second one shows the MSE and MAE for each Cholesky element prediction of the corresponding model and the benchmark.

Furthermore, the overall performances of the models were individually captured in different plots, where the prediction errors (MAE and MSE) and standard deviations were indicated. Each model's performance was measured against its corresponding benchmark, where it could be graphically seen how the models performed overall when predicting each covariance.

The last section was an in-depth analysis of the results of the three experiments. Several points were noted, such as the superiority in the forecasting of the nonlinear models or how a bigger lag value to regress changed the performance of the models.

Chapter 6

Conclusion and Future Work

This Chapter is the last block of the thesis and is divided into two sections. The first one includes the conclusions of the research, dividing them into the contributions to science and practical Machine Learning and the contributions to Portfolio Construction and Quantitative Finance. The second section presents the future work and in what directions should the future research focus.

6.1 Conclusions

In this thesis, we have proposed a time series analysis of cryptocurrencies as well as a comparative analysis of different regression models, namely Linear Regression, Support Vector Regression and Genetic-Algorithm Support Vector Regression. In this Chapter, some final conclusions are presented, which are divided into the more academic ones, focusing on the contributions to science and practical Machine Learning, and the more practical ones, focusing on Portfolio Allocation.

Concretely, at the beginning of this thesis, we introduced two questions we hoped to answer. We'll revisit each of the questions, in turn quickly presenting the conclusions of the thesis.

1. Whether by combining two Machine Learning models, we can improve the accuracy of a predictive model, and how is this reflected in our GA-SVR and simple SVR models.

We found that the best-performing models in all different scenarios and settings were

the two GA-SVR models with different lags. These two outperformed the same models without the Genetic Algorithm optimization. We can conclude that a model that combines another one for hyperparameter optimization outperforms the simple one, thus encouraging the merging of two models when possible. Furthermore, the lags used in the models that served as data points to regress on played an important role in the forecasting task, and in the case of the GA-SVR, the more values included in the lags, the more accurate the predictions were.

2. Whether there are nonlinearities in the covariance between assets of the cryptocurrency portfolio.
We knew that Machine Learning had had groundbreaking results in many areas, partly due to its ability to capture nonlinear data patterns. However, econometrics models have been sufficient in many financial applications, and many models avoid accounting for nonlinear relationships. Our experiments and results demonstrated that the nonlinear SVRs, except for the polynomial ones, outperformed the rest, including the Linear Regression, which served as a benchmark. These experiments demonstrated that there is a robust nonlinear component that describes how crypto-assets move with respect to each other, and it is essential to account for this in order to forecast the future covariance of a portfolio accurately. To the best of our knowledge, this research is the first one to present a novel approach based on a GA-SVR model for covariance matrix prediction on a cryptocurrency portfolio.

Finally, this work can benefit Portfolio Managers and many other financial institutions to better assess the risk in their investments. Capturing nonlinear relationships between financial assets is an arduous task that is essential for Asset Management firms to account for all types of risks taken within a portfolio. We have shown that linear models do not capture the actual dynamics of the data and that Machine Learning models capable of capturing nonlinear patterns are essential for the risk management task of a portfolio of assets. Our proposed model (GA-SVR) is not a black box, and can be thus explained and presented to investors or other parties in order to back investment decisions.

6.2 Future directions of work

It is believed that this thesis opens the doors for further research in multiple directions. This brief section discusses these directions, does a critique on some aspects of this work, and concludes the thesis.

Firstly, the dataset used in this research was not ideal. Only five different cryptocurrencies were included in the portfolio. The reasons for this were that not all cryptocurrencies listed in Yahoo Finance had the same length of time series, and the more assets included in the analysis, the more computing time was required for the predictive models. The five assets included were chosen based on their popularity and the length of their series. It is worth mentioning that cryptocurrencies are a relatively new asset, so many of them have not been trading for an extended period of time; thus, they lack public data. For these reasons, further research could be carried out in a more extensive portfolio, including traditional assets like equities, fixed income or foreign exchange.

Secondly, the original goal of this research was to compute the covariance matrix with Open, High, Low and Close (OHLC) prices of the assets, which was inspired by the research carried out in [32]. One of the benefits of computing the covariance this way is that it accounts for the intraday movements of the assets without requiring the actual intraday data, which in many cases, is hard to obtain. However, their results were irreproducible because the covariance matrix computed with the OHLC prices was not Positive Definite, as opposed to what was stated in their work, so a Cholesky decomposition could not be applied to it. For this reason, further research could be done on covariance matrix construction and the different methods available in the literature.

Focusing more on the regression models, some future research is encouraged to merge more models together. For example, the GA-SVR model could be combined with a model from the econometrics literature and explore if this combination better captures the dynamics of the time series. As combined models have proven to lead to better results, further research can be carried out on this topic. Furthermore, different lags can be explored, as we only tested lags 1, 5 and 15, and we concluded that the more lags, the better prediction accuracy for the GA-SVR models.

Finally, one idea that was explored in detail but not included due to lack of time was trying to include prediction information from predicted Cholesky elements into other predictions. For example, if there was an SVR model that predicted a value for a 1-day-ahead Cholesky element for BNB-ETH, and another model predicted another value for ETH-XRP, some information could be inferred from these in order to be included in the model that was in charge of predicting the 1-day-ahead Cholesky element for BNB-XRP, as it is correlated. $Scikit-learn$ has an in-built function $RegressorChain()$ that takes several models and makes a prediction in the order specified by the chain using all of the available features provided to the model, in addition to the predictions of models that are earlier in the chain.

Appendix A

Experiment 1 plots

In the following appendix, we attach a series of plots from Experiment 1. These are divided in two categories: the forecast and the errors plot. The first type of plot shows the fit of the models (SVR/GA-SVR and LR) and the actual data. The second one shows the MSE and MAE of the models (SVR/GA-SVR and LR) for each Cholesky element. The Appendix is divided in subsections of each model (in the format: *model* | number of lagged elements in which to regress | days-ahead to forecast), which will contain the two types of plot described above that correspond to the model.

Linear SVR | Lag 1 | 1-day-ahead forecast

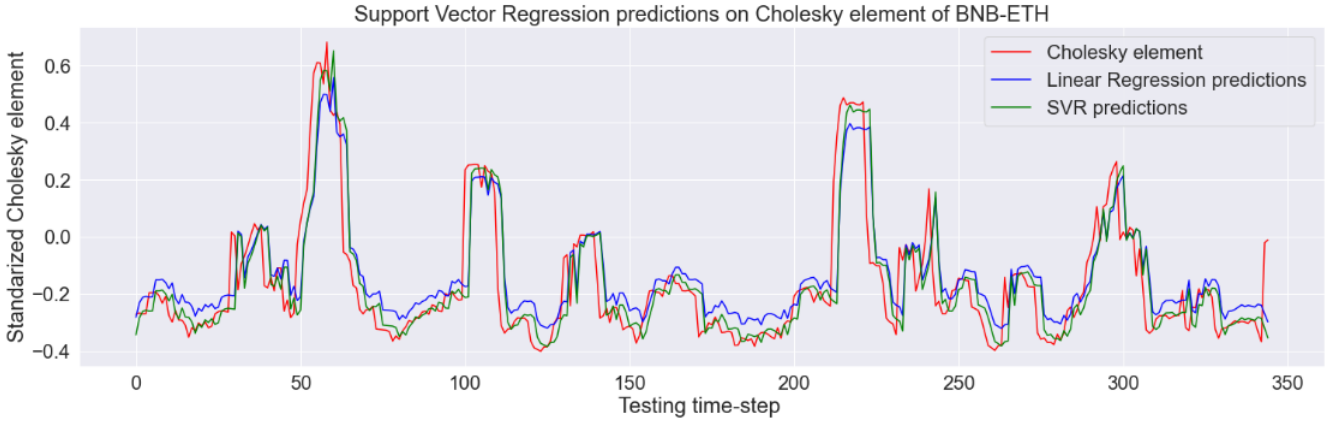


Figure A.1: Plot showing the predicted values of Linear SVR | Lag 1 | 1-day-ahead forecast, LR and the true datapoints.

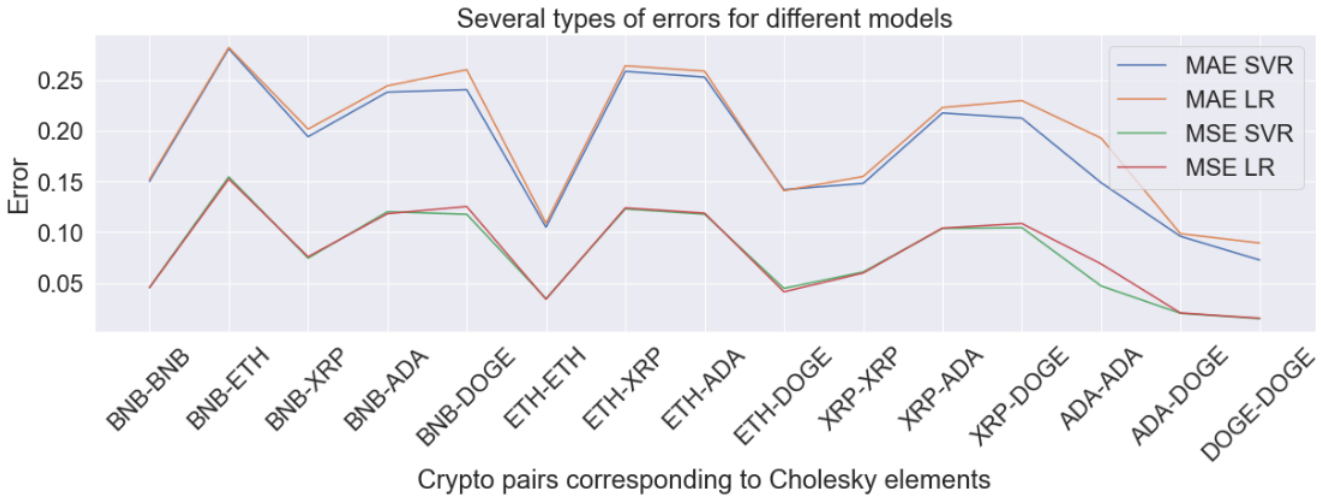


Figure A.2: Plot showing the MSE and MAE of Linear SVR | Lag 1 | 1-day-ahead forecast and LR.

Linear SVR | Lag 5 | 1-day-ahead forecast

Figure A.3: Plot showing the predicted values of Linear SVR | Lag 5 | 1-day-ahead forecast, LR and the true datapoints.

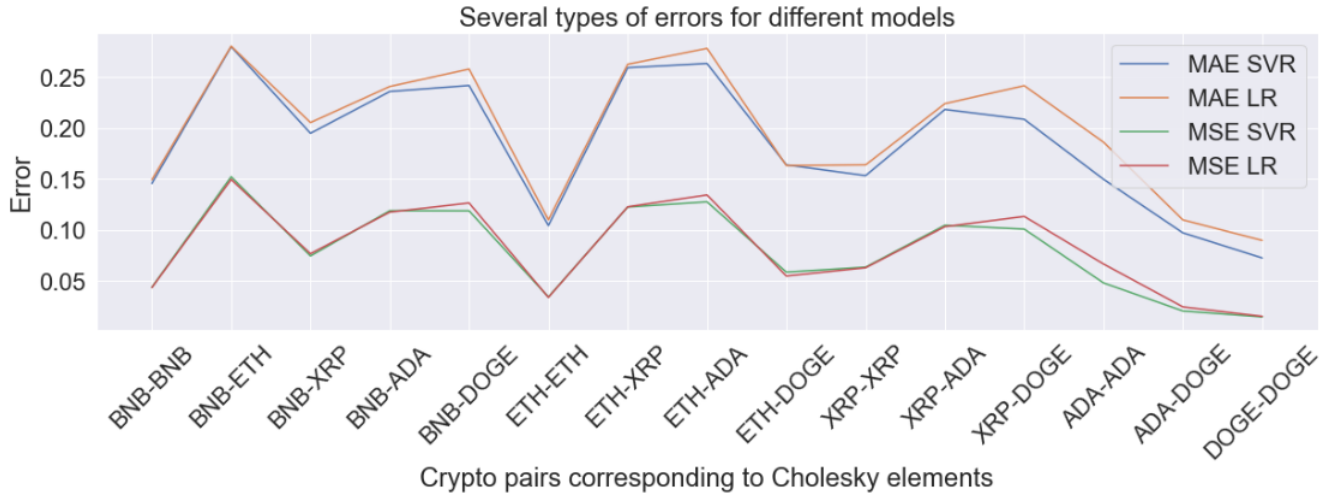


Figure A.4: Plot showing the MSE and MAE of Linear SVR | Lag 5 | 1-day-ahead forecast and LR.

Linear SVR | Lag 15 | 1-day-ahead forecast

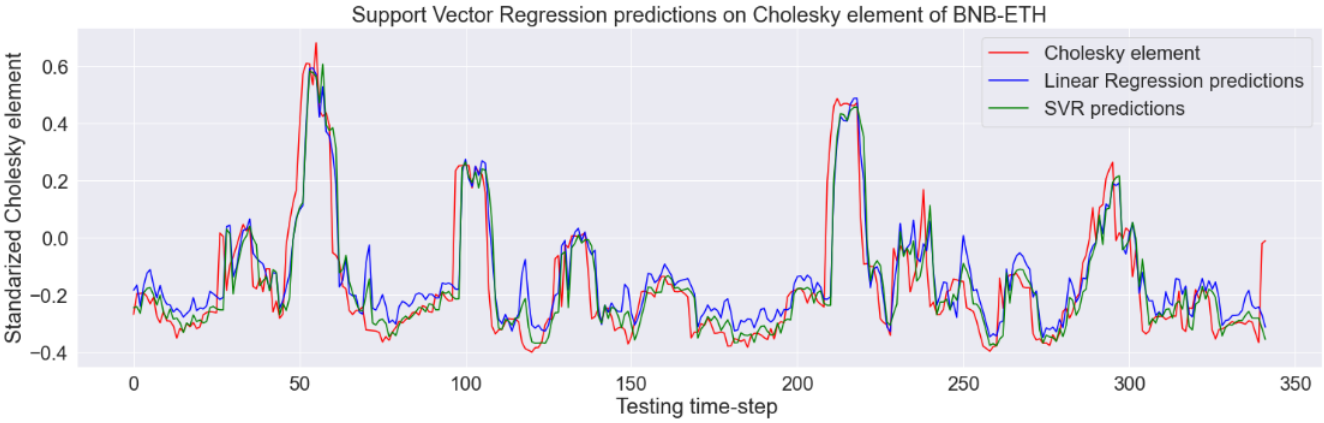


Figure A.5: Plot showing the predicted values of Linear SVR | Lag 15 | 1-day-ahead forecast, LR and the true datapoints.

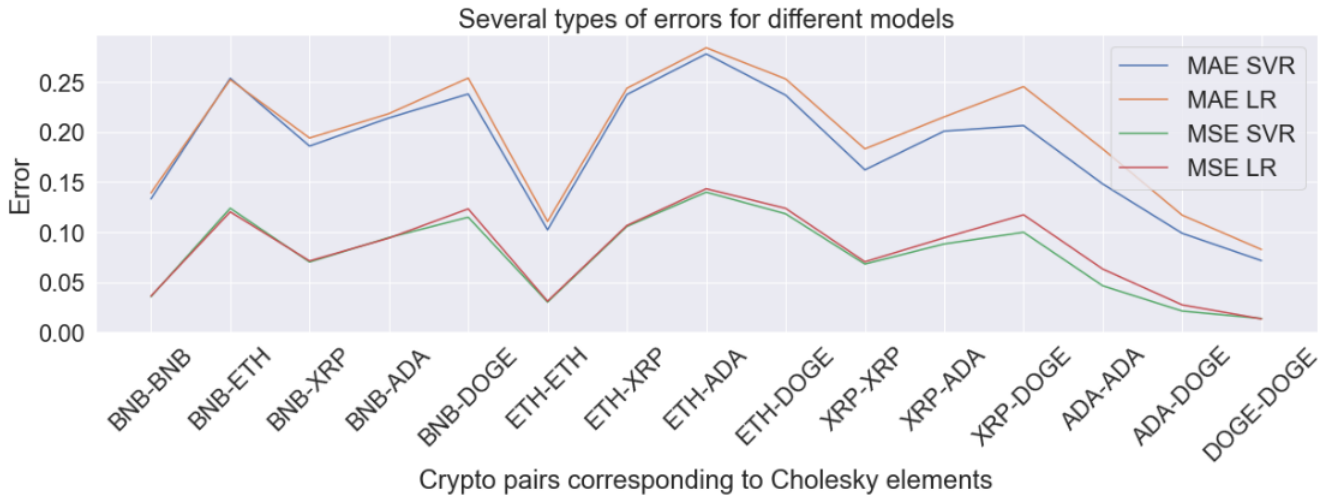


Figure A.6: Plot showing the MSE and MAE of Linear SVR | Lag 15 | 1-day-ahead forecast and LR.

Polynomial (degree 2) SVR | Lag 15 | 1-day-ahead forecast

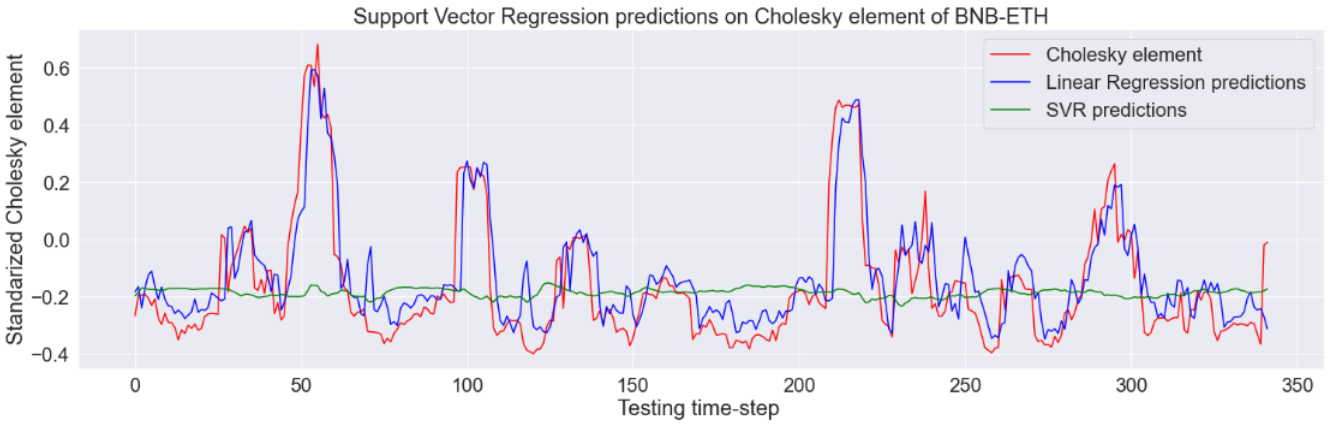


Figure A.7: Plot showing the predicted values of Polynomial (degree 2) SVR | Lag 15 | 1-day-ahead forecast, LR and the true datapoints.

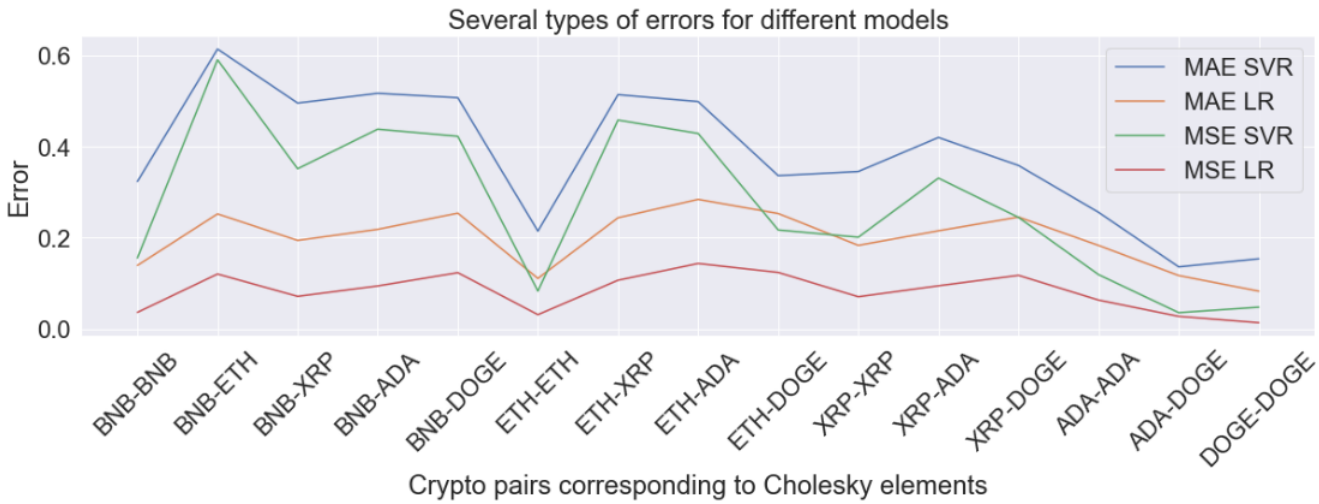


Figure A.8: Plot showing the MSE and MAE of Polynomial (degree 2) SVR | Lag 15 | 1-day-ahead forecast and LR.

Polynomial (degree 3) SVR | Lag 15 | 1-day-ahead forecast

Figure A.9: Plot showing the predicted values of Polynomial (degree 3) SVR | Lag 15 | 1-day-ahead forecast, LR and the true datapoints.

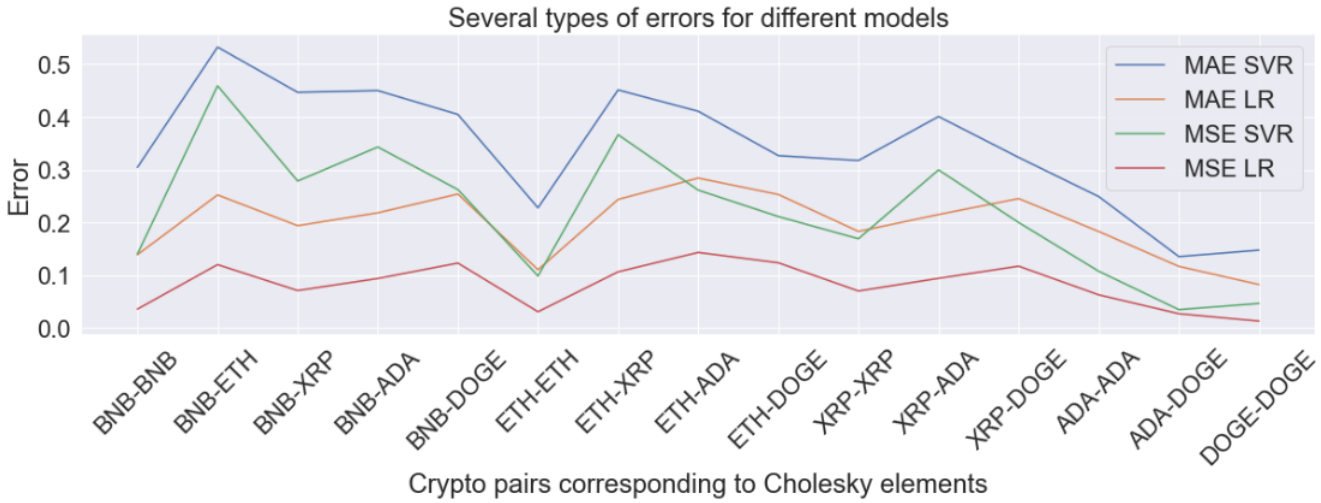


Figure A.10: Plot showing the MSE and MAE of Polynomial (degree 3) SVR | Lag 15 | 1-day-ahead forecast and LR.

Polynomial (degree 4) SVR | Lag 15 | 1-day-ahead forecast

Figure A.11: Plot showing the predicted values of Polynomial (degree 4) SVR | Lag 15 | 1-day-ahead forecast, LR and the true datapoints.

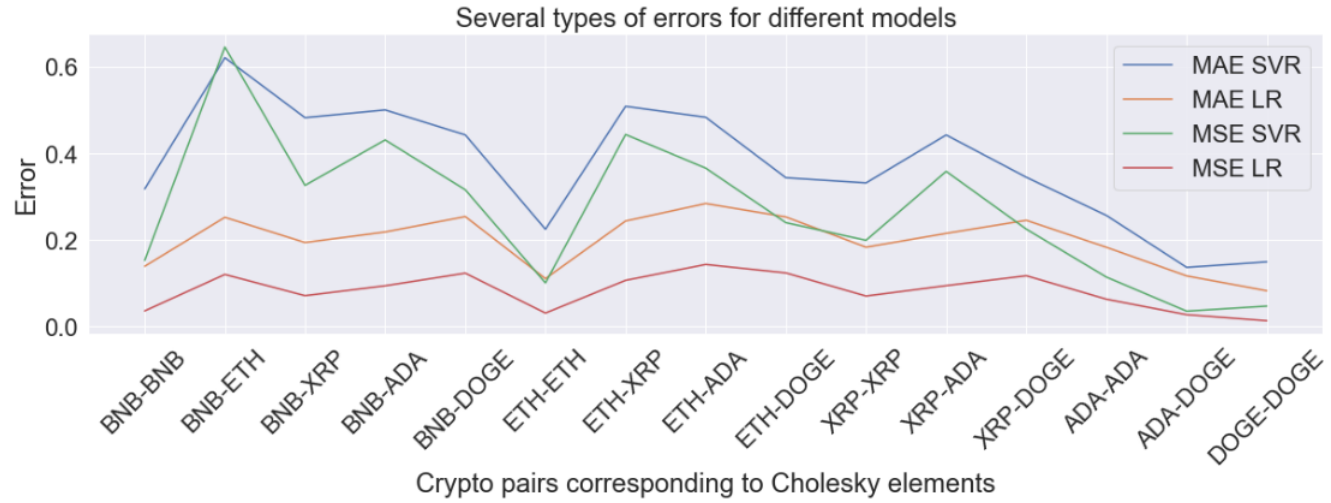


Figure A.12: Plot showing the MSE and MAE of Polynomial (degree 4) SVR | Lag 15 | 1-day-ahead forecast and LR.

RBF SVR | Lag 1 | 1-day-ahead forecast

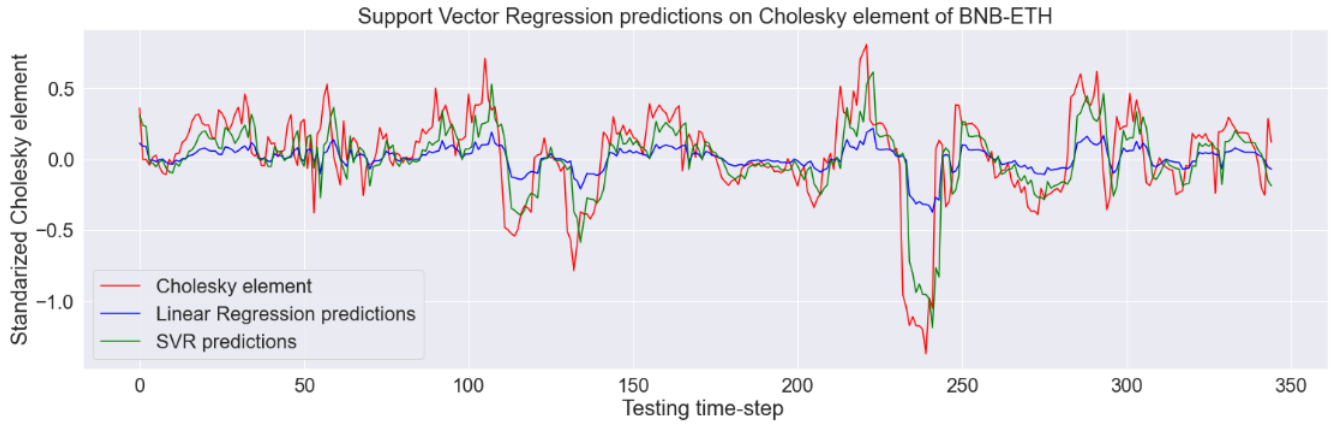


Figure A.13: Plot showing the predicted values of RBF SVR | Lag 1 | 1-day-ahead forecast, LR and the true datapoints.

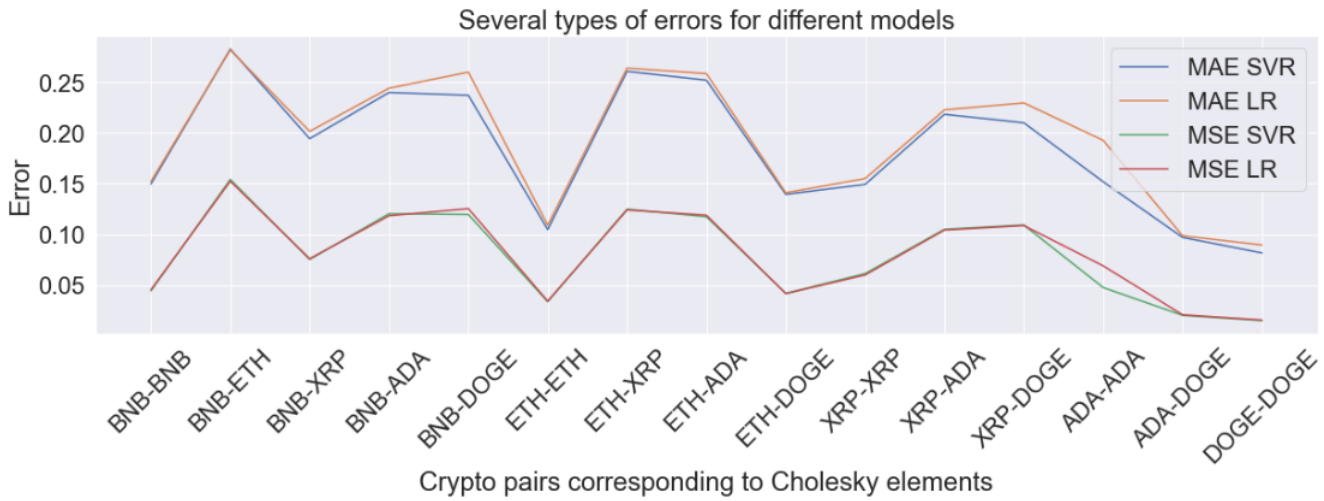


Figure A.14: Plot showing the MSE and MAE of RBF SVR | Lag 1 | 1-day-ahead forecast and LR.

RBF SVR | Lag 5 | 1-day-ahead forecast

Figure A.15: Plot showing the predicted values of RBF SVR | Lag 5 | 1-day-ahead forecast, LR and the true datapoints.

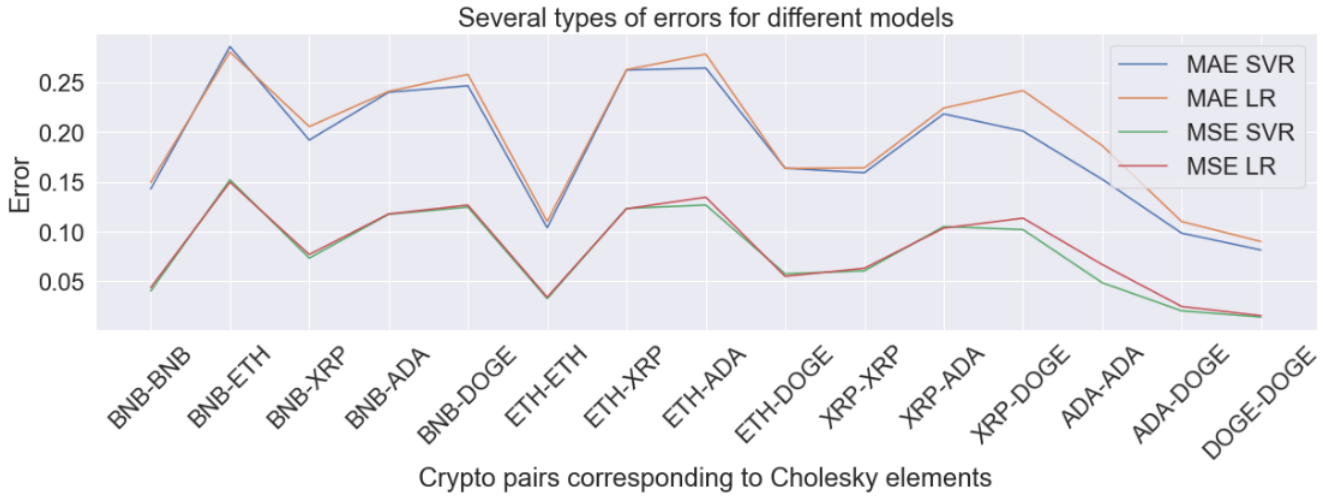


Figure A.16: Plot showing the MSE and MAE of RBF SVR | Lag 5 | 1-day-ahead forecast and LR.

RBF SVR | Lag 15 | 1-day-ahead forecast

Figure A.17: Plot showing the predicted values of RBF SVR | Lag 15 | 1-day-ahead forecast, LR and the true datapoints.

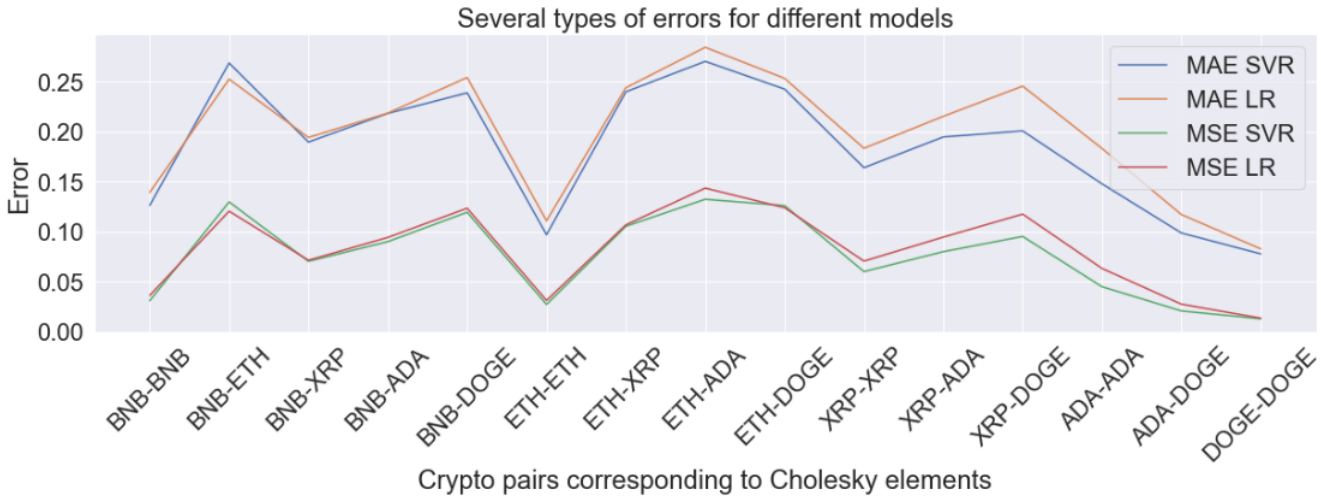


Figure A.18: Plot showing the MSE and MAE of RBF SVR | Lag 15 | 1-day-ahead forecast and LR.

Appendix B

Experiment 2 plots

Linear SVR | Lag 1 | 2-day-ahead forecast

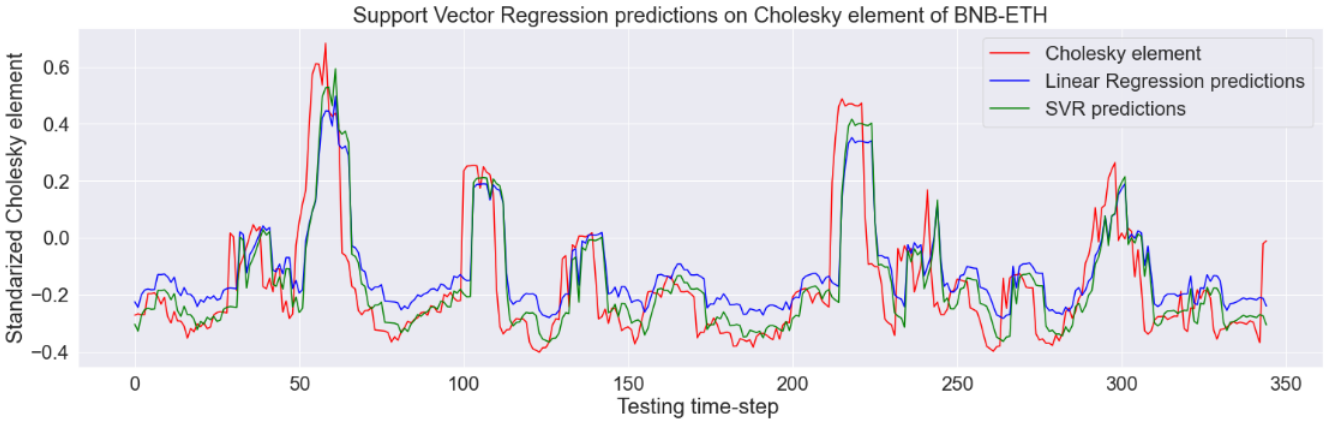


Figure B.1: Plot showing the predicted values of Linear SVR | Lag 1 | 2-day-ahead forecast, LR and the true datapoints.

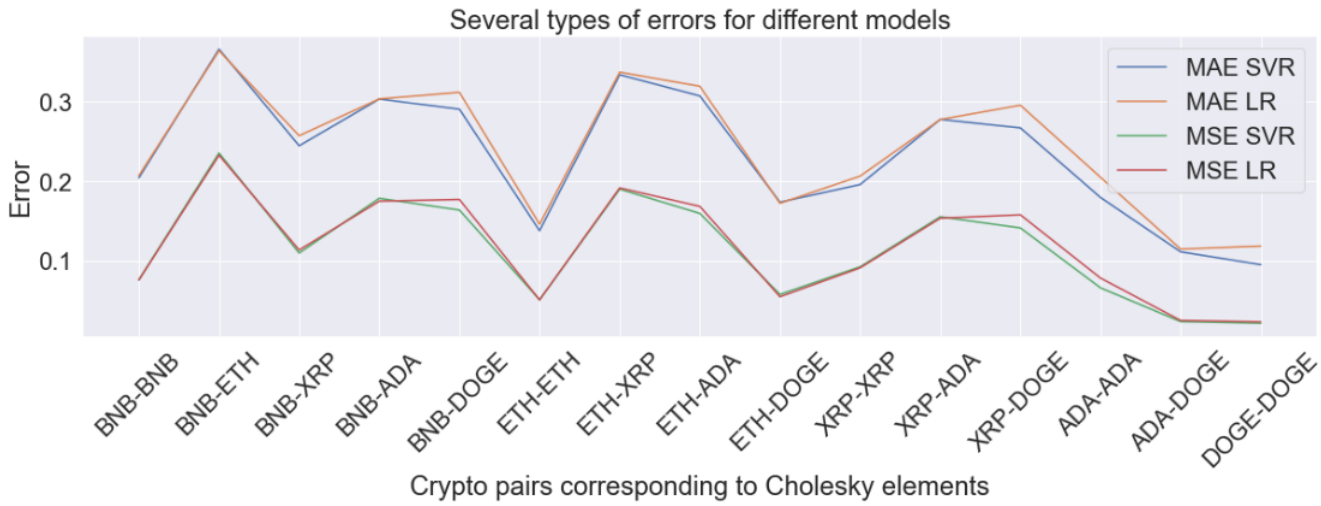


Figure B.2: Plot showing the MSE and MAE of Linear SVR | Lag 1 | 2-day-ahead forecast and LR.

Linear SVR | Lag 5 | 2-day-ahead forecast

Figure B.3: Plot showing the predicted values of Linear SVR | Lag 5 | 2-day-ahead forecast, LR and the true datapoints.

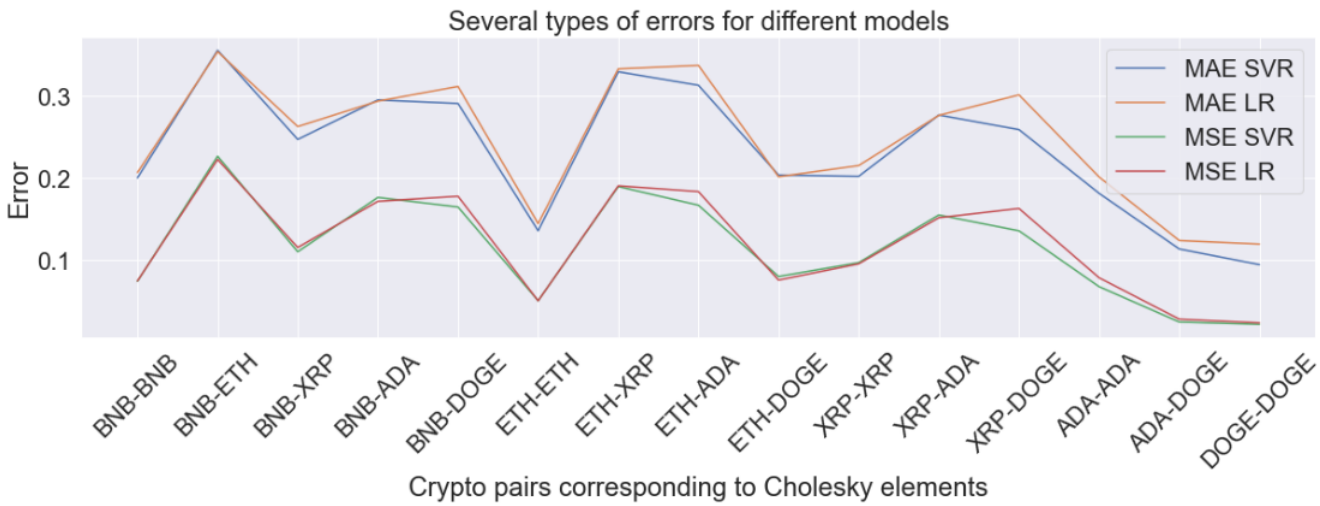


Figure B.4: Plot showing the MSE and MAE of Linear SVR | Lag 5 | 2-day-ahead forecast and LR.

Linear SVR | Lag 15 | 2-day-ahead forecast

Figure B.5: Plot showing the predicted values of Linear SVR | Lag 15 | 2-day-ahead forecast, LR and the true datapoints.

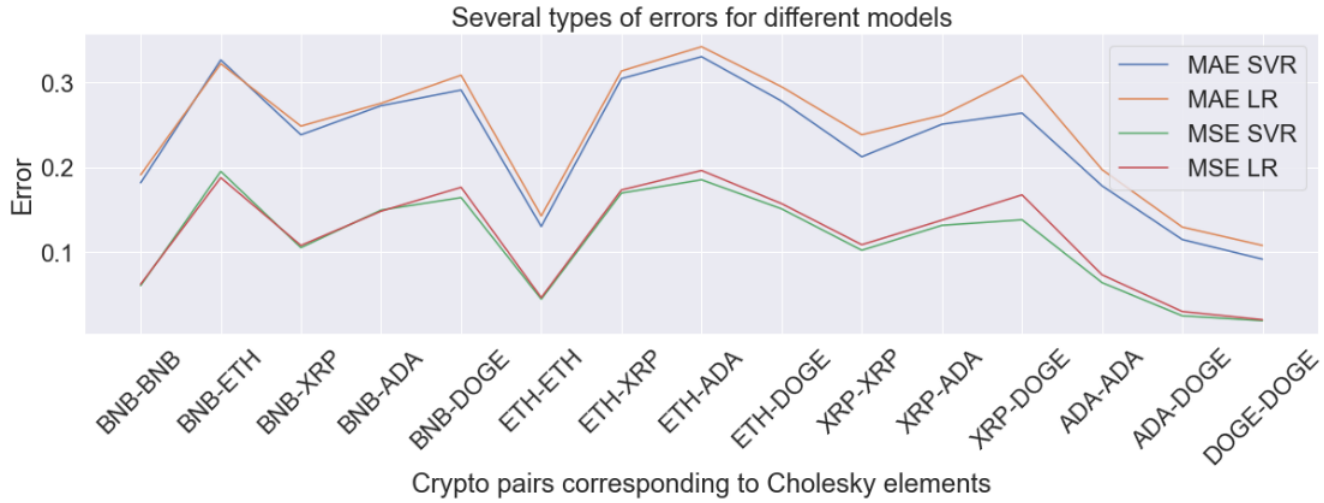


Figure B.6: Plot showing the MSE and MAE of Linear SVR | Lag 15 | 2-day-ahead forecast and LR.

Polynomial (degree 2) SVR | Lag 15 | 2-day-ahead forecast

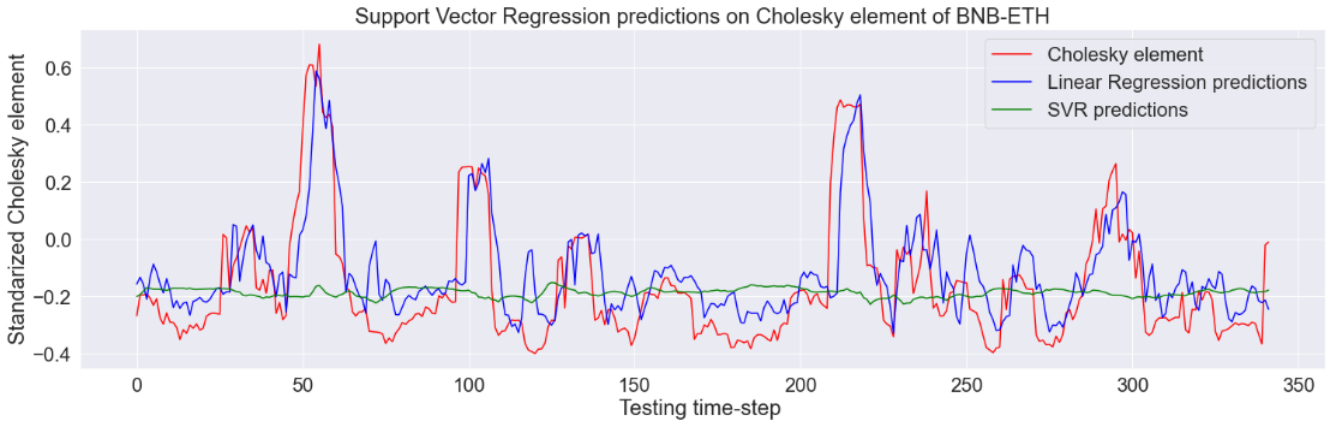


Figure B.7: Plot showing the predicted values of Polynomial (degree 2) SVR | Lag 15 | 2-day-ahead forecast, LR and the true datapoints.

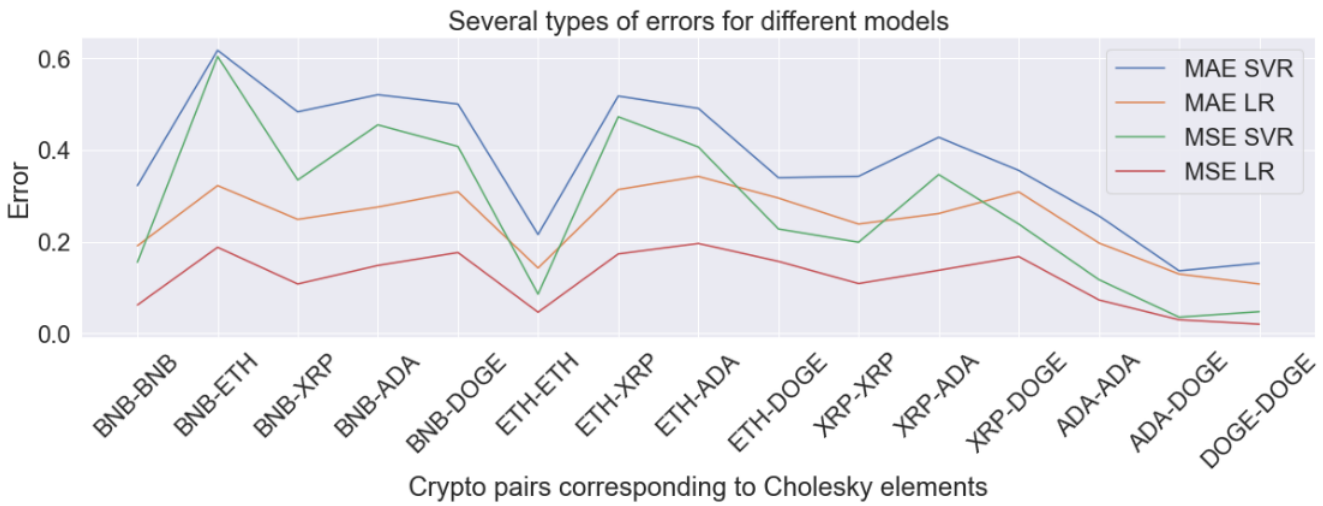


Figure B.8: Plot showing the MSE and MAE of Polynomial (degree 2) SVR | Lag 15 | 2-day-ahead forecast and LR.

Polynomial (degree 3) SVR | Lag 15 | 2-day-ahead forecast

Figure B.9: Plot showing the predicted values of Polynomial (degree 3) SVR | Lag 15 | 2-day-ahead forecast, LR and the true datapoints.

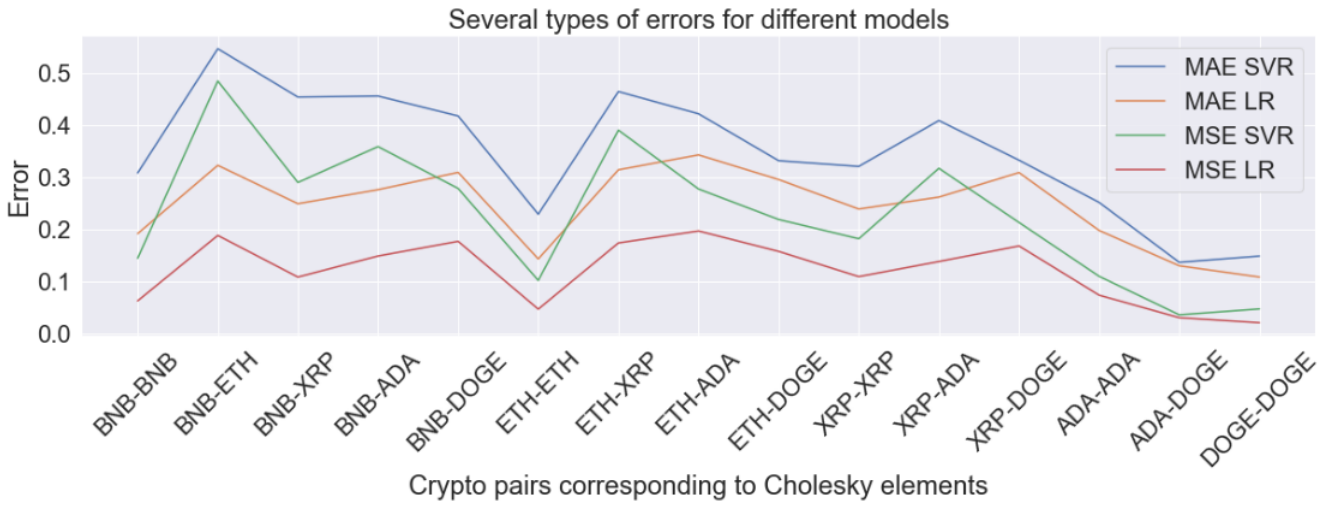


Figure B.10: Plot showing the MSE and MAE of Polynomial (degree 3) SVR | Lag 15 | 2-day-ahead forecast and LR.

Polynomial (degree 4) SVR | Lag 15 | 2-day-ahead forecast

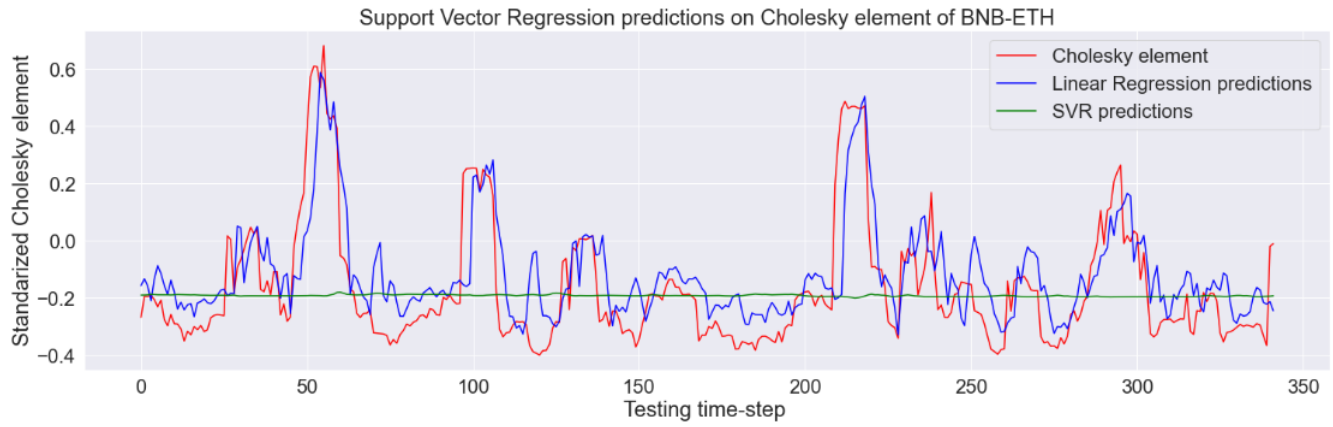


Figure B.11: Plot showing the predicted values of Polynomial (degree 4) SVR | Lag 15 | 2-day-ahead forecast, LR and the true datapoints.

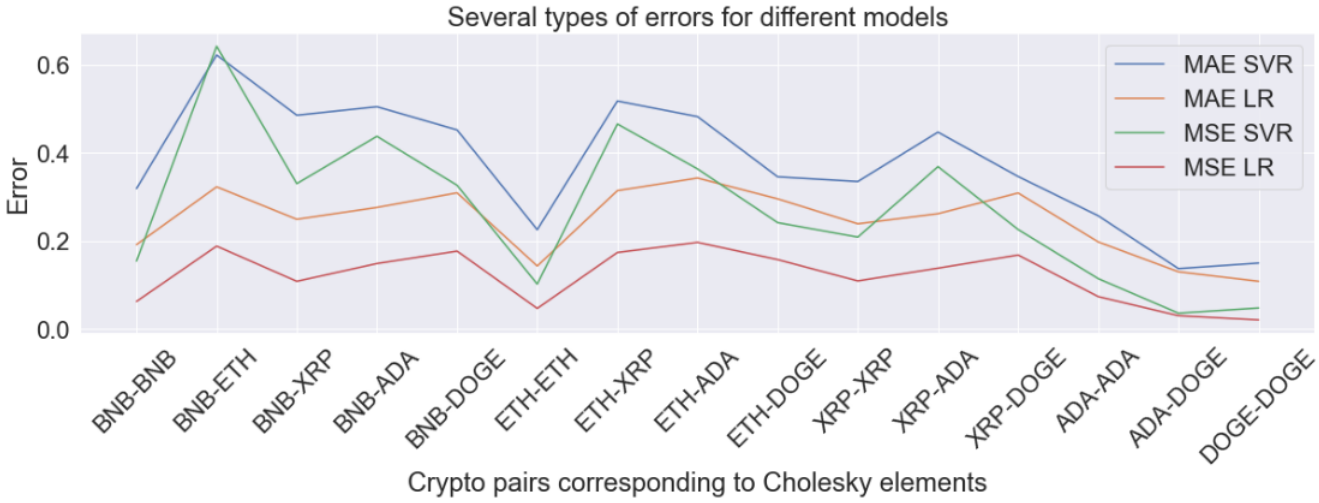


Figure B.12: Plot showing the MSE and MAE of Polynomial (degree 4) SVR | Lag 15 | 2-day-ahead forecast and LR.

RBF SVR | Lag 1 | 2-day-ahead forecast

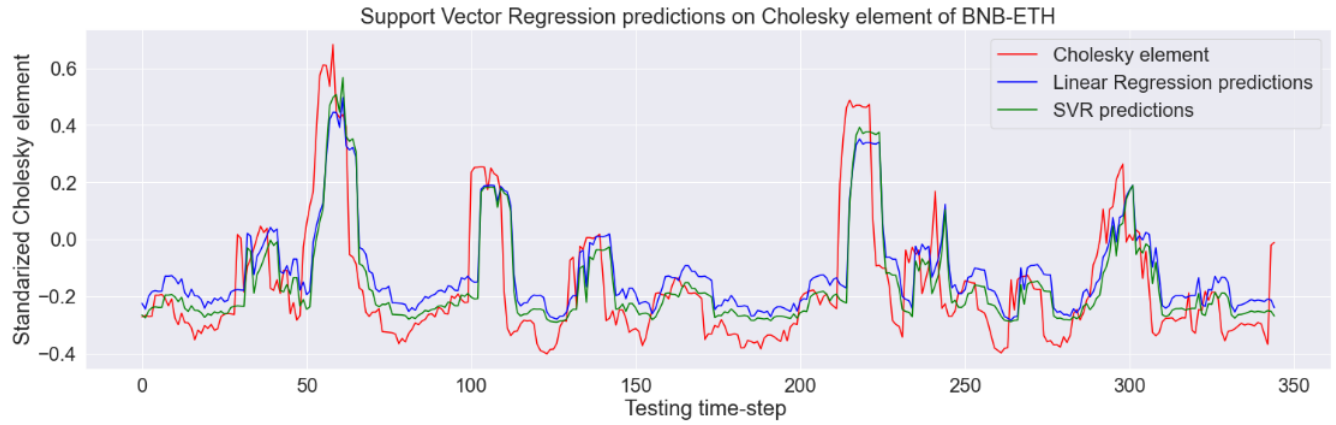


Figure B.13: Plot showing the predicted values of RBF SVR | Lag 1 | 2-day-ahead forecast, LR and the true datapoints.

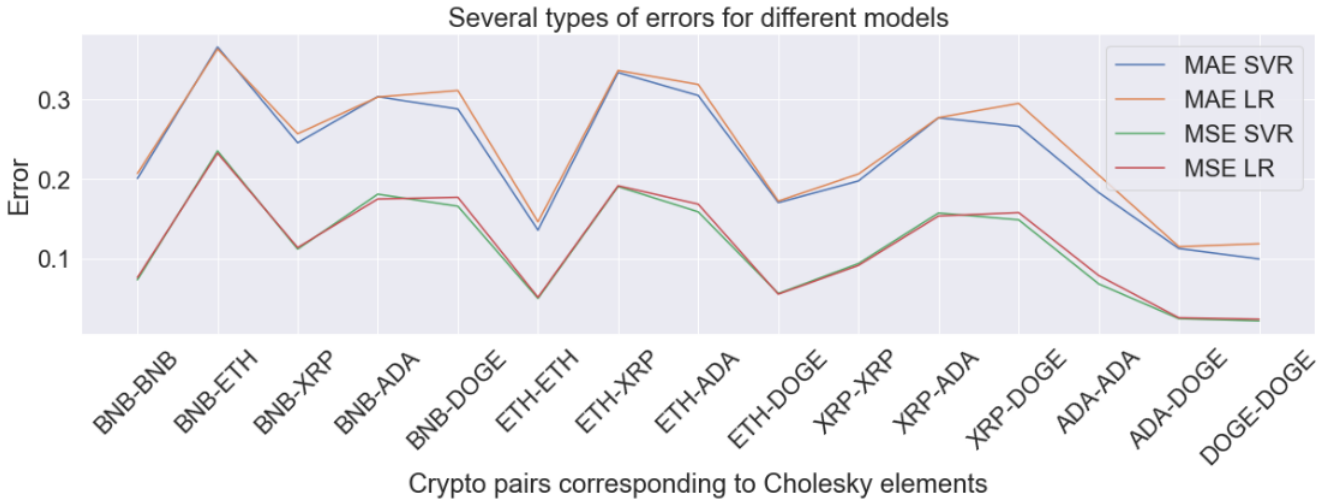


Figure B.14: Plot showing the MSE and MAE of RBF SVR | Lag 1 | 2-day-ahead forecast and LR.

RBF SVR | Lag 5 | 2-day-ahead forecast

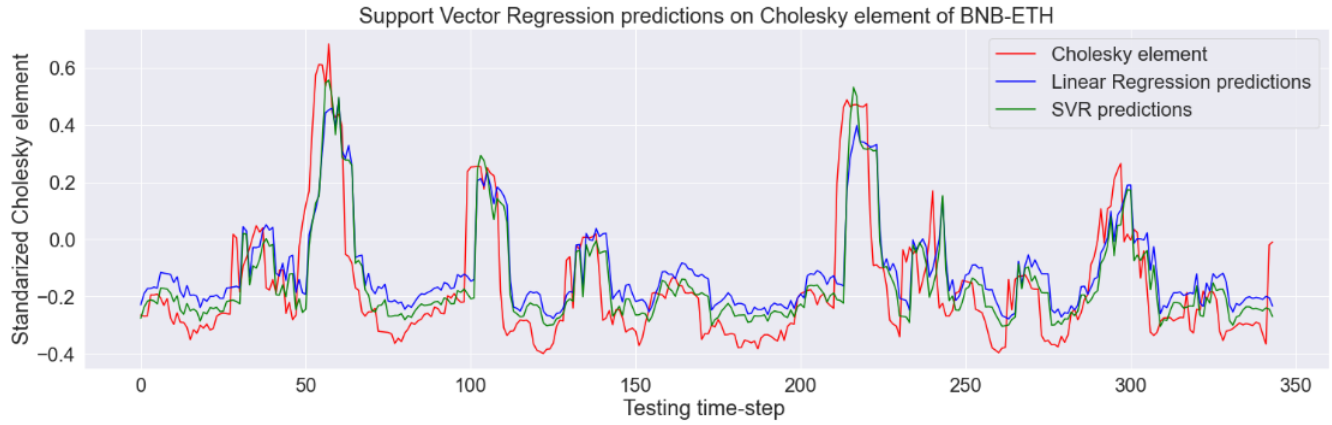


Figure B.15: Plot showing the predicted values of RBF SVR | Lag 5 | 2-day-ahead forecast, LR and the true datapoints.

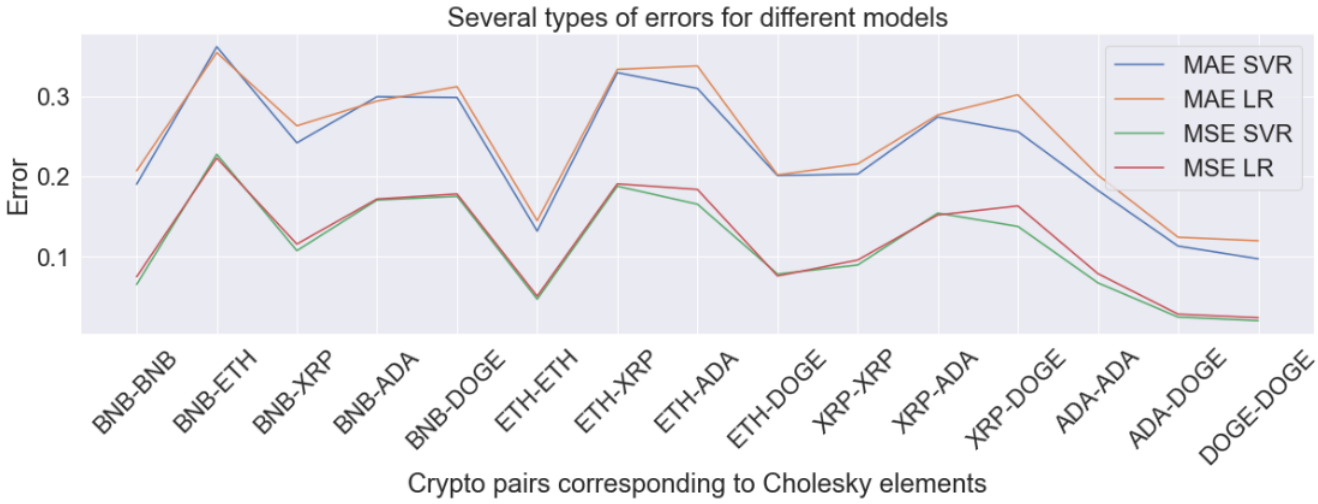


Figure B.16: Plot showing the MSE and MAE of RBF SVR | Lag 5 | 2-day-ahead forecast and LR.

RBF SVR | Lag 15 | 2-day-ahead forecast

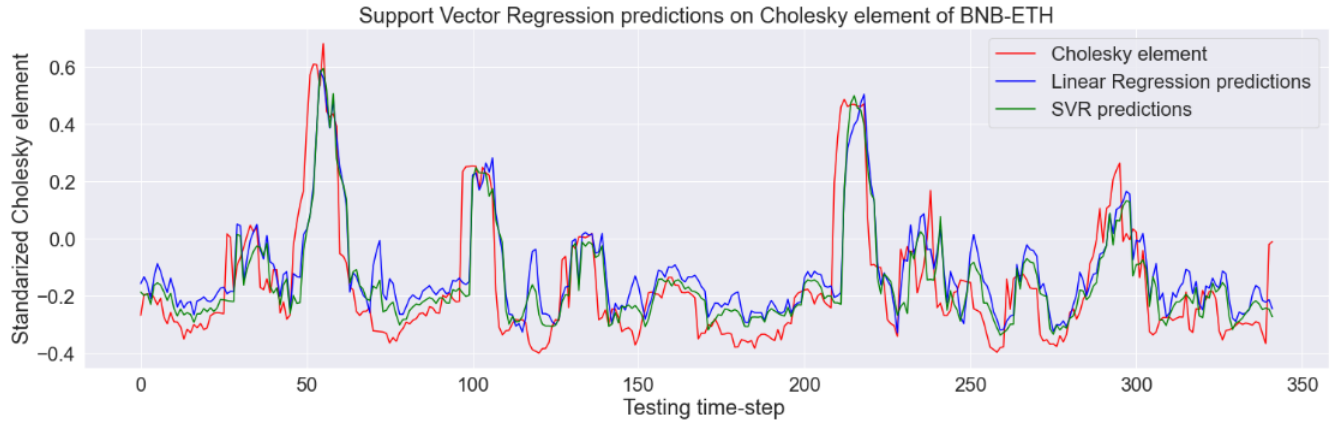


Figure B.17: Plot showing the predicted values of RBF SVR | Lag 15 | 2-day-ahead forecast, LR and the true datapoints.

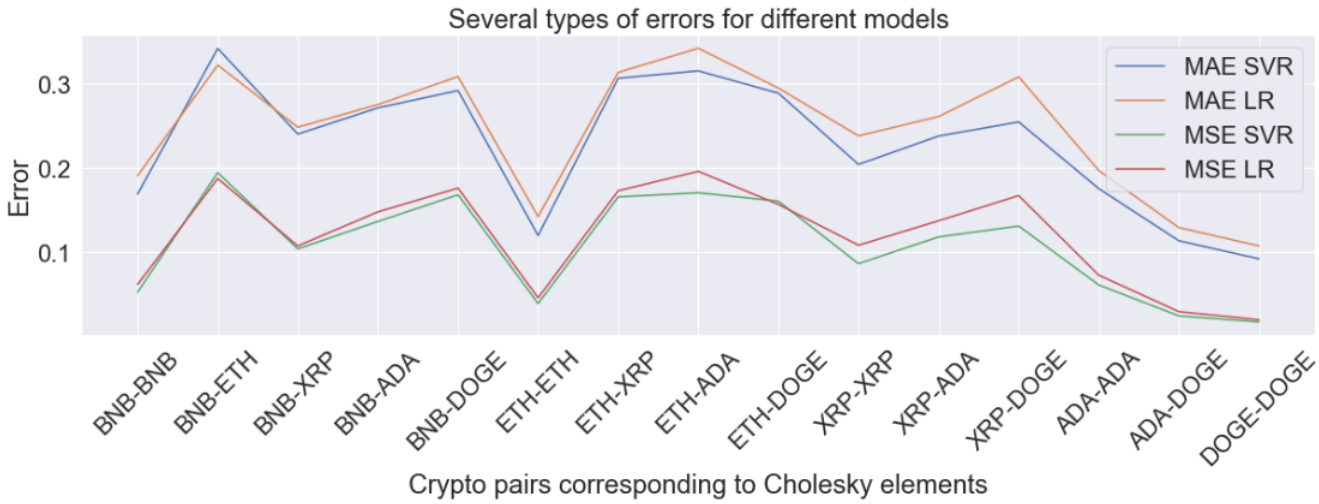


Figure B.18: Plot showing the MSE and MAE of RBF SVR | Lag 15 | 2-day-ahead forecast and LR.

Appendix C

Experiments 3 plots

Linear SVR | Lag 1 | 3-day-ahead forecast

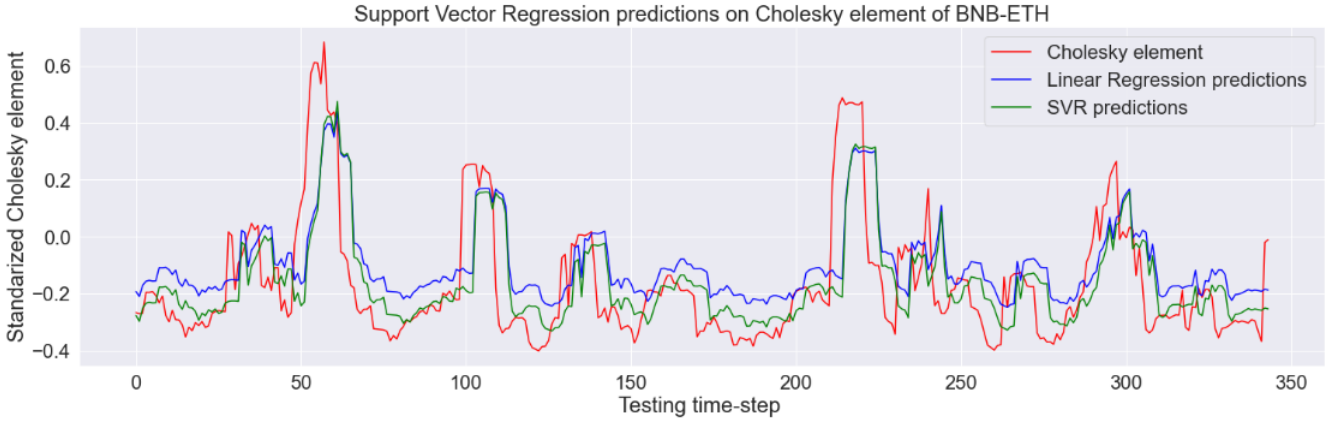


Figure C.1: Plot showing the predicted values of Linear SVR | Lag 1 | 3-day-ahead forecast, LR and the true datapoints.

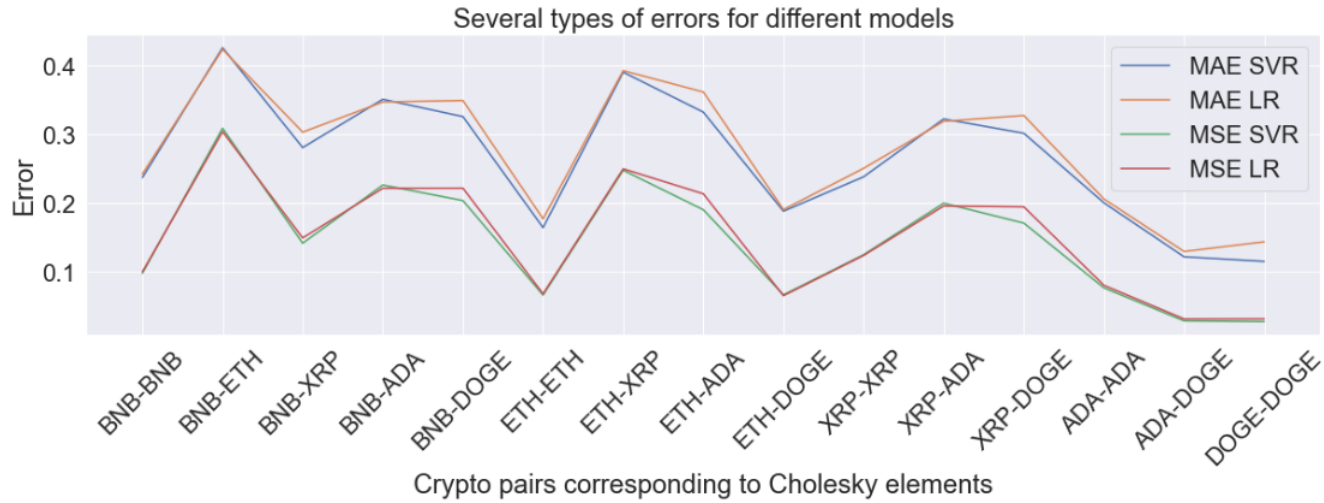


Figure C.2: Plot showing the MSE and MAE of Linear SVR | Lag 1 | 3-day-ahead forecast and LR.

Linear SVR | Lag 5 | 3-day-ahead forecast

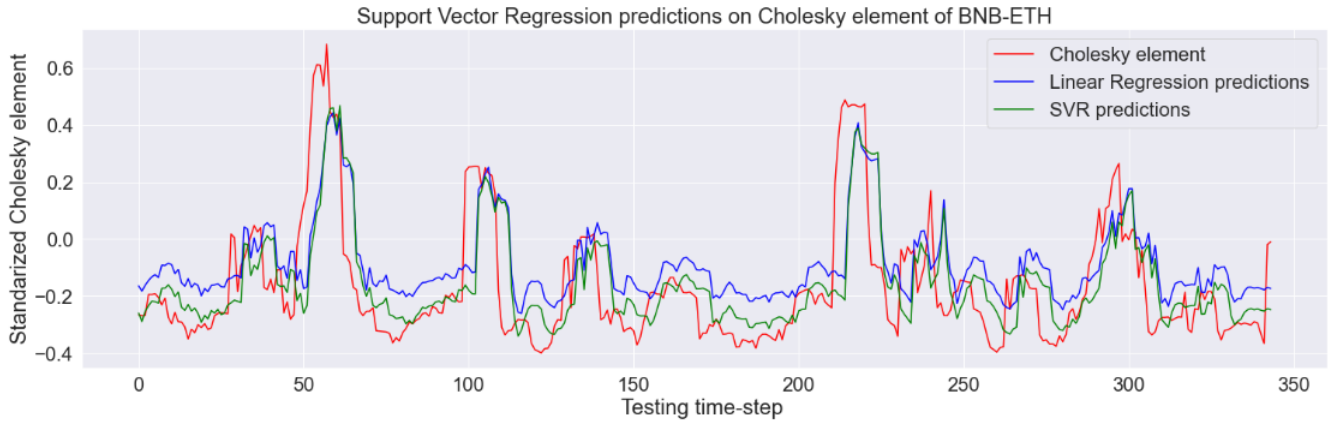


Figure C.3: Plot showing the predicted values of Linear SVR | Lag 5 | 3-day-ahead forecast, LR and the true datapoints.

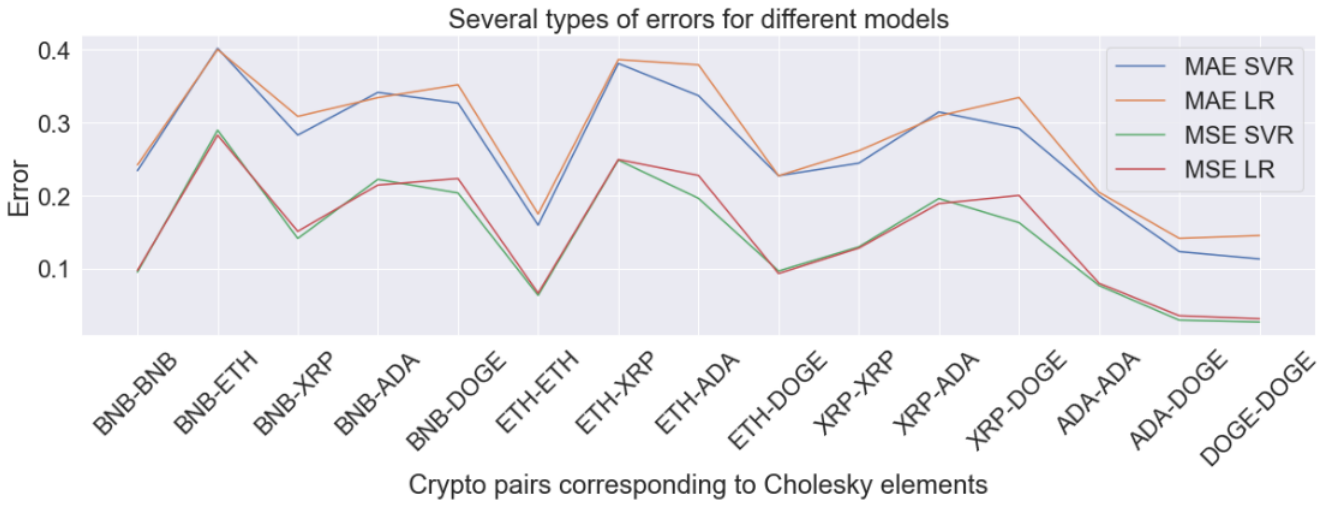


Figure C.4: Plot showing the MSE and MAE of Linear SVR | Lag 5 | 3-day-ahead forecast and LR.

Linear SVR | Lag 15 | 3-day-ahead forecast

Figure C.5: Plot showing the predicted values of Linear SVR | Lag 15 | 3-day-ahead forecast, LR and the true datapoints.

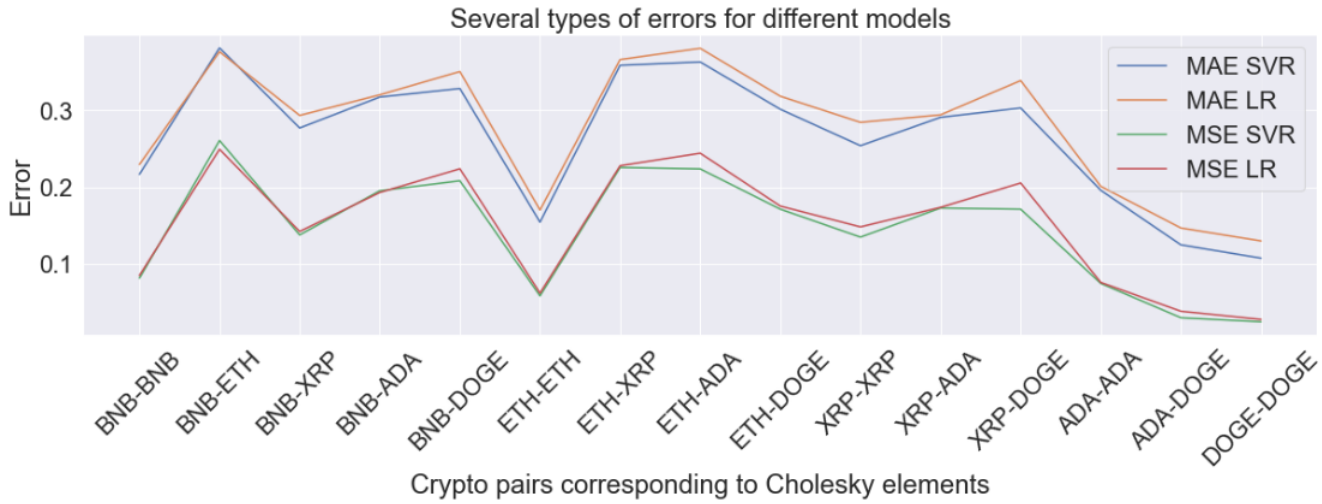


Figure C.6: Plot showing the MSE and MAE of Linear SVR | Lag 15 | 3-day-ahead forecast and LR.

Polynomial (degree 2) SVR | Lag 15 | 3-day-ahead forecast

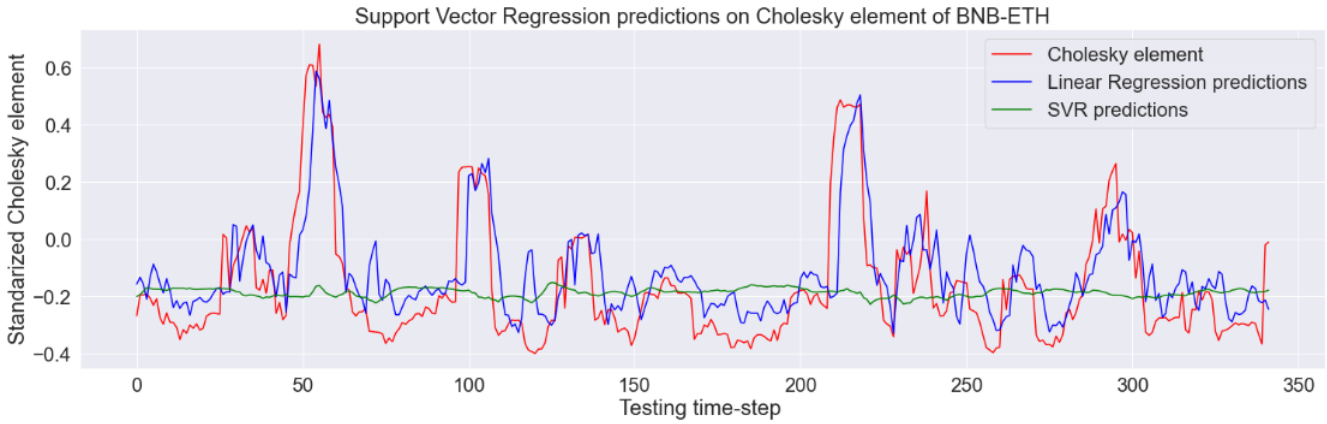


Figure C.7: Plot showing the predicted values of Polynomial (degree 2) SVR | Lag 15 | 3-day-ahead forecast, LR and the true datapoints.

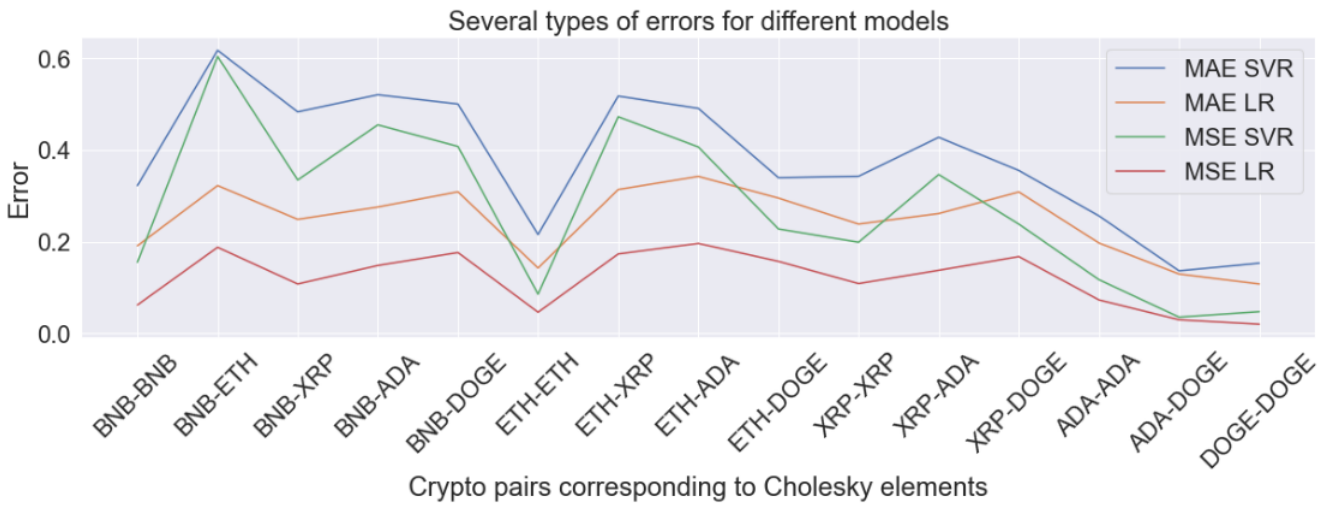


Figure C.8: Plot showing the MSE and MAE of Polynomial (degree 2) SVR | Lag 15 | 3-day-ahead forecast and LR.

Polynomial (degree 3) SVR | Lag 15 | 3-day-ahead forecast

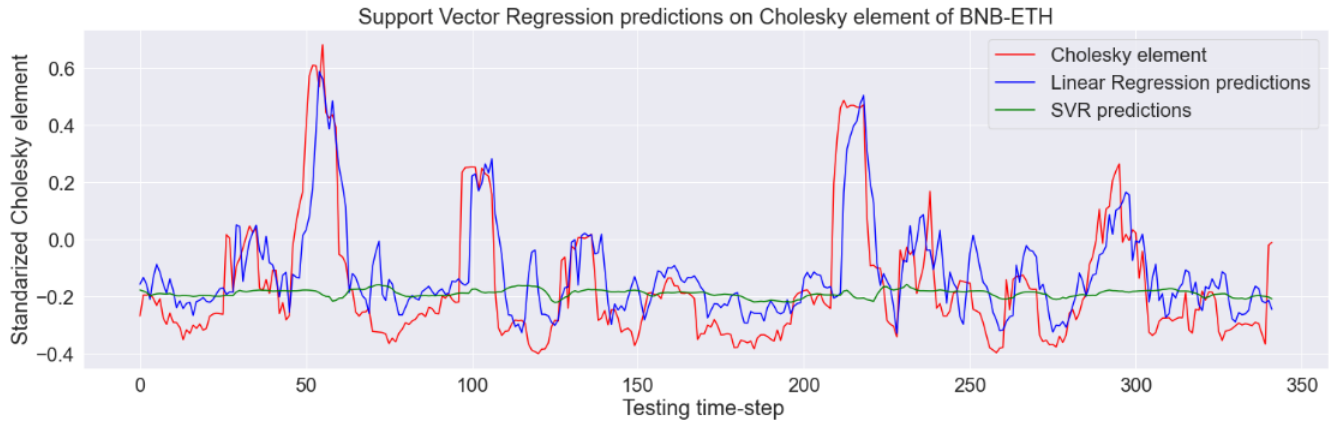


Figure C.9: Plot showing the predicted values of Polynomial (degree 3) SVR | Lag 15 | 3-day-ahead forecast, LR and the true datapoints.

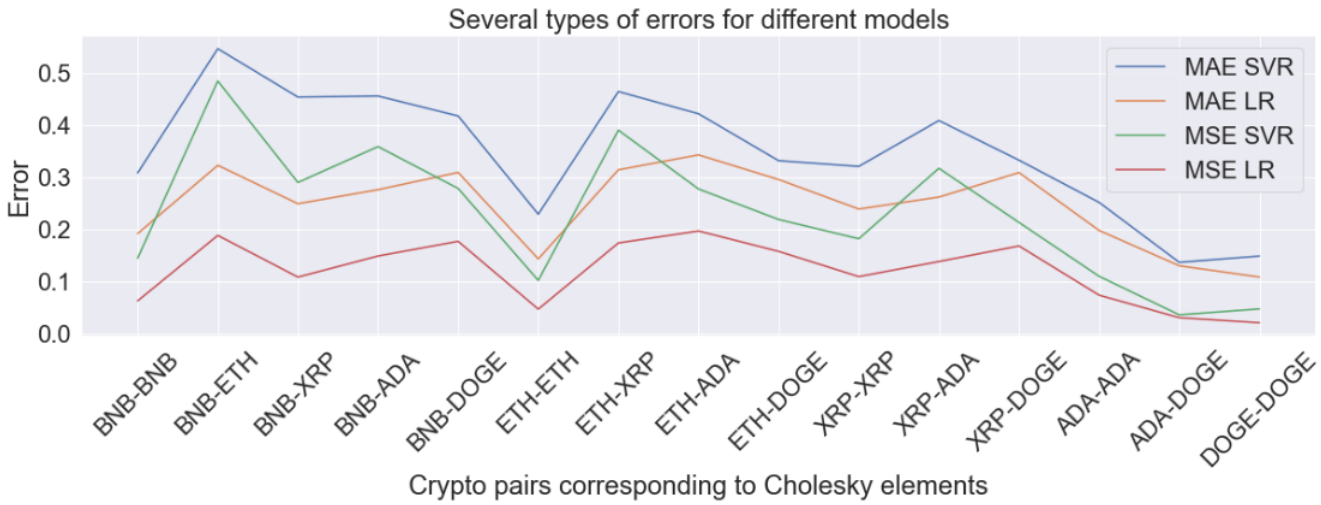


Figure C.10: Plot showing the MSE and MAE of Polynomial (degree 3) SVR | Lag 15 | 3-day-ahead forecast and LR.

Polynomial (degree 4) SVR | Lag 15 | 3-day-ahead forecast

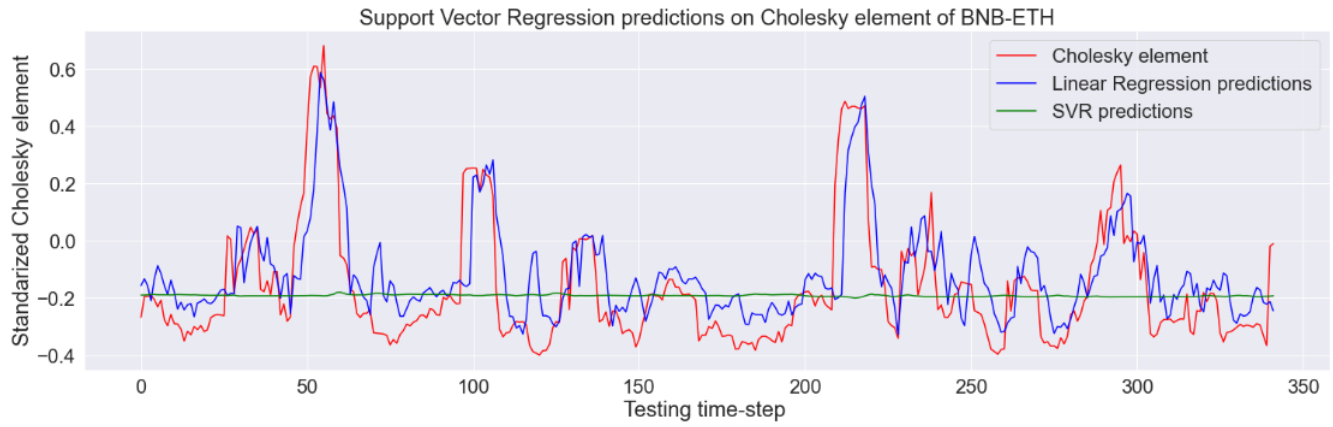


Figure C.11: Plot showing the predicted values of Polynomial (degree 4) SVR | Lag 15 | 3-day-ahead forecast, LR and the true datapoints.

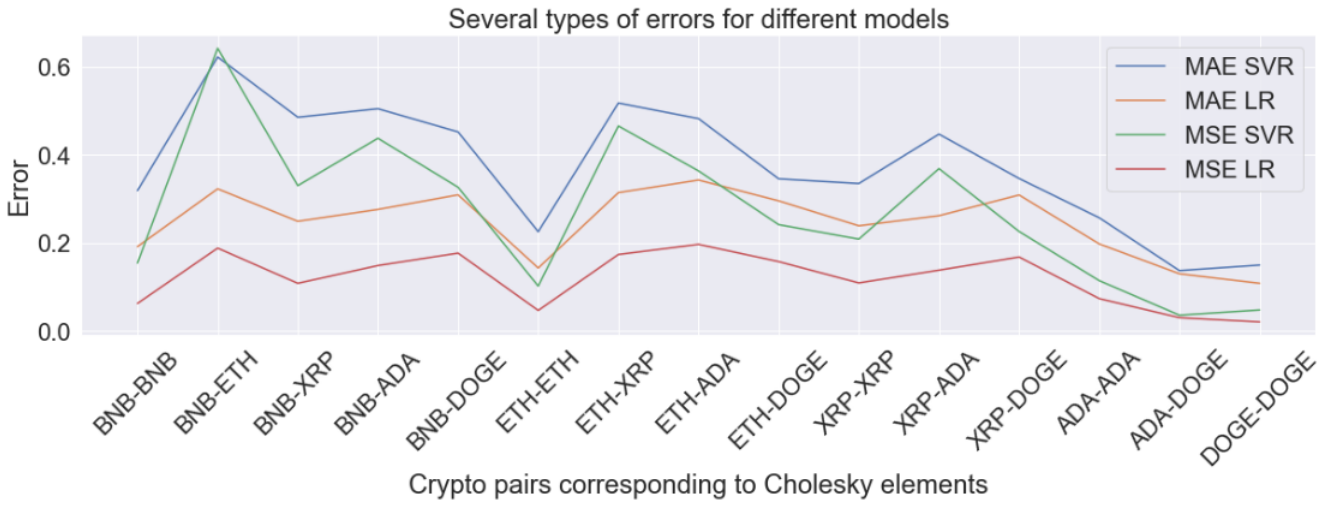


Figure C.12: Plot showing the MSE and MAE of Polynomial (degree 4) SVR | Lag 15 | 3-day-ahead forecast and LR.

RBF SVR | Lag 1 | 3-day-ahead forecast

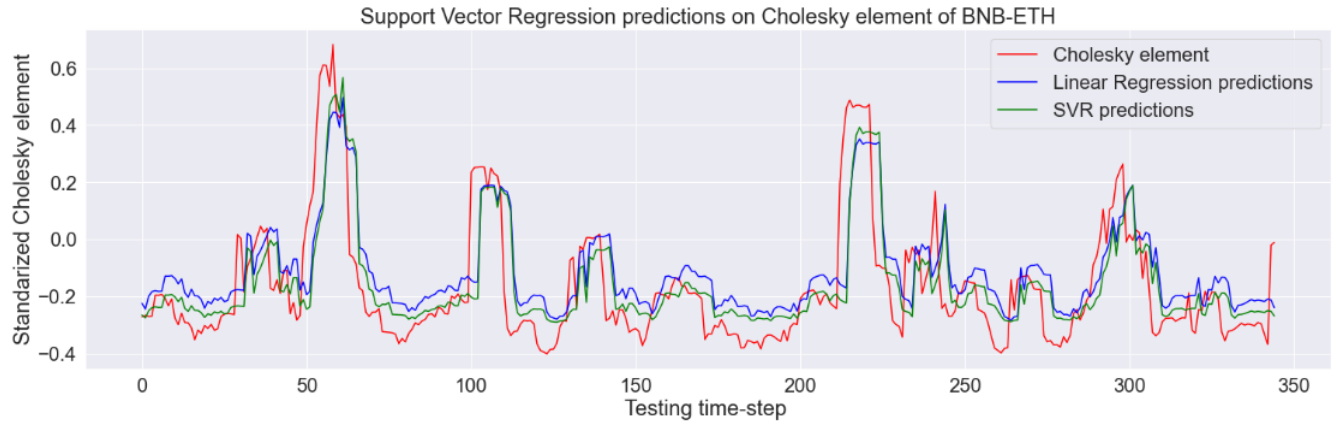


Figure C.13: Plot showing the predicted values of RBF SVR | Lag 1 | 3-day-ahead forecast, LR and the true datapoints.

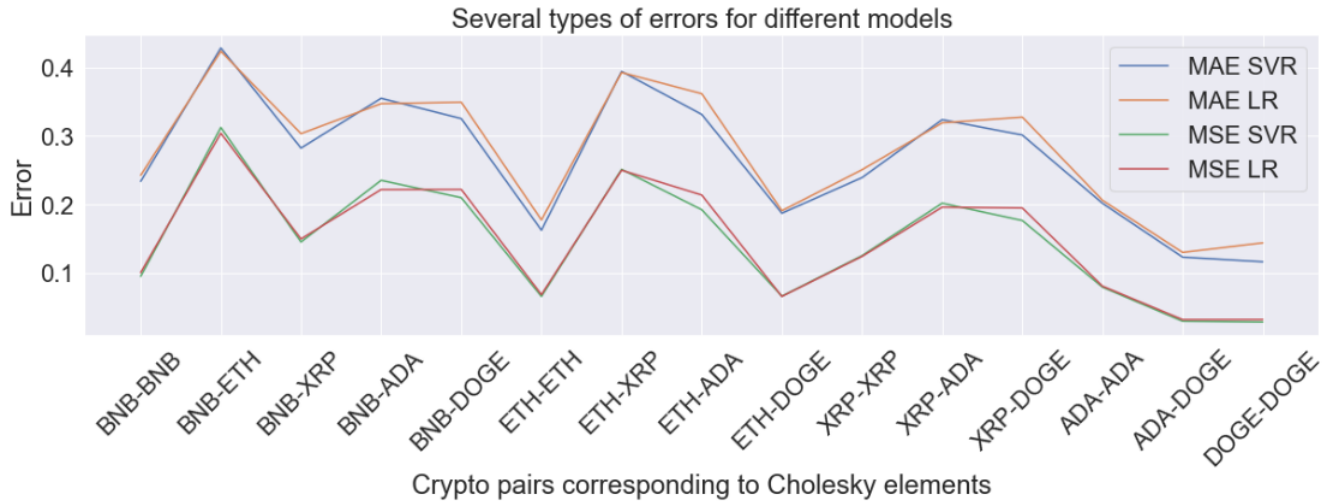


Figure C.14: Plot showing the MSE and MAE of RBF SVR | Lag 1 | 3-day-ahead forecast and LR.

RBF SVR | Lag 5 | 3-day-ahead forecast

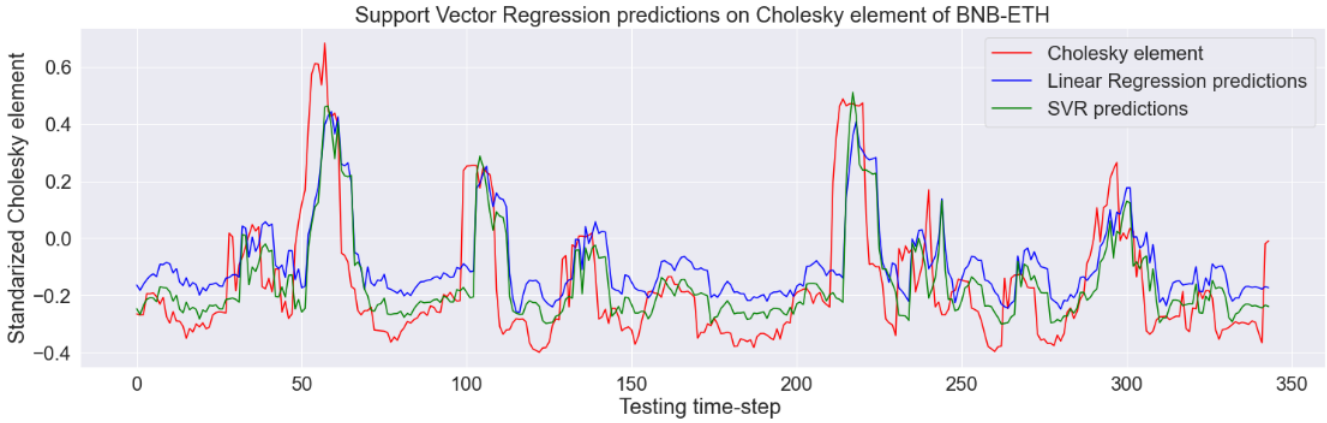


Figure C.15: Plot showing the predicted values of RBF SVR | Lag 5 | 3-day-ahead forecast, LR and the true datapoints.

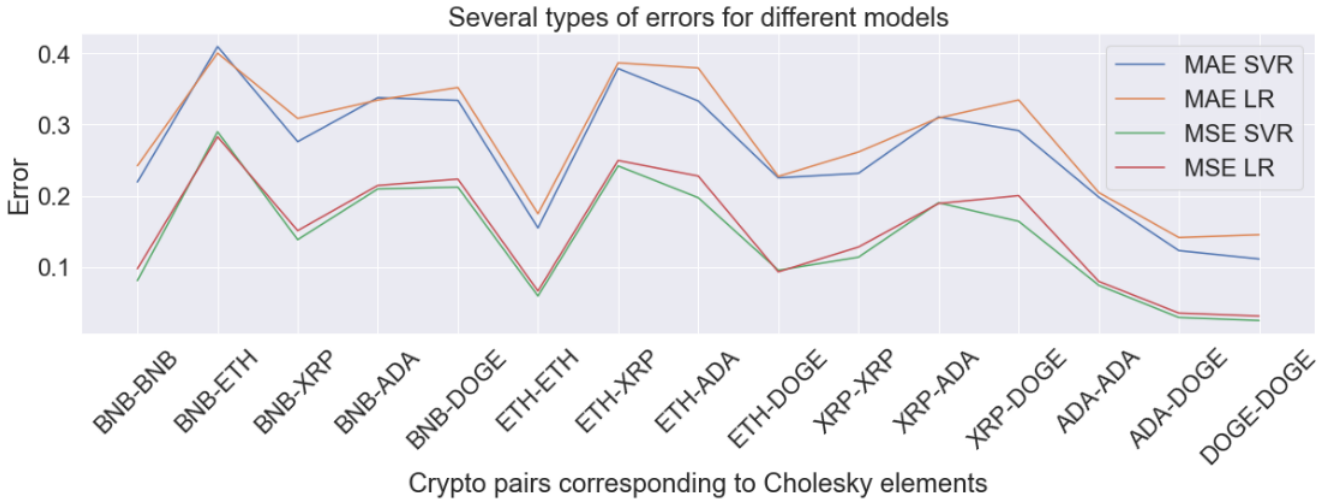


Figure C.16: Plot showing the MSE and MAE of RBF SVR | Lag 5 | 3-day-ahead forecast and LR.

RBF SVR | Lag 15 | 3-day-ahead forecast

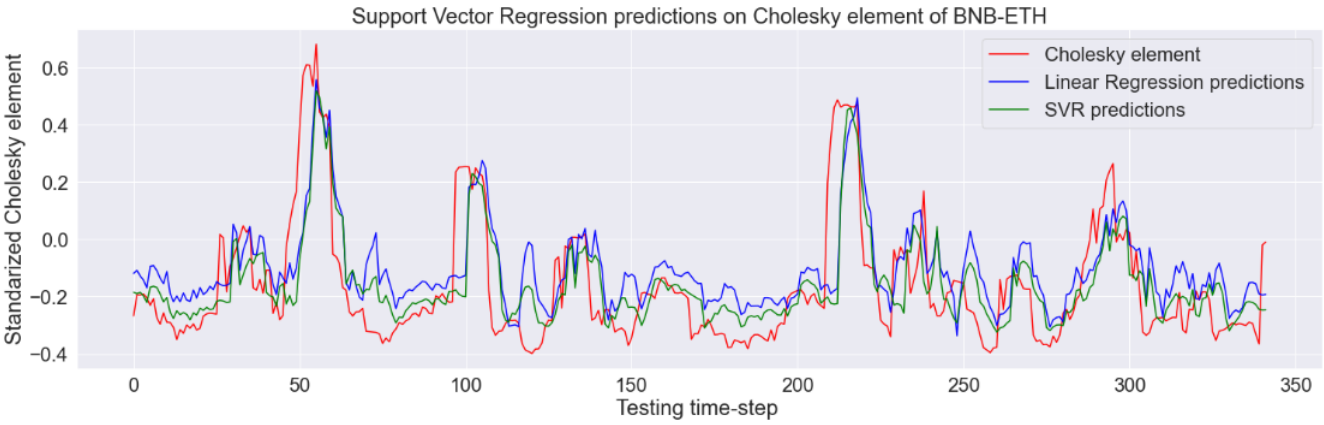


Figure C.17: Plot showing the predicted values of RBF SVR | Lag 15 | 3-day-ahead forecast, LR and the true datapoints.

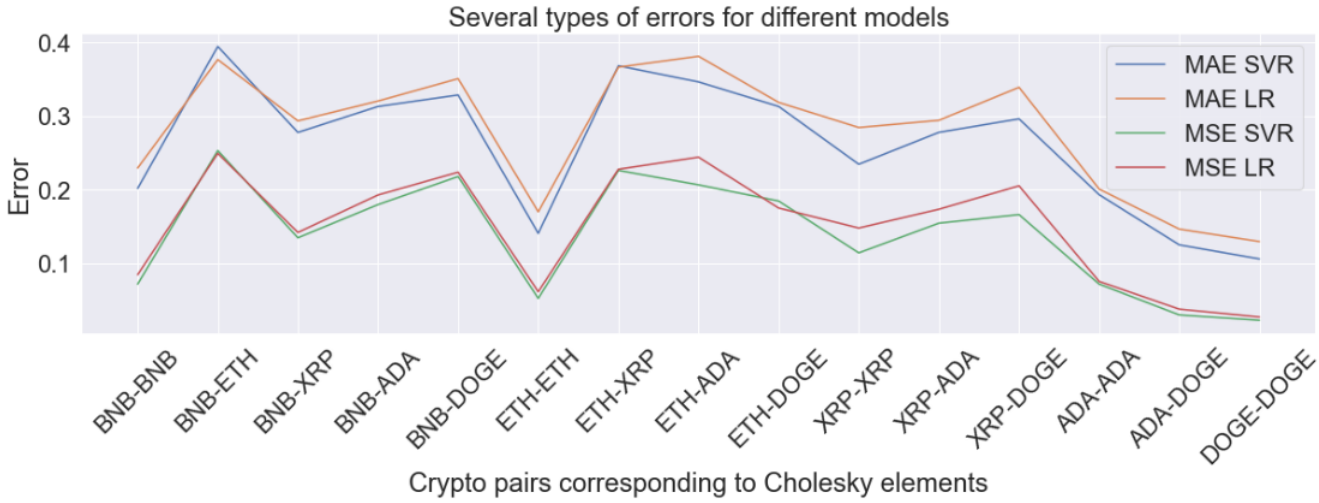


Figure C.18: Plot showing the MSE and MAE of RBF SVR | Lag 15 | 3-day-ahead forecast and LR.

Bibliography

[1] Yfinance python library. https://aroussi.com/post/python-yahoo-finance. Accessed: 2022-08-02.

[2] Anaconda software distribution, 2020.

[3] Shamim Ahmed, Ziwen Bu, and Daniel Tsvetanov. Best of the best: A comparison of factor models. *Journal of Financial and Quantitative Analysis*, 54(4):1713–1758, 2019.

[4] Angelo Aspris, Sean Foley, Jiri Svec, and Leqi Wang. Decentralized exchanges: The "wild west" of cryptocurrency trading. *International Review of Financial Analysis*, 77:101845, 2021.

[5] Jushan Bai and Shuzhong Shi. Estimating high dimensional covariance matrices and its applications. 2011.

[6] Richard T Baillie, G Geoffrey Booth, Yiuman Tse, and Tatyana Zabotina. Price discovery and common factor models. *Journal of financial markets*, 5(3):309–321, 2002.

[7] Söhnke M Bartram and Yaw-Huei Wang. Another look at the relationship between cross-market correlation and volatility. *Finance Research Letters*, 2(2):75–88, 2005.

[8] Illya Barziy, Marcin Chlebus, et al. *HRP performance comparison in portfolio optimization under various codependence and distance metrics*. University of Warsaw, Faculty of Economic Sciences, 2020.

[9] Sylwester Bejger and Piotr Fiszeder. Forecasting currency covariances using machine learning tree-based algorithms with low and high prices. *Przeglad Statystyczny*, 68(3):1–15, 2021.

[10] Guorui Bian, Michael McAleer, and Wing-Keung Wong. Robust estimation and forecasting of the capital asset pricing model. *Annals of Financial Economics*, 8(02):1350007, 2013.

[11] Kris Boudt, Jon Danielsson, and Sébastien Laurent. Robust forecasting of dynamic conditional correlation garch models. *International Journal of Forecasting*, 29(2):244–257, 2013.

[12] Joël Bun, Jean-Philippe Bouchaud, and Marc Potters. Cleaning large correlation matrices: tools from random matrix theory. *Physics Reports*, 666:1–109, 2017.

[13] Rezzy Eko Caraka, Rung Ching Chen, Toni Toharudin, Muhammad Tahmid, Bens Pardamean, and Richard Mahendra Putra. Evaluation performance of svr genetic algorithm and hybrid pso in rainfall forecasting. *ICIC Express Lett Part B Appl*, 11(7):631–639, 2020.

[14] Louis KC Chan, Jason Karceski, and Josef Lakonishok. On portfolio optimization: Forecasting covariances and choosing the risk model. *The review of Financial studies*, 12(5):937–974, 1999.

[15] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[16] Ruoling Chen, H Tunstall-Pedoe, and R Tavendale. Environmental tobacco smoke and lung function in employees who never smoked: the scottish monica study. *Occupational and Environmental Medicine*, 58(9):563–568, 2001.

[17] Shiyi Chen, Wolfgang K Härdle, and Kiho Jeong. Forecasting volatility with support vector machine-based garch model. *Journal of Forecasting*, 29(4):406–433, 2010.

[18] Yin-Wong Cheung and Kon S Lai. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics*, 13(3):277–280, 1995.

[19] Hyeong Kyu Choi. Stock price correlation coefficient prediction with arima-lstm hybrid model. *arXiv preprint arXiv:1808.01560*, 2018.

[20] Wai-Chung Chung, Chuan-Yu Chang, and Chien-Chuan Ko. A svm-based committee machine for prediction of hong kong horse racing. In *2017 10th International*

Conference on Ubi-media Computing and Workshops (Ubi-Media), pages 1–4. IEEE, 2017.

[21] Christophe Croux and Catherine Dehon. Influence functions of the spearman and kendall correlation measures. *Statistical methods & applications*, 19(4):497–515, 2010.

[22] Marcos M López de Prado. *Machine learning for asset managers*. Cambridge University Press, 2020.

[23] Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *The review of Financial studies*, 22(5):1915–1953, 2009.

[24] Ran Duchin and Haim Levy. Markowitz versus the talmudic portfolio diversification strategies. *The Journal of Portfolio Management*, 35(2):71–74, 2009.

[25] J. Durbin. The fitting of time-series models. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 28(3):233–244, 1960.

[26] James Durbin. The fitting of time-series models. *Revue de l'Institut International de Statistique*, pages 233–244, 1960.

[27] Marcin Fałdziński, Piotr Fiszeder, and Witold Orzeszko. Forecasting volatility of energy commodities: Comparison of garch models with support vector regression. *Energies*, 14(1):6, 2020.

[28] Eugene F Fama and Kenneth R French. The cross-section of expected stock returns. *the Journal of Finance*, 47(2):427–465, 1992.

[29] Eugene F Fama and Kenneth R French. The capital asset pricing model: Theory and evidence. *Journal of economic perspectives*, 18(3):25–46, 2004.

[30] Jianqing Fan, Yingying Fan, and Jinchi Lv. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, 147(1):186–197, 2008. Econometric modelling in finance and risk management: An overview.

[31] Yanwen Fang, Philip LH Yu, and Yaohua Tang. Cnn-based realized covariance matrix forecasting. *arXiv preprint arXiv:2107.10602*, 2021.

[32] Piotr Fiszeder and Witold Orzeszko. Covariance matrix forecasting using support vector regression. *Applied intelligence*, 51(10):7029–7042, 2021.

[33] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[34] Firas Gerges, Germain Zouein, and Danielle Azar. Genetic algorithms with local optima handling to solve sudoku puzzles. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, ICCAI 2018, page 19–22, New York, NY, USA, 2018. Association for Computing Machinery.

[35] Ali Ghaheri, Saeed Shoar, Mohammad Naderan, and Sayed Shahabuddin Hoseini. The applications of genetic algorithms in medicine. *Oman medical journal*, 30(6):406, 2015.

[36] Arjan Gijsberts, Giorgio Metta, and Léon Rothkrantz. Evolutionary optimization of least-squares support vector machines. In *Data mining*, pages 277–297. Springer, 2010.

[37] Arthur Stanley Goldberger et al. Econometric theory. *Econometric theory.*, 1964.

[38] Arthur Stanley Goldberger and Arthur Stanley Goldberger Goldberger. *A course in econometrics*. Harvard University Press, 1991.

[39] CWJ Granger. Combining forecasts—twenty years later. *Essays in Econometrics: Collected Papers of Clive WJ Granger*, 32:411, 2001.

[40] Pankaj Gupta, Mukesh Kumar Mehlawat, and Garima Mittal. Asset portfolio optimization using support vector machines and real-coded genetic algorithm. *Journal of Global Optimization*, 53(2):297–315, 2012.

[41] James Douglas Hamilton. *Time series analysis*. Princeton university press, 2020.

[42] Seng Hansun. A new approach of moving average method in time series analysis. In *2013 conference on new media studies (CoNMedia)*, pages 1–4. IEEE, 2013.

[43] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[44] Akinori Hirabayashi, Claus Aranha, and Hitoshi Iba. Optimization of the trading rule in foreign exchange using genetic algorithm. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1529–1536, 2009.

[45] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, 36(3):1171–1220, 2008.

[46] Petro Hrytsiuk, Tetiana Babych, and Larysa Bachyshyna. Cryptocurrency portfolio optimization using value-at-risk measure. In *6th International Conference on Strategies, Models and Technologies of Economic Systems Management (SMTESM 2019)*, pages 385–389. Atlantis Press, 2019.

[47] Shujun Huang, Nianguang Cai, Pedro Penzuti Pacheco, Shavira Narrandes, Yang Wang, and Wayne Xu. Applications of support vector machine (svm) learning in cancer genomics. *Cancer genomics & proteomics*, 15(1):41–51, 2018.

[48] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[49] S Kazarlis and Vassilios Petridis. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *International conference on parallel problem solving from nature*, pages 211–220. Springer, 1998.

[50] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[51] Serhiy Kozak, Stefan Nagel, and Shrihari Santosh. Interpreting factor models. *The Journal of Finance*, 73(3):1183–1223, 2018.

[52] Denis Kwiatkowski, Peter CB Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[53] Olivier Ledoit and Michael Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of empirical finance*, 10(5):603–621, 2003.

[54] Chien-Chiang Lee, Jun-De Lee, and Chi-Chuan Lee. Stock prices and the efficient market hypothesis: Evidence from a panel stationary test with structural breaks. *Japan and the world economy*, 22(1):49–58, 2010.

[55] Yuze Li, Shangrong Jiang, Yunjie Wei, and Shouyang Wang. Take bitcoin into your portfolio: a novel ensemble portfolio optimization framework for broad commodity assets. *Financial Innovation*, 7(1):1–26, 2021.

[56] Kian-Ping Lim, Robert D Brooks, and Melvin J Hinich. Nonlinear serial dependence and the weak-form efficiency of asian emerging stock markets. *Journal of International Financial Markets, Institutions and Money*, 18(5):527–544, 2008.

[57] Aranildo Rodrigues Lima Junior, David Augusto Silva, Paulo Salgado Mattos Neto, and Tiago AE Ferreira. An experimental study of fitness function and time series forecasting using artificial neural networks. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*, pages 2015–2018, 2010.

[58] Guiqian Liu, Xiangdong Gao, Deyong You, and Nanfeng Zhang. Prediction of high power laser welding status based on pca and svm classification of multiple sensors. *Journal of Intelligent Manufacturing*, 30(2):821–832, 2019.

[59] Ilya Loshchilov and Frank Hutter. Cma-es for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

[60] Spyros Makridakis and Michele Hibon. Arma models and the box–jenkins methodology. *Journal of forecasting*, 16(3):147–163, 1997.

[61] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[62] Mariano Matilla-García. Nonlinear dynamics in energy futures. *The Energy Journal*, 28(3), 2007.

[63] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[64] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[65] Gabor Papp, Szilard Pafka, Maciej A Nowak, and Imre Kondor. Random matrix filtering in portfolio optimization. *arXiv preprint physics/0509235*, 2005.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] Yaohao Peng, Pedro Henrique Melo Albuquerque, Igor Ferreira do Nascimento, and João Victor Freitas Machado. Between nonlinearities, complexity, and noises: An application on portfolio selection using kernel principal component analysis. *Entropy*, 21(4), 2019.

[68] Fernando Pérez-Cruz, Julio A Afonso-Rodriguez, and Javier Giner. Estimating garch models using support vector machines. *Quantitative Finance*, 3(3):163, 2003.

[69] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, Brown Univ Providence Ri Inst for Brain and Neural Systems, 1992.

[70] Momtchil Pojarliev and Wolfgang Polasek. Applying multivariate time series forecasts for active portfolio management. *Financial Markets and Portfolio Management*, 15(2):201, 2001.

[71] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[72] Fred L Ramsey. Characterization of the partial autocorrelation function. *The Annals of Statistics*, pages 1296–1301, 1974.

[73] Fei Ren, Ya-Nan Lu, Sai-Ping Li, Xiong-Fei Jiang, Li-Xin Zhong, and Tian Qiu. Dynamic portfolio strategy using clustering approach. *PloS one*, 12(1):e0169299, 2017.

[74] David N Reshef, Yakir A Reshef, Hilary K Finucane, Sharon R Grossman, Gilean McVean, Peter J Turnbaugh, Eric S Lander, Michael Mitzenmacher, and Pardis C Sabeti. Detecting novel associations in large data sets. *science*, 334(6062):1518–1524, 2011.

[75] Stephen A Ross. The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3):341–360, 1976.

[76] Guillermo Santamaría-Bonfil, Juan Frausto-Solís, and Ignacio Vázquez-Rodarte. Volatility forecasting using support vector regression and a hybrid genetic algorithm. *Computational Economics*, 45(1):111–133, 2015.

[77] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36. IEEE, 2004.

[78] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[79] George AF Seber and Alan J Lee. *Linear regression analysis*. John Wiley & Sons, 2012.

[80] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[81] William F Sharpe. A simplified model for portfolio analysis. *Management science*, 9(2):277–293, 1963.

[82] Robert H Shumway, David S Stoffer, and David S Stoffer. *Time series analysis and its applications*, volume 3. Springer, 2000.

[83] Noah Simon and Robert Tibshirani. Comment on" detecting novel associations in large data sets" by reshef et al, science dec 16, 2011. *arXiv preprint arXiv:1401.7645*, 2014.

[84] Dinesh Sivagourou. Portfolio optimization with cryptocurrencies. Master's thesis, Humboldt-Universität zu Berlin, 2018.

[85] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[86] Terry Speed. A correlation for the 21st century. *Science*, 334(6062):1502–1503, 2011.

[87] Ruili Sun, Tiefeng Ma, and Shuangzhe Liu. Portfolio selection based on semivariance and distance correlation under minimum variance framework. *Statistica Neerlandica*, 73(3):373–394, 2019.

[88] Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794, 2007.

[89] John S Tyssedal and Dag Tjøstheim. An autoregressive model with suddenly changing parameters and an application to stock market prices. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 37(3):353–369, 1988.

[90] Roy Van der Weide. Go-garch: a multivariate generalized orthogonal garch model. *Journal of Applied Econometrics*, 17(5):549–564, 2002.

[91] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[92] Vladimir N Vapnik. The nature of statistical learning. *Theory*, 1995.

[93] Liang-Ying Wei, Ching-Hsue Cheng, and Hsin-Hung Wu. A hybrid anfis based on n-period moving average model to forecast taiex stock. *Applied Soft Computing*, 19:86–92, 2014.

[94] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[95] Peter Whittle. *Hypothesis testing in time series analysis*, volume 4. Almqvist & Wiksells boktr., 1951.

[96] Björn Wolff. *Support vector regression for solar power prediction*. PhD thesis, Universität Oldenburg, 2017.

[97] Chih-Hung Wu, Gwo-Hshiung Tzeng, Yeong-Jia Goo, and Wen-Chang Fang. A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. *Expert systems with applications*, 32(2):397–408, 2007.

[98] Ni Xin, Xiaofeng Gu, Hao Wu, Yuzhu Hu, and Zhonglin Yang. Application of genetic algorithm-support vector regression (ga-svr) for quantitative analysis of herbal medicines. *Journal of Chemometrics*, 26(7):353–360, 2012.

[99] Fong-Ching Yuan and Chao-Hui Lee. Using least square support vector regression with genetic algorithm to forecast beta systematic risk. *Journal of Computational Science*, 11:26–33, 2015.

[100] George Udny Yule. Vii. on a method of investigating periodicities disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226(636-646):267–298, 1927.

[101] Valeriy Zakamulin. A test of covariance-matrix forecasting methods. *The Journal of Portfolio Management*, 41(3):97–108, 2015.

[102] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[103] Ming Zhu and Lipo Wang. Intelligent trading using support vector regression and multilayer perceptrons optimized with genetic algorithms. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5. IEEE, 2010.