Individual Diploma Thesis

Predictive Modeling for the Digital Asset Markets: Feature Selection and Comparative Analysis of Machine Learning Algorithms

Eliada Polydorou

Department of Computer Science

University of Cyprus

May 2024

UNIVERSITY OF CYPRUS

Faculty of Pure and Applied Sciences

Department of Computer Science

Predictive Modeling for the Digital Asset Markets: Feature Selection and Comparative Analysis of Machine Learning Algorithms

Eliada Polydorou

Supervisor

Dr Chryssis Georgiou

The Thesis was submitted in partial fulfillment of the requirements for obtaining the Computer Science degree of the Department of Computer Science of the University of Cyprus

May 2024

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Georgiou Chryssis, Professor at the Department of Computer Science at the University of Cyprus, for selecting me for this topic and helping me at every stage of my thesis.

Furthermore, I wish to thank Giorgos Demosthenous for his continuous guidance and feedback throughout all stages of my thesis.

Finally, I am very thankful to my parents and friends for their support and encouragement throughout this journey.

ABSTRACT

Cryptocurrencies have become a hot topic in the financial world, especially with the rise of Bitcoin and other digital assets. Unlike traditional currencies, cryptocurrencies operate independently of any central authority, making their prices highly volatile and challenging to predict accurately. This make significant challenges for investors and traders who rely on precise forecasts to make informed decisions and manage risks effectively.

In this thesis, we aim to identify the indicators that most significantly affect our target, the Crypto100 index, and explore the strengths and weaknesses of various machine learning algorithms in predicting it. Specifically, we analyze seven distinct regression model algorithms: Linear Regression, Support Vector Regression, K-Nearest Neighbors, Decision Tree, AdaBoost, XGBoost, and Random Forest. Additionally, we investigate the performance of three deep learning algorithms: Long Short-Term Memory, Convolutional Neural Networks, and Recurrent Neural Networks. We focus on predicting the price movements of the Crypto100 index for different window sizes for future price predictions. Our goal was to assess the models' capabilities for short-term, medium-term, and long-term predictions and to see how different features affect our target variable across these prediction windows. We evaluate these models based on their accuracy in predicting price changes over these different time frames, using metrics such as RMSE, MAE, MedAE, MAPE, and R-squared.

The study reveals that while traditional models and ensemble methods can provide some insights, they are not the best for predicting the Crypto100 index due to the dynamic nature of the cryptocurrency market. On the other hand, neural networks proved to be more accurate. More specifically, the LSTM model achieved high accuracy in predicting the price for the next day and the next seven days. These findings suggest that neural networks are more effective at capturing the complex relationships and patterns in cryptocurrency price data.

TABLE OF CONTENTS

ABSTRACT iii

TABLE OF CONTENTS iv

LIST OF TABLES vi

LIST OF FIGURES vii

LIST OF ABBREVIATIONS ix

1 Introduction 1

1.1 Motivation . . . 1
1.2 Goals of the Study . . . 2
1.3 Methodology . . . 2
1.4 Document Organization . . . 4

2 Background Knowledge 5

2.1 Introduction to Cryptocurrencies . . . 6
2.1.1 Understanding Cryptocurrencies: Key Questions . . . 6
2.1.2 Crypto100 Index Price . . . 7
2.2 Machine learning Algorithms . . . 7
2.2.1 Linear Regression . . . 8
2.2.2 K-Nearest Neighbors . . . 8
2.2.3 Decision Tree . . . 9
2.2.4 Random Forest . . . 10
2.2.5 AdaBoost . . . 10
2.2.6 XGBoost . . . 11
2.2.7 Support Vector Regression . . . 11
2.3 Deep Learning Algorithms . . . 12
2.3.1 Recurrent Neutal Network . . . 13
2.3.2 Long Short-term Memory . . . 13

2.3.3 Convolutional Neural Netowrk . . . 14
2.4 Feature Importance Algorithms . . . 16
2.4.1 Mean Decrease Impurity (MDI) . . . 16
2.4.2 Permutation Feature Importance . . . 16
2.4.3 SHAP Algorithm . . . 16
2.5 Evaluation Metrics . . . 18
2.5.1 Root Mean Squared Error . . . 18
2.5.2 Mean Absolute Error . . . 18
2.5.3 Mean Absolute Percentage Error . . . 19
2.5.4 Median Absolute Error . . . 19
2.5.5 R-squared . . . 20

3 Data Collection and Analysis **21**
3.1 Data Collection . . . 21
3.2 Data Preprocessing . . . 22
3.3 Data Analysis . . . 24
3.3.1 Period Selection . . . 24
3.3.2 Methodology for Feature Selection . . . 25

4 Results and Analysis **29**
4.1 Machine Learning Algorithms . . . 29
4.1.1 Methodology . . . 29
4.1.2 Results . . . 30
4.2 Deep Learning Algorithms . . . 53
4.2.1 Methodology . . . 53
4.2.2 Results . . . 53
4.3 Model Comparison . . . 63

5 Discussion **66**
5.1 Summary and Final Words . . . 66
5.2 Future work . . . 67

BIBLIOGRAPHY **68**

LIST OF TABLES

1.1 Chapter Descriptions 4

3.1 Number of rows and features for each dataset 22
3.2 Individual Analysis Start Dates for Each Dataset 26
3.3 Number of features in each dataset 27

4.1 Linear regression metrics 30
4.2 Support Vector Regression Metrics 34
4.3 K-Nearest Neighbors Metrics 37
4.4 Decision Tree Best Parameters 41
4.5 Decision Tree Metrics 41
4.6 AdaBoost Best Parameters 44
4.7 AdaBoost Metrics 44
4.8 XGBoost Best Parameters 47
4.9 XGBoost Metrics 47
4.10 Random Forest Best Parameters 50
4.11 Random Forest Metrics 50
4.12 CNN Model Architecture 53
4.13 CNN Model Metrics 54
4.14 RNN Architecture 57
4.15 RNN Model Performance Metrics 57
4.16 LSTM Model Architecture and Parameters 60
4.17 LSTM Evaluation Metrics 60

LIST OF FIGURES

1.1 Abstract methodology representation of Thesis 3

2.1 Example of a Regression Decision Tree 10
2.2 Example of the architecture of a simple NN 12
2.3 The architecture of LSTM network 14
2.4 The architecture of CNN 15
2.5 Example of SHAP Values of some features for Crypto100 Index prediction . . 17

3.1 Number of Features by Their First Date of Appearance - All Datasets Combined (Normalized Dates) 23
3.2 Number of Columns Grouped by First Non-Empty Entry Date 24
3.3 Market Bottom and Start of New Cycle from 1 January 2019 25
3.4 Comparison between Reduction Algorithm and SHAP Features for Different Window Sizes in 2017 and 2019 Periods 28

4.1 Linear Regression Predictions vs Actual Prices for Different Window Sizes in 2017 32
4.2 Linear Regression Predictions vs Actual Prices for Different Window Sizes in 2019 33
4.3 SVR Predictions vs Actual Prices for Different Window Sizes in 2017 35
4.4 SVR Predictions vs Actual Prices for Different Window Sizes in 2019 36
4.5 KNN Predictions vs Actual Prices for Different Window Sizes in 2017 38
4.6 KNN Predictions vs Actual Prices for Different Window Sizes in 2019 39
4.7 ML Model Weakness 40
4.8 Decision Tree Predictions vs Actual Prices for Different Window Sizes in 2017 42
4.9 Decision Tree Predictions vs Actual Prices for Different Window Sizes in 2019 43
4.10 AdaBoost Predictions vs Actual Prices for Different Window Sizes in 2017 . . 45
4.11 AdaBoost Predictions vs Actual Prices for Different Window Sizes in 2019 . . 46
4.12 XGBoost Predictions vs Actual Prices for Different Window Sizes in 2017 . . . 48
4.13 XGBoost Predictions vs Actual Prices for Different Window Sizes in 2019 . . . 49
4.14 Random Forest Predictions vs Actual Prices for Different Window Sizes in 2017 51

4.15 Random Forest Predictions vs Actual Prices for Different Window Sizes in 2019 52
4.16 CNN Predictions vs Actual Prices for Different Window Sizes in 2017 55
4.17 CNN Predictions vs Actual Prices for Different Window Sizes in 2019 56
4.18 RNN Predictions vs Actual Prices for Different Window Sizes in 2017 58
4.19 RNN Predictions vs Actual Prices for Different Window Sizes in 2019 59
4.20 LSTM Predictions vs Actual Prices for Different Window Sizes in 2017 61
4.21 LSTM Predictions vs Actual Prices for Different Window Sizes in 2019 62
4.22 Comparison of RMSE Across ML Models 63
4.23 Comparison of MedAE Across ML Models 64
4.24 Comparison of RMSE Across DL Models 65
4.25 Comparison of MedAE Across DL Models 65

LIST OF ABBREVIATIONS

ML	Machine Learning
DL	Deep Learning
NN	Neural Network
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MedAE	Median Absolute Error
LSTM	Long Short-Term Memory
ReLu	Rectified Linear Unit
RNN	Recurrent Neural Network
MDI	Mean Decrease Impurity
PFI	Permutation Feature Importance
SHAP	SHapley Additive exPlanations
SVR	Support Vector Regression
KNN	K-Nearest Neighbors
CNN	Convolutional Neural Network

1 Introduction

Contents

1.1 Motivation 1
1.2 Goals of the Study 2
1.3 Methodology 2
1.4 Document Organization 4

1.1 Motivation

Cryptocurrencies have become a hot topic in the financial world, especially with the rise of Bitcoin [1] and other digital assets. Unlike traditional currencies, cryptocurrencies operate independently of any central authority, making their prices highly volatile and challenging to predict accurately [2]. This makes significant challenges for investors and traders who rely on precise forecasts to make informed decisions and manage risks effectively.

To address these challenges, researchers have turned to machine learning (ML) techniques [3] to enhance price prediction accuracy. ML algorithms, with their ability to learn from data and identify patterns, offer a promising approach to understanding cryptocurrency market dynamics. By utilizing ML models, investors can potentially uncover valuable insights into market trends and improve their investment strategies.

However, despite the potential of ML in cryptocurrency price prediction, the effectiveness of these models can vary significantly. While some studies have achieved promising results with good accuracies, others have struggled to provide reliable predictions due to the dynamic nature of the cryptocurrency market.

In this thesis, we aim to explore the effectiveness of various ML algorithms, including traditional regression-based methods and ensemble techniques, alongside neural networks, in predicting the price movements of the Crypto100 index [4] for different time windows into the future. Additionally, we compare the performance of these algorithms to determine their relative strengths and weaknesses.

1.2 Goals of the Study

Our primary objective was to gather a diverse range of data from various indicators, specifically focusing on Technical, Fundamental, Market Psychology, and On-chain Information. This extensive dataset served as the foundation for our subsequent analysis, aiming to identify the most effective indicators that would yield the best predictions. By considering a wide range of indicators, we aimed to select those that could effectively identify patterns crucial for accurate predictions.

Following data collection, our attention turned to data processing and feature selection, which is crucial for the models to perform well. Given the substantial amount of data available, our next step involved processing the collected data and utilizing feature selection algorithms to reduce the number of features. Our goal was to keep only the most relevant indicators that significantly contributed to the model's accuracy. This led to the creation of datasets that resulted in more precise predictions.

Our final goal was to apply various machine learning algorithms to the processed datasets and then compare and analyze their performance using predetermined evaluation metrics. This approach allowed us to assess the strengths and weaknesses of various ML algorithms in predicting cryptocurrency prices.

1.3 Methodology

Figure 1.1 constitutes an abstract representation of the overall methodology followed throughout this thesis.

Our study began with the collection of data, where we utilized various sources and APIs to obtain indicators from technical, fundamental, market psychology, and on-chain information.

In the data analysis and preprocessing phase, we cleaned and preprocessed each sub-dataset to ensure consistency. We converted all data to a daily format, addressed missing values, and removed irrelevant features or duplicates. We then analyzed both individual sub-datasets and explored combined analyses to identify potential relationships between features. Importantly, we defined the start and end dates for each dataset and designated the Crypto100 index as our target variable for prediction. Additionally, we established various window sizes for future price predictions. Our aim was to assess the models' capabilities for short, medium, and long-term predictions and to see how different features affecting our target variable change across these different prediction windows. By the end of this stage, we had prepared clean and structured datasets suitable for training and testing ML models.

Next, we employed feature importance algorithms such as permutation feature importance and the Shapley Additive Explanations (SHAP) algorithm [5]. These algorithms helped identify the most influential features for predictions. A detailed breakdown of this analysis is provided in Section 3.3 for further reference.

Following these steps, the datasets were prepared for model fine-tuning. Seven distinct regression model algorithms, including Linear Regression [6], AdaBoost [7], Decision Tree [8], K-Nearest Neighbors [9], Random Forest [10], Support Vector Regressions [11], and XGBoost [12], as well as three Deep Learning Algorithms, including LSTM [13], CNN [14], and RNN [15] were explored and analyzed in Section 2.2. Each model possesses parameters that can be adjusted (tuned) to improve its performance on a specific dataset. Therefore, we conducted a fine-tuning process, experimenting with different parameter values for each model and selecting the combination that yielded the best results.

Finally, after fine-tuning the models, we divided the datasets into training and testing sets. The models were then trained using the optimized hyperparameters, and predictions were generated. Our final step involved comparing and analyzing the performance of each model based on pre-selected evaluation metrics. This approach allowed us to assess the strengths and weaknesses of various ML and DL algorithms in predicting cryptocurrency prices. The findings from this evaluation are presented and discussed in subsequent sections.

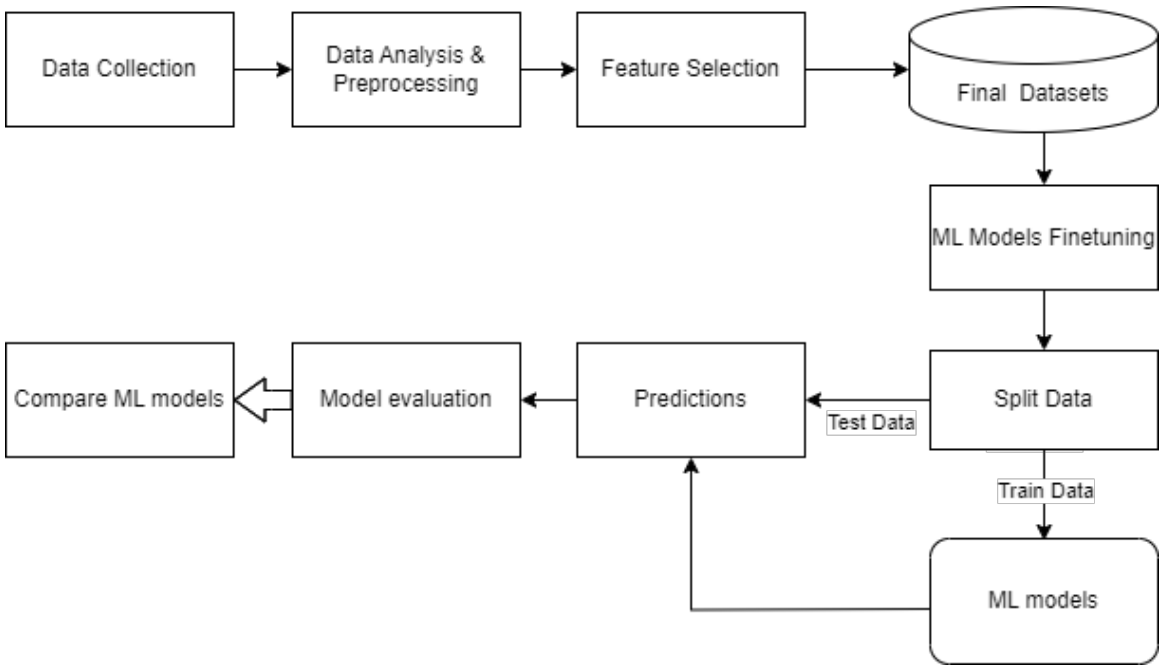

Figure 1.1: Abstract methodology representation of Thesis

1.4 Document Organization

Chapter	Description
Chapter 2	This chapter describes the background knowledge for this thesis. Specifically, it goes through an overview of cryptocurrency. It also describes the machine learning algorithms that were investigated, the algorithms that we use for selecting the features of our data, as well as the metrics used to evaluate the performance of the prediction models.
Chapter 3	This chapter focuses on data collection, processing, and analysis. It describes how the data was collected, the processing steps of the data, and also explains how we select our features of the datasets and present the final data that we are going to use for the models.
Chapter 4	This chapter describes the methodology followed for the regression models of this thesis. It contains information about fine-tuning, selection, and performance evaluation for both machine learning and deep learning algorithms. Additionally, it includes a comparison of the models based on evaluation metrics and provides a commentary on the results.
Chapter 5	This chapter describes the methodology followed for the regression models of this thesis. It contains information about fine-tuning, selection, and performance evaluation for both machine learning and deep learning algorithms. Additionally, it includes a commentary on the results.

Table 1.1: Chapter Descriptions

2 Background Knowledge

Contents

2.1 Introduction to Cryptocurrencies **6**
2.1.1 Understanding Cryptocurrencies: Key Questions 6
2.1.2 Crypto100 Index Price 7
2.2 Machine learning Algorithms **7**
2.2.1 Linear Regression 8
2.2.2 K-Nearest Neighbors 8
2.2.3 Decision Tree 9
2.2.4 Random Forest 10
2.2.5 AdaBoost 10
2.2.6 XGBoost 11
2.2.7 Support Vector Regression 11
2.3 Deep Learning Algorithms **12**
2.3.1 Recurrent Neutal Network 13
2.3.2 Long Short-term Memory 13
2.3.3 Convolutional Neural Netowrk 14
2.4 Feature Importance Algorithms **16**
2.4.1 Mean Decrease Impurity (MDI) 16
2.4.2 Permutation Feature Importance 16
2.4.3 SHAP Algorithm 16
2.5 Evaluation Metrics **18**
2.5.1 Root Mean Squared Error 18
2.5.2 Mean Absolute Error 18
2.5.3 Mean Absolute Percentage Error 19
2.5.4 Median Absolute Error 19
2.5.5 R-squared 20

This section covers concepts related to the research for this thesis. First, it provides an overview of what cryptocurrency is. Afterward, it provides a brief overview of machine learning and discusses the specific algorithms employed. Additionally, it examines the algorithms used in data analysis for feature importance and highlights the evaluation metrics.

2.1 Introduction to Cryptocurrencies

Cryptocurrencies are digital or virtual currencies that use cryptography for security, operating on decentralized platforms using blockchain technology [16]. The most well-known cryptocurrency is Bitcoin [1], but there are thousands of others, including Ethereum [17] and Litecoin [18]. Cryptocurrencies function without a central authority or banks, instead, transactions are verified by network nodes through cryptography and recorded in a publicly distributed ledger called a blockchain.

2.1.1 Understanding Cryptocurrencies: Key Questions

In the next paragraphs, we will try to answer some key questions to help us understand cryptocurrencies better.

What is the Story of Bitcoin? Bitcoin is the first and most well-known cryptocurrency. It was proposed and created by an anonymous person or group of people using the pseudonym Satoshi Nakamoto in 2009 [19]. Bitcoin was designed as a peer-to-peer electronic payment system that relies on cryptographic proof rather than trust. In April 2010, Bitcoin was first publicly traded at a price of approximately €0.03. By January 2013, Bitcoin's price still did not exceed €12, with an intermittent peak close to €900 in December of the same year. Today, in March 2024, Bitcoin's price is approximately €57,000.

How Do Cryptocurrencies Work? A blockchain is a distributed ledger that records all transactions across a network of computers. Each transaction is grouped into a block, and these blocks are linked together in a chain, hence the term "blockchain". This technology ensures transparency and security, as every transaction is publicly recorded and cannot be altered once added to the blockchain.

Why Does Bitcoin Consume So Much Energy? Bitcoin consumes a large amount of energy primarily due to its Proof-of-Work (PoW) [20] consensus mechanism. Miners solve complex mathematical problems to validate transactions and secure the network, requiring powerful computers that run continuously. This competitive mining process drives the use of more powerful hardware and significant electricity consumption. Although energy-intensive, this process ensures Bitcoin's security and decentralization.

Why is Bitcoin Important and Why is its Price So Volatile? Bitcoin plays a crucial role as a digital store of value and medium of exchange, widely used for transactions, investments, and protection against inflation. However, Bitcoin's price is very volatile, influenced by factors such as market demand, investor sentiment, regulatory news, technological advancements, and macro-economic conditions. This volatility, combined with its global 24/7 trading environment, makes predicting Bitcoin's price challenging, requiring intense analysis and an understanding of various influencing factors.

2.1.2 Crypto100 Index Price

The Crypto100 Index [4] is a benchmark that tracks the performance of the top 100 cryptocurrencies by market capitalization [21]. Market capitalization, or "market cap," is calculated by multiplying the current price of a cryptocurrency by its total supply of coins. It serves as an indicator of the overall health and trends of the cryptocurrency market. In this thesis, the Crypto100 Index is the target of our prediction efforts. The index value is calculated using the following equation:

$$\text{Crypto100 Index} = \frac{\sum_{i=1}^{100} \text{Market Cap}_i}{\left(\log_{10}\left(\sum_{i=1}^{100} \text{Market Cap}_i\right)\right)^7},$$

where Market Cap_i is the market capitalization of the i-th cryptocurrency.

2.2 Machine learning Algorithms

Machine learning is a sub-field of AI (Artificial Intelligence) that has rapidly developed in recent years, due to the availability of big datasets and powerful computing resources [22]. ML focuses on the development of computational algorithms and models that can autonomously learn and improve from data. It includes a collection of statistical methods and algorithms that, without the need for explicit programming, allow computer systems to analyze and find complex patterns and relationships within data. Given that machine learning algorithms learn from data, the quality and quantity of data play a crucial role in their performance [23].

There are three main categories of Machine Learning algorithms: supervised, unsupervised, and reinforcement learning [24]. Supervised learning, often referred to as learning with a teacher, uses labeled data where each data point has a corresponding desired outcome. This allows the algorithm to identify patterns that the data points corresponding to their desired target have and make predictions for new data. Unsupervised learning, on the other hand, deals with unlabeled data. These algorithms discover hidden common patterns and relationships in data without the need for human intervention. A common technique in unsupervised learning is clustering, which

groups similar data points together. Finally, reinforcement learning works through trial and error, receiving rewards for correct actions and penalties for incorrect ones, similar to training under a judge's guidance. Additionally, within supervised learning, we can further categorize problems into two main types: regression for predicting continuous values (e.g., price) and classification for predicting categorical values (e.g., spam/not spam) [25].

In this thesis, we will focus on supervised learning for a regression problem. We will use labeled datasets to train algorithms with the desired output being the price of cryptocurrency on specific dates. Since our goal is to predict a continuous value (price), this falls under the category of regression within supervised learning.

2.2.1 Linear Regression

Linear regression [6] is a statistical method used to model the relationship between a dependent variable y and one or more independent variables x. It assumes a linear relationship between the independent and dependent variables, which can be represented by the equation:

$$y = \beta_n x_n + \cdots + \beta_1 x_1 + \beta_0$$

In this equation:

- y represents the dependent variable we want to predict.
- x_i represents the i-th independent variable.
- β_i represents the coefficient associated with x_i.
- β_0 is the intercept term, representing the value of y when all independent variables are 0.

Linear regression aims to estimate the values of β_i coefficients that best fit the observed data, minimizing the overall error between the observed and predicted values of y. Once the coefficients are estimated, the model can be used to make predictions for new values of x by plugging them into the equation.

2.2.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) [9] is a non-parametric regression technique that predicts continuous values based on similarity. It assumes data points close together in the feature space likely share similar target values. When presented with a new data point, KNN identifies the k closest data points (neighbors) from the training data based on a chosen distance metric, such as Euclidean, Manhattan, or Minkowski distance. Finally, it predicts the target value for the new point by averaging the target values of its k neighbors. This approach essentially

estimates the target value based on the ”local trend” within the region defined by the k nearest neighbors. The appropriate number of neighbors (k) is crucial for optimal performance, requiring careful selection to avoid underfitting or overfitting.

2.2.3 Decision Tree

In the context of regression, Decision Trees [8] partition the feature space into regions, with each region representing a decision or prediction. The tree structure is constructed recursively by selecting the feature and the split point that best separates the data into homogeneous groups, aiming to minimize the variance of the target variable within each partition. An illustration of the decision tree structure can be seen in Figure 2.1, where each internal node makes a decision based on a feature, and each leaf node provides a prediction.

Mathematically, at each node of the decision tree, the algorithm calculates the entropy before and after a potential split. Entropy measures the impurity or disorder of a dataset, and it is calculated using the formula:

$$\text{Entropy}(t) = -\sum_{i} p_i \log_2(p_i)$$

Where:

- p_i represents the proportion of samples in class i at node t.

Information gain measures the reduction in entropy achieved by a particular split. It is calculated as the difference between the entropy before the split and the weighted average of the entropies of the resulting child nodes:

$$IG(t) = \text{Entropy}(t) - \sum_{i} \frac{N_i}{N} \text{Entropy}(t_i)$$

Where:

- N is the total number of samples at node t
- N_i is the number of samples in the ith child node
- t_i represents each child node.

According to the above equation, the decision tree algorithm selects the feature and split point that maximizes the information gain until a stopping criterion is met. When making predictions for new data points, the decision tree traverses from the root node to a leaf node, and the prediction is typically the mean or median of the target values of the training samples falling into that leaf node.

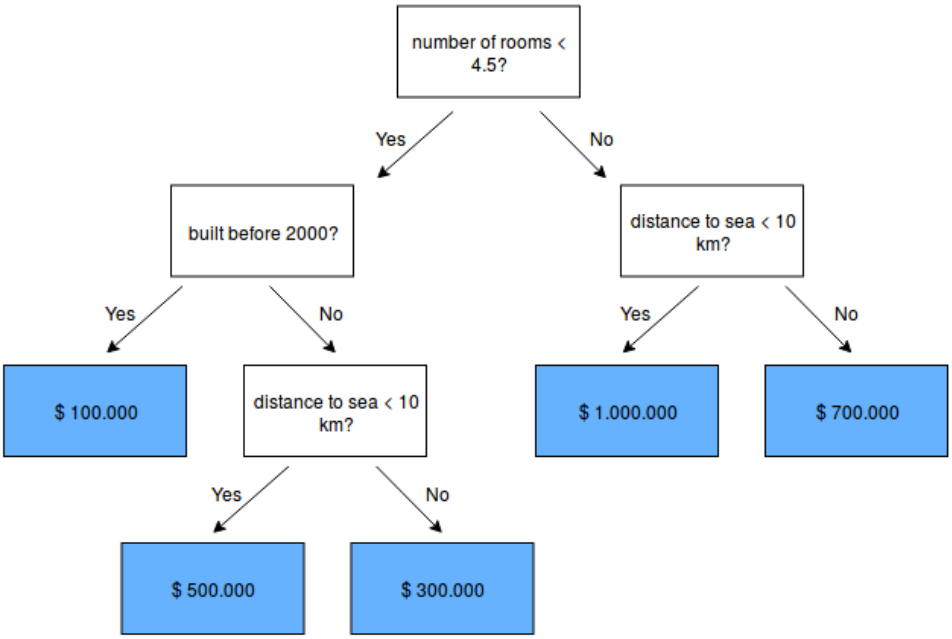

Figure 2.1: Example of a Regression Decision Tree

2.2.4 Random Forest

Random Forest [10] constructs multiple decision trees during training and combines their predictions. Unlike a single decision tree, Random Forest introduces randomness in two key aspects: the selection of data points used for training each tree and the selection of features considered for splitting at each node. This randomness helps to reduce overfitting and improve generalization performance. Each decision tree in the forest independently produces a prediction for the input data point, and the final prediction of the Random Forest is obtained by aggregating the predictions of all individual trees. Random Forest is known for its robustness, scalability, and ability to handle complex data relationships, making it widely used in various fields. Additionally, it provides insights into feature importance, aiding in data pattern interpretation.

2.2.5 AdaBoost

AdaBoost, short for Adaptive Boosting [7], is an algorithm that combines multiple weak learners, typically decision trees, to make predictions. Unlike Random Forest, which constructs multiple trees independently, AdaBoost builds trees sequentially, with each subsequent tree focusing more on the training instances that the previous trees poorly predicted. During each iteration, AdaBoost assigns higher weights to instances with larger residual errors, which are the differences between predicted and actual values, effectively ”boosting” their influence on subsequent trees to correct errors and improve overall

performance. Consequently, the final prediction is obtained by weighing the predictions of all individual weak learners, with more weight assigned to those with higher accuracy. The boosting process continues until a predefined stopping criterion is met, such as reaching a maximum number of iterations or when the model's performance no longer improves.

2.2.6 XGBoost

XGBoost [12] is an advanced implementation of the gradient boosting algorithm used for regression tasks. XGBoost, similar to AdaBoost, sequentially creates a series of decision trees. Each new tree aims to correct the errors of its predecessors by focusing on the residuals from prior models. However, XGBoost differs in its approach. It utilizes a gradient boosting approach, where during each iteration, it calculates the gradients, representing both the direction and magnitude of error, concerning the predictions of the previous model. This information guides the construction of the new tree, enabling it to specifically focus on correcting the errors made by its predecessors.

2.2.7 Support Vector Regression

SVR [11] works by finding the optimal hyperplane that effectively fits the data points while maximizing the margin between the hyperplane and the closest data points, known as support vectors. This can be done by mapping the input data points into a higher-dimensional feature space using a kernel function, where the optimal hyperplane is determined. The kernel function allows SVR to efficiently handle nonlinear relationships between features.

SVR regression aims to minimize the following kernel function:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(|y_i - f(x_i)| - \epsilon)_+,$$

where:

- $||w||$ is the norm of the weight vector,
- C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the error,
- $f(x_i)$ is the predicted output for the ith data point x_i,
- y_i is the true output for the ith data point,
- ϵ is a margin parameter that specifies the maximum deviation tolerated for a data point to be considered in the margin.

By adjusting the parameters C and ϵ, SVR regression can be fine-tuned to achieve the desired balance between model complexity and generalization.

2.3 Deep Learning Algorithms

Deep learning [26] is a subset of machine learning that relies on NNs (Neural Networks), mathematical models inspired by the structure and function of the human brain. NNs include layers of interconnected nodes, called neurons. The first mathematical model of a neuron was introduced by McCulloch and Pitts in 1943 [27]. As illustrated in Figure 2.2, each neuron connects to others and has its own associated weight. The idea is that a single neuron in the human brain receives thousands of signals from other neurons. In NN, signals travel between nodes and assign corresponding weights. A heavily weighted node will exert more influence on the next layer of nodes. The final layer compiles the weighted inputs to produce an output. The term 'deep' in deep learning refers to the use of multiple layers in the network. During training, these weights are adjusted iteratively to minimize the difference between the predicted outputs and the actual targets. This architecture enables DL models to learn complex patterns and make predictions based on data.

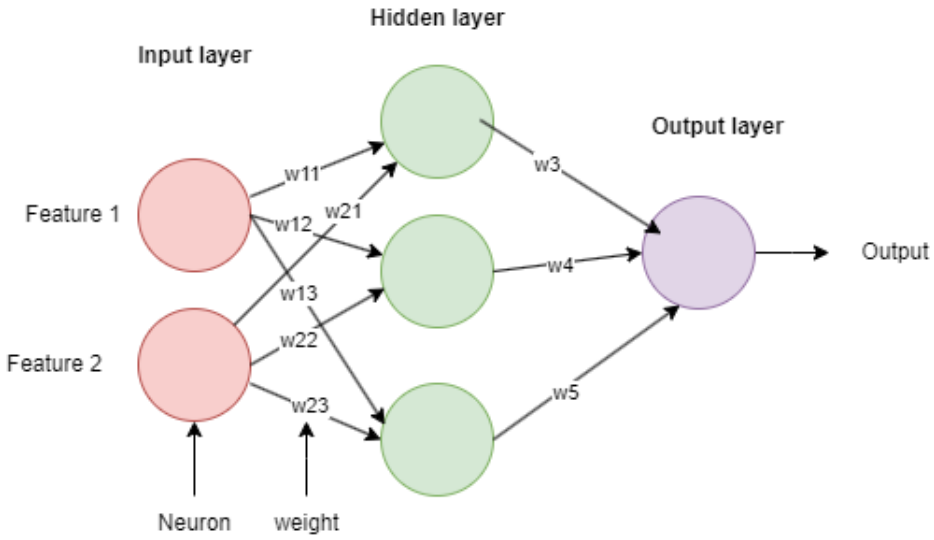

Figure 2.2: Example of the architecture of a simple NN

2.3.1 Recurrent Neutal Network

Recurrent Neural Networks (RNNs) [15] are a type of neural network designed to handle sequential data, making them particularly effective for tasks like time series prediction, language modeling, and speech recognition. Unlike traditional neural networks, RNNs have a unique architecture that allows them to consider past information. RNN has loops where the information is transmitted back into itself, allowing taking into account the previous input along with the current input. This enables RNNs to handle sequential data. When training an RNN, the network's parameters are adjusted through a process called back-propagation, where the error signal is propagated backward from the output to the input layers. Gradients are computed, indicating how the error changes relative to the network's parameters. These gradients are then used to update the parameters and improve the network's predictions. However, RNNs often encounter challenges in capturing long-term dependencies due to the vanishing gradient problem, where gradients diminish as they propagate back through time during training.

2.3.2 Long Short-term Memory

Long Short-Term Memory (LSTM) networks [13] are a specialized type of recurrent neural network (RNN) designed to address the limitations of traditional RNNs in capturing long-term dependencies. LSTMs like RNNs are particularly effective for tasks involving sequential data.

We will start explaining the components of the LSTM architecture as shown in Figure 2.3. First, at the core of an LSTM network are memory cells that can maintain information over long periods of time. These memory cells are equipped with three essential components: the input gate, the forget gate, and the output gate. Each gate is responsible for controlling the flow of information into, out of, and within the memory cell.

During the forward pass of training or inference, information enters the memory cell through the input gate, which determines what new information from the current input to store in the cell. The forget gate decides what information from the previous cell to forget. Finally, the output gate controls which information to output from the memory cell to the next time step or layer in the network.

Furthermore, the LSTM gates employ activation functions such as tanh and sigmoid [28], crucial for controlling the flow of information. The sigmoid function returns values between 0 and 1, to represent the opened or closed status of the gates. The tanh function returns values ranging from -1 to 1, to represent the strength of the memory cell state.

Additionally, in LSTMs network there are two states: the cell state and the hidden state. The cell state represents the memory of the network, storing information over time, while the hidden

state contains the information that is passed to the next time step.

Lastly, the LSTM network like RNN leverages backpropagation to iteratively adjust its parameters during training, aiming to optimize performance and accuracy.

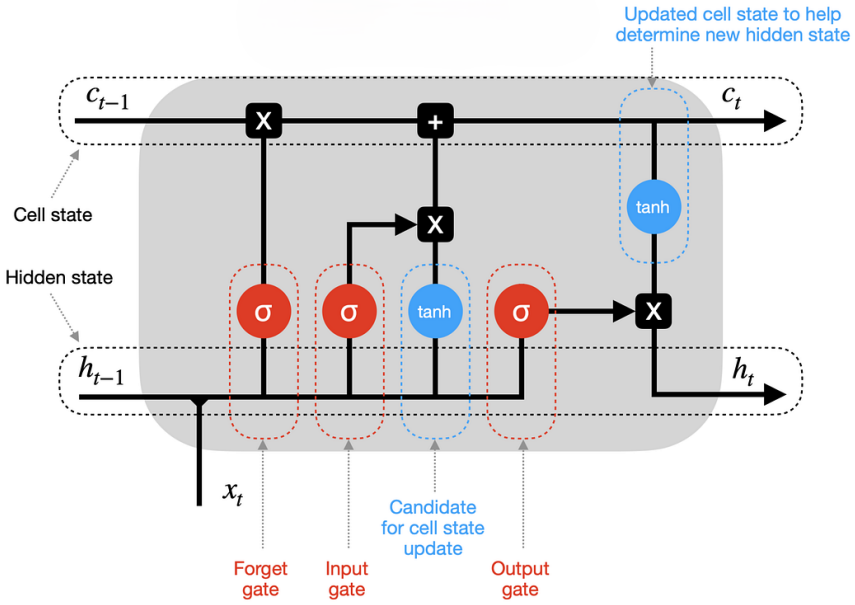

Figure 2.3: The architecture of LSTM network [29]

2.3.3 Convolutional Neural Netowrk

Convolutional Neural Networks (CNNs) [14] are particularly good at capturing spatial patterns and features in data, making them ideal for tasks like image recognition and classification. However, CNNs can also be adapted to analyze other types of data, such as time series data. In our case, we want to use the CNN for a regression problem predicting the price of crypto. CNNs can help us identify patterns and relationships sequentially, treating the data as an image, and extract meaningful features. They can also capture local patterns and temporal dependencies in the data.

A CNN contains several types of layers as displayed in Figure 2.4.

Convolutional Layer: This layer applies a set of learnable filters (kernels) to the input data. Each filter extracts specific features from the input, such as edges, textures, or shapes, by performing convolution operations across the input.

Activation Function: After the convolution operation, an activation function like ReLU [28] is applied element-wise to introduce non-linearity into the network and enable it to learn complex patterns.

Pooling Layer: The pooling layer reduces the spatial dimensions of the feature maps produced by the convolutional layer while retaining the most important information. This helps reduce computational complexity and prevents overfitting by introducing spatial invariance.

Flattening: The output of the pooling layer is flattened into a one-dimensional vector to be fed into the fully connected layers of the network.

Fully Connected Layers: These layers perform classification or regression tasks by learning complex relationships between the features extracted by the convolutional layers and the target variable. This layer includes a FeedForward Network, such as a Multilayer Perceptron (MLP).

Output Layer: The output layer produces the final prediction or regression value.In this case represents the predicted price of the cryptocurrency.

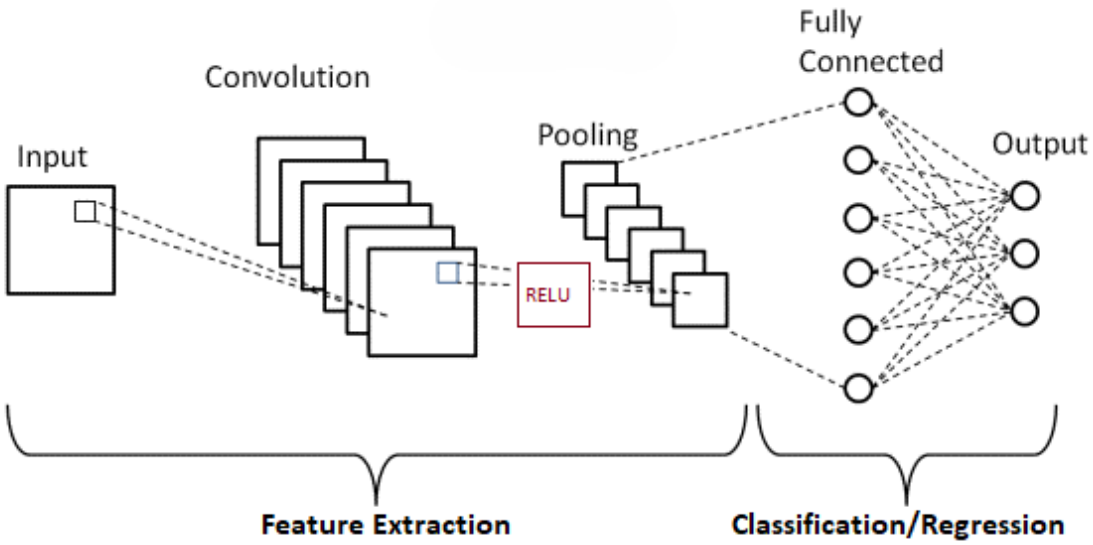

Figure 2.4: The architecture of CNN [30]

2.4 Feature Importance Algorithms

Feature importance algorithms help in identifying the most significant features in a dataset that contribute to the prediction of the target variable. In this section, we briefly explain the three algorithms that we use in this thesis: Mean Decrease Impurity (MDI), Permutation Feature Importance, and the SHAP algorithm.

2.4.1 Mean Decrease Impurity (MDI)

Mean Decrease Impurity (MDI) [31] is a method used to evaluate the importance of a feature based on how much it decreases the impurity in a decision tree model. In regression tasks, impurity is often measured using metrics such as mean squared error (MSE) or mean absolute error (MAE), reflecting the variance within the nodes.

In this thesis, we use both Random Forest and XGBoost regression models to determine feature importance based on their contribution to reducing the prediction error. The MDI for a feature is computed as the average reduction in error across all trees in the forest for Random Forest or across all boosting rounds for XGBoost. Features that result in a greater reduction in error are considered more important.

2.4.2 Permutation Feature Importance

Permutation Feature Importance [32] evaluates the importance of a feature by measuring the increase in prediction error after randomly shuffling the feature's values among the instances. The idea is that if a feature is important, shuffling its values will disrupt the model's predictions and lead to a significant increase in error. This method can be applied to any regression model and is particularly useful because it considers the feature's interaction with other features. By permuting the features and observing the change in model performance, this approach offers a robust way to identify the most critical features in a dataset.

2.4.3 SHAP Algorithm

The SHAP (SHapley Additive exPlanations) algorithm [5] is a promising tool for explainable AI, making machine learning models more transparent and understandable. SHAP values measure the contribution of each feature to a model's prediction, explaining why a certain decision was made. This transparency is particularly important in sensitive applications such as finance, healthcare, and legal systems, where understanding the reasoning behind a decision is crucial.

The way that SHAP works is by utilizing a game-theoretic approach, similar to how players' contributions to a final outcome are measured in a game. Each feature is assigned an importance value that represents its impact on the model's output. Also, SHAP values can highlight the significance of each feature compared to others and reveal how feature interactions influence the model's decisions.

Let's see an example of the contribution of a feature in our scenario using SHAP for predicting our target as Figure 2.5 shows. Consider the metric AdrBalUSD1Cnt, which is the number of active users on the network who have a balance greater than or equal to $1 USD. We can see that most of the high values (red points) of AdrBalUSD1Cnt affect the output of the model positively (pushed to higher value predictions), while smaller values (blue points) have a negative impact (pushed to lower value predictions). This can be explained by the fact that a rise in AdrBalUSD1Cnt indicates more users are holding the cryptocurrency as a result of growing interest and therefore an increase in price.

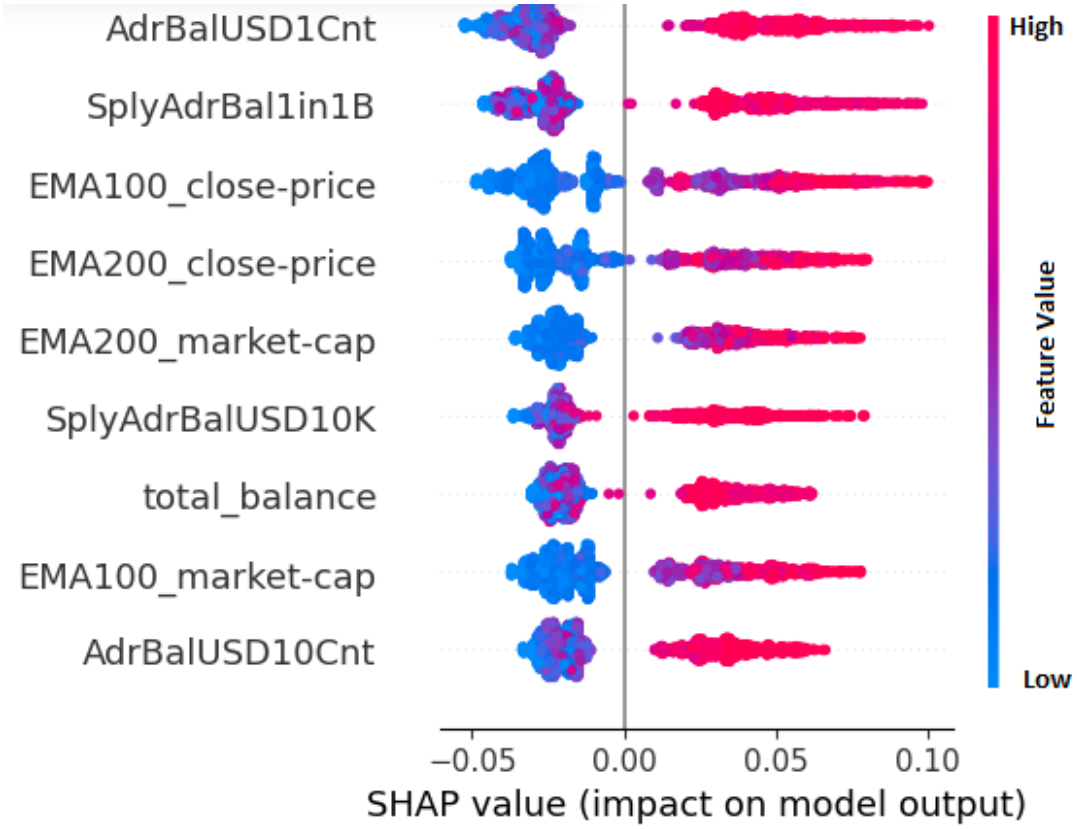

Figure 2.5: Example of SHAP Values of some features for Crypto100 Index prediction

2.5 Evaluation Metrics

Evaluation metrics play a crucial role in evaluating the performance of predictive models. Relying just on a single metric can provide an incomplete picture. Employing a variety of metrics offers a better understanding of each model's strengths and weaknesses, and helps us make meaningful comparisons of the potential of algorithms.

2.5.1 Root Mean Squared Error

The Root Mean Squared Error (RMSE) [33] measures the square root of the average squared difference between the predicted values and the actual values. In simpler terms, it quantifies how far, on average, the model's predictions deviate from the actual values we're trying to predict. The square root is applied to ensure that the result is in the same unit as the target variable. Lower RMSE values indicate better model performance. RMSE is calculated using the formula:

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

Where:

- n is the number of observations,
- y_i is the actual value of the target variable for the ith observation,
- $\hat{y}_i$ is the predicted value of the target variable for the ith observation.

2.5.2 Mean Absolute Error

The Mean Absolute Error (MAE) [33] measures the average absolute difference between the predicted values and the actual values. In simpler terms, MAE measures the average magnitude of the errors between the predicted and actual values, without considering their direction. Unlike RMSE, which penalize larger errors more heavily due to squaring, MAE treats all errors equally, making it less sensitive to outliers.

MAE is calculated using the formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- n is the number of observations,

- y_i is the actual value of the target variable for the ith observation,
- $\hat{y}_i$ is the predicted value of the target variable for the ith observation.

2.5.3 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) [33] measures the average absolute percentage difference between predicted and actual values. By taking the absolute value, MAPE avoids the issue of positive and negative errors canceling each other out, providing a clearer picture of the model's performance. A low percentage error signifies that the predicted values are closer to the actual values, indicating better model performance. MAPE is calculated using the formula:

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100$$

Where:

- n is the number of observations,
- y_i is the actual value of the target variable for the i-th observation,
- $\hat{y}_i$ is the predicted value of the target variable for the i-th observation.

2.5.4 Median Absolute Error

The Median Absolute Error (MedAE) [33] calculates the median of these absolute differences. The median is less sensitive to outliers compared to the mean.

MedAE is calculated using the formula:

$$MedAE = \text{median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_n - \hat{y}_n \mid)$$

Where:

- y_i is the actual value of the target variable for the ith observation,
- $\hat{y}_i$ is the predicted value of the target variable for the ith observation.

2.5.5 R-squared

R-squared (R^2) [33] , is a statistical measure that represents the proportion of the variance in the dependent variable that is explained by the independent variables in the model. In simpler terms, R-squared indicates how well the regression model fits the observed data points.

Higher R^2 values signify a better fit, with 0 implying no explanation of the data's variability and 1 indicating perfect prediction. However, negative R² suggests the model performs worse than a simple average line. While R² is valuable, it's crucial to consider other metrics for a completed evaluation.

R^2 is calculated using the formula:

$$R^2 = \frac{ESS}{TSS}$$

Where:

- ESS is the sum of squared differences between the predicted values and the mean of the dependent variable.
- TSS is the sum of squared differences between the actual values and the mean of the dependent variable.

3 Data Collection and Analysis

Contents

3.1 Data Collection **21**

3.2 Data Preprocessing **22**

3.3 Data Analysis **24**

3.3.1 Period Selection 24

3.3.2 Methodology for Feature Selection 25

3.1 Data Collection

We focused on collecting four different types of data, which are considered important indicators for the price of digital assets. Digital assets include cryptocurrencies, tokens, and other forms of digital currency that rely on blockchain technology. The types of data collected are: Technical, Fundamental, Market Psychology & Interest, and On-chain Information. The goal was to collect data from various categories so that, at the end of the analysis, we could choose the best features that most affect our target.

Technical indicators are derived from historical price and volume data and include metrics such as moving averages (which smooth out price data to identify trends), relative strength index (RSI, which measures the speed and change of price movements), and Bollinger Bands (which indicate volatility and price levels relative to historical norms). These indicators help identify market trends, potential reversal points, and price momentum, providing insights into future price movements based on past performance. More specifically, we used Coingecko [34] and Coinmarketcap [35] as sources to gather this data.

Fundamental analysis involves evaluating the intrinsic value of a cryptocurrency by analyzing factors such as the development team, technological innovation, use cases, and market demand. This type of data helps understand the underlying health and potential growth of a cryptocurrency project. We collected fundamental data from various sources including Investing.com [36], FRED (Federal Reserve Economic Data) [37], CME Group [38], ECB (European Central Bank) [39], and Policy Uncertainty [40] websites.

Market Psychology & Interest assesses market sentiment and interest levels through social media activity, search trends, and news sentiment. This includes tracking mentions on platforms like Twitter, analyzing search trends on Google, and sentiment analysis of news articles. These

indicators provide insights into the overall market mood and can predict price movements driven by public perception and sentiment. We gathered this data from sources such as the Fear and Greed Index (production.dataviz.cnn.io and alternative.me) [41, 42], Lunarcrush.com [43], and Augmento.ai [44].

On-chain Information involves analyzing data recorded on the blockchain, such as transaction volumes, wallet addresses activity, and large transactions (whale movements). This type of data helps understand the actual usage and flow of cryptocurrencies, providing a clear picture of the supply and demand dynamics. For on-chain data, we primarily used Coinmetrics.io [45].

By utilizing these diverse and comprehensive sources, we ensured a robust collection of indicators that could significantly impact cryptocurrency price movements, laying a strong foundation for our subsequent analysis and predictive modeling.

3.2 Data Preprocessing

To ensure consistency and reliability in our analysis, we processed the collected data through several key steps. Table 3.1 below shows the number of rows and features for each dataset:

Dataset	**#columns**	**#rows**
On-chain BTC	137	2357
Stable coin dataset (USDC)	94	1694
Fundamental	9	2357
Market Psychology & Interest	42	1377
Technical Analysis (TA) BTC	59	2326
Technical Analysis Crypto100	32	2326
Traditional	65	1625

Table 3.1: Number of rows and features for each dataset

To handle missing values, features with missing values were managed using interpolation or monthly averages. For example, missing daily values were estimated by interpolating between existing values, while for monthly averages, the missing values were replaced with the average value of that month. Features with a large number of missing values were removed entirely to maintain data integrity.

All data was converted to a daily frequency to align with the majority of the datasets. This involved averaging or summing (if it was a count) intraday values for the specific day, and repeating weekly or monthly values for each day within the respective week or month. For instance, hourly data was averaged to get a single value for each day, while weekly or monthly data used the same value for every day within that week or month.

JSON files obtained from APIs were converted to CSV files, and formatted to follow the same rules and structure as the other datasets to ensure uniformity.

We selected the period from 3 January 2017, to 17 June 2023, because, as shown in Figure 3.1, it contains the most features starting from 3 January 2017. Additionally, some of the features we gathered were only available after 4 August 2018, as the sources provided values for these features starting from this date.

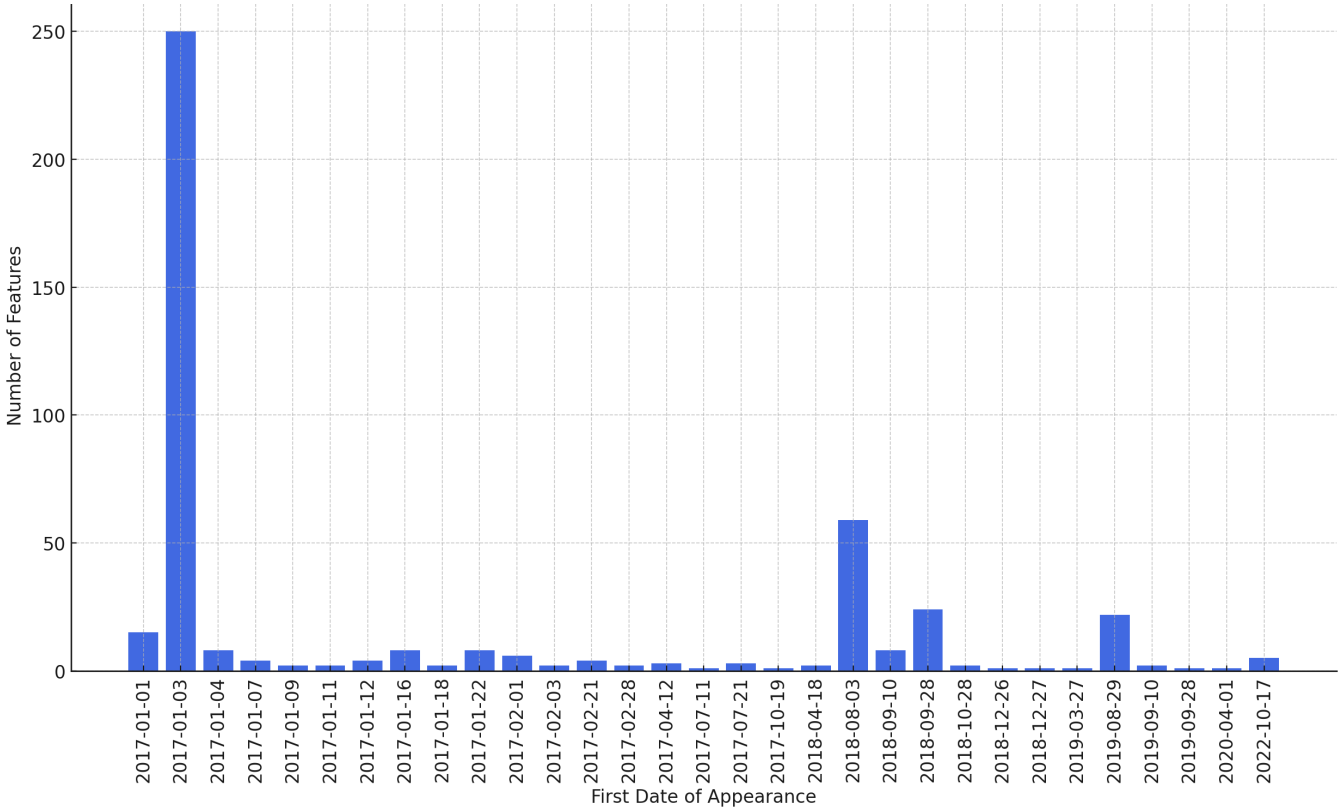

Figure 3.1: Number of Features by Their First Date of Appearance - All Datasets Combined (Normalized Dates)

Also, we decided on our target to be the Crypto100 index and set some future window predictions. We aimed to predict the price at various future points: after 1 day, 7 days, 30 days, 90 days, and 180 days. The latter three window sizes presented a more challenging task, but we aimed to see if the models could identify patterns that help predict future prices over these extended periods.

3.3 Data Analysis

3.3.1 Period Selection

We started our analysis by examining the dates of our datasets. After observing the features from specific dates, as shown in Figure 3.2, we noticed that more features were available after 3 January 2017. To avoid discarding important features, we decided to define two distinct periods for our analysis: from 2017 to 2023 and from 2019 to 2023, with the second period having more features. We chose 2019 as the start of the second period because it marks the beginning of the next market cycle, which is evident from the market bottom depicted in Figure 3.3. This split into two periods led us to create 10 different datasets, as each period dataset will have 5 future window size predictions.

Figure 3.2 categorizes the columns into three groups based on their entry dates:

- *Group 1*: These are features that have their first non-empty data entry before 4 February 2017. This group represents the features with the earliest available data.
- *Group 2*: These features have their first non-empty data entry before 2 January 2019. This group includes features that began recording data after those in Group 1 but before the start of 2019. This group has the most features but with fewer rows.
- *Group 3*: These features have their first non-empty data entry on 2 January 2019 or later.

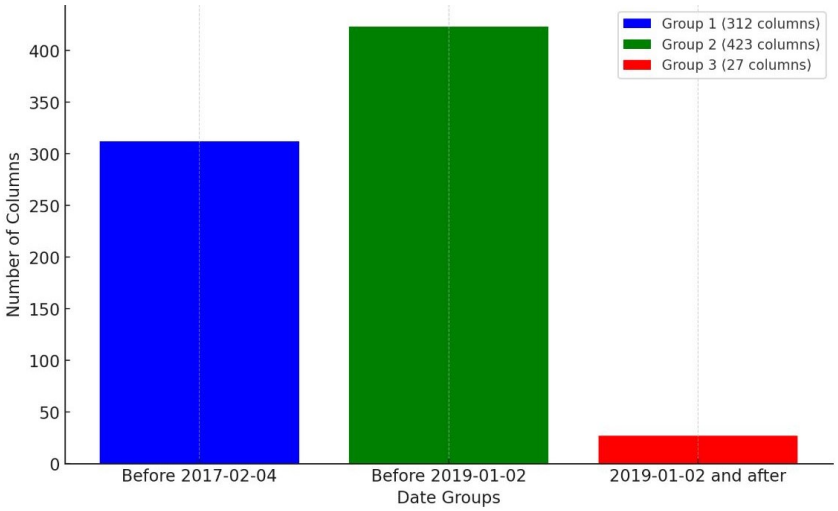

Figure 3.2: Number of Columns Grouped by First Non-Empty Entry Date

Figure 3.3: Market Bottom and Start of New Cycle from 1 January 2019

3.3.2 Methodology for Feature Selection

Our methodology for selecting the best features involved evaluating their importance in relation to the target variable, which is the price for the specific future windows we decided on (after 1, 7, 30, 90, 180 days). Initially, we had 429 features, and after the first analysis, we reduced this number to 329. The algorithms we used were the Mean Decrease Impurity (MDI) feature importance method using Random Forest and XGBoost regression models and Permutation Feature Importance (PFI) using the same models. Initially, both Random Forest and XGBoost models were fine-tuned using grid search with 5-fold cross-validation to optimize their performance for all different windows. Additionally, in the final phase, we used the SHAP algorithm. All the algorithms are explained in Section 2.4. The goal was to remove features that offered no value under any scenario.

To avoid losing any features, we began with individual analyses for each dataset based on their specific start dates as Table 3.2 shows. First, we inspected the line charts of all features against the target variable and we removed features that were practically the same as others. The result was to remove the ”Technical Analysis Crypto100” dataset entirely because it didn’t offer any additional insight, reducing the feature set by 30 features. Additionally, 43 more features were removed across all other datasets.

Dataset	Start Date
On-chain BTC	03/01/2017
Traditional	03/01/2017
Fundamental	03/01/2017
Technical Analysis (TA) BTC	03/02/2017
Technical Analysis Crypto100	03/02/2017
Stable coin dataset (USDC)	28/10/2018
Market Psychology & Interest	10/09/2019

Table 3.2: Individual Analysis Start Dates for Each Dataset

After completing the individual analysis, we continued the process separately for the two periods: 2017-2023 and 2019-2023. For each dataset, we applied the MDI feature importance and PFI algorithms using RF and XGB models for both periods to get the importance value of each feature in each dataset separately based on different algorithms. Following this, we combined all the features of the seven datasets and applied the same methodology for the two periods, considering all the features together to find their importance to the target (for all future window sizes).

To ensure comprehensive feature elimination, we devised an evaluation process that removes features only when they consistently rank at the bottom 50% of features in their individual datasets, across all windows, and do not exhibit any linear correlation with the target variable. Specifically, a feature is removed if all the following conditions are met for all windows:

- It ranks at the bottom 50% when using RF MDI feature importance
- It ranks at the bottom 50% when using XGB MDI feature importance
- It ranks at the bottom 50% when using RF permutation feature importance
- It ranks at the bottom 50% when using XGB permutation feature importance
- The absolute value of the correlation $|\text{Correlation}|$ is less than 0.5

Using this stringent method, which we named the reduction algorithm, we removed an additional 46 features.

At the end of this analysis, our goal was to decrease the feature set to around 100. To achieve this, we used the SHAP algorithm in combination with the results of the reduction algorithm mentioned earlier. We used the combined dataset for each period for the different target windows and obtained the top 100 most important features based on the output of the SHAP algorithm. After comparing the most important features based on the outputs of these two algorithms, we decided to create the final datasets by taking the union of the top 75 features from SHAP and the reduction algorithm explained previously. Table 3.3 shows how many features are left in the final datasets after the final reduction process.

Dataset	Number of Features
2017_1	79
2017_7	79
2017_30	81
2017_90	86
2017_180	88
2019_1	100
2019_7	97
2019_30	100
2019_90	91
2019_180	90

Table 3.3: Number of features in each dataset

The Venn diagrams in Figure 3.4 illustrate the overlap and distinct number of features selected by the reduction algorithm and the SHAP algorithm across different window sizes for both the 2017 and 2019 periods.

Figure 3.4: Comparison between Reduction Algorithm and SHAP Features for Different Window Sizes in 2017 and 2019 Periods

4 Results and Analysis

Contents

4.1 Machine Learning Algorithms **29**
4.1.1 Methodology 29
4.1.2 Results 30
4.2 Deep Learning Algorithms **53**
4.2.1 Methodology 53
4.2.2 Results 53
4.3 Model Comparison **63**

4.1 Machine Learning Algorithms

4.1.1 Methodology

The following methodology was employed to fine-tune the models, prepare the data, and evaluate the performance of the predictive algorithms used in this study.

Initially, we fine-tuned our models to identify the best parameter combinations that yield the highest accuracies. All the code for the methodology, from fine-tuning to training, was implemented using the scikit-learn library in Python [46] and can be found in the GitHub repository [47].

An essential step in our methodology was to normalize the data to ensure better model performance. Specifically, we applied standardization to scale the feature values. Standardization, implemented using the `StandardScaler`, was chosen over normalization (such as min-max scaling) because it does not set a fixed boundary (e.g., 0 to 1). This is particularly important for cryptocurrency prices, which can continually increase and are not constrained by upper limits.

Next, We used the k-fold cross-validation method [48] to split the dataset, where 80% of the data was used for training and 20% for testing, with 5 folds. This method helps in averaging the performance metrics across all folds to provide a more reliable evaluation. Due to the smaller size of the 2019 period dataset, the predictions will be made for fewer days compared to the 2017 period.

To maintain the temporal sequence of the data, we did not shuffle the rows before selecting the folds. We kept the chunks of folds in chronological order to simulate real-world prediction scenarios. This means the test data were from a start date to an end date. After selecting the dates for test and train sets, we then shuffled the data separately within the test and train sets. Shuffling the data before splitting and selecting random dates for test and train sets would lead to information leakage and potentially overfitting, which we aimed to avoid. In all runs, the randomness was initialized with a seed value of 42 to ensure consistent results.

Finally, we chose the following metrics to evaluate our models: MSE, MAE , MedAE, MAPE, and the R^2. For each algorithm, we also provided graphs showing the predicted versus actual prices to facilitate a better understanding of the model performance.

For each algorithm, we also provided graphs showing the predicted versus actual prices (of the last fold) to facilitate a better understanding of the model performance.

4.1.2 Results

4.1.2.1 Linear Regression

Linear regression models were employed without any fine-tuning parameters. The results are summarized in Table 4.1.

Dataset	RMSE	MAE	Median	MAPE	R-squared
2017_1d	6083.65	4812.97	4346.48	15.16%	-1.34
2017_7d	11053.17	9310.20	8806.43	91.89%	-5.40
2017_30d	16061.45	12581.29	9458.39	41.22%	-11.75
2017_90d	14291.36	11540.60	9922.20	50.40%	-10.79
2017_180d	30647.70	23251.28	16952.61	140.54%	-87.09
2019_1d	5772.32	4964.94	5165.49	21.63%	-6.87
2019_7d	24506.65	20114.50	18440.72	164.76%	-185.47
2019_30d	22013.04	16309.29	12899.83	123.12%	-70.84
2019_90d	10741.50	7989.44	6025.64	52.23%	-31.74
2019_180d	17462.14	14834.56	15109.04	111.60%	-48.88

Table 4.1: Linear regression metrics

The linear regression results demonstrate the limitations of linear models in predicting cryptocurrency prices. The RMSE, MAE, and MedAE values indicate significant prediction errors as we can see in Table 4.1.

In Figure 4.1, which illustrates the predictions for the last fold, we observe that the linear regression model's predictions tend to follow the linear trend of the last observed values. If the last few observations show an increasing trend, the model will predict continued increases, and if they show a decreasing trend, the model will predict continued decreases. This behavior is

expected, as linear regression assumes a linear relationship between the input features and the target variable. This linearity leads to over-simplified predictions, failing to capture the complex, non-linear patterns often present in cryptocurrency price movements. As such, while linear regression can provide a basic benchmark, it is insufficient for accurate predictions in this highly volatile and dynamic market.

In the case of predicting the next day's price (first graph in Figure 4.1), the linear regression model performed relatively better. This can be attributed to the fact that short-term predictions are less complex and more likely to follow the recent trend closely. However, as the predictions extend further into the future, the MAPE increases, reaching up to 140% for the 180-day predictions. This means that the average percentage difference between the predicted and actual values is more than twice the actual values.

For the 2019 period, from Figure 4.2 we can see that the model was not able to learn and the results were very poor. This can be explained by the smaller dataset size for the 2019 period, which has fewer rows (a shorter period of data).

Additionally, we can see that as the future window size increases, the errors increase (see Table 4.1), a behavior that was expected as finding relationships with this far window size is more difficult and maybe there are none.

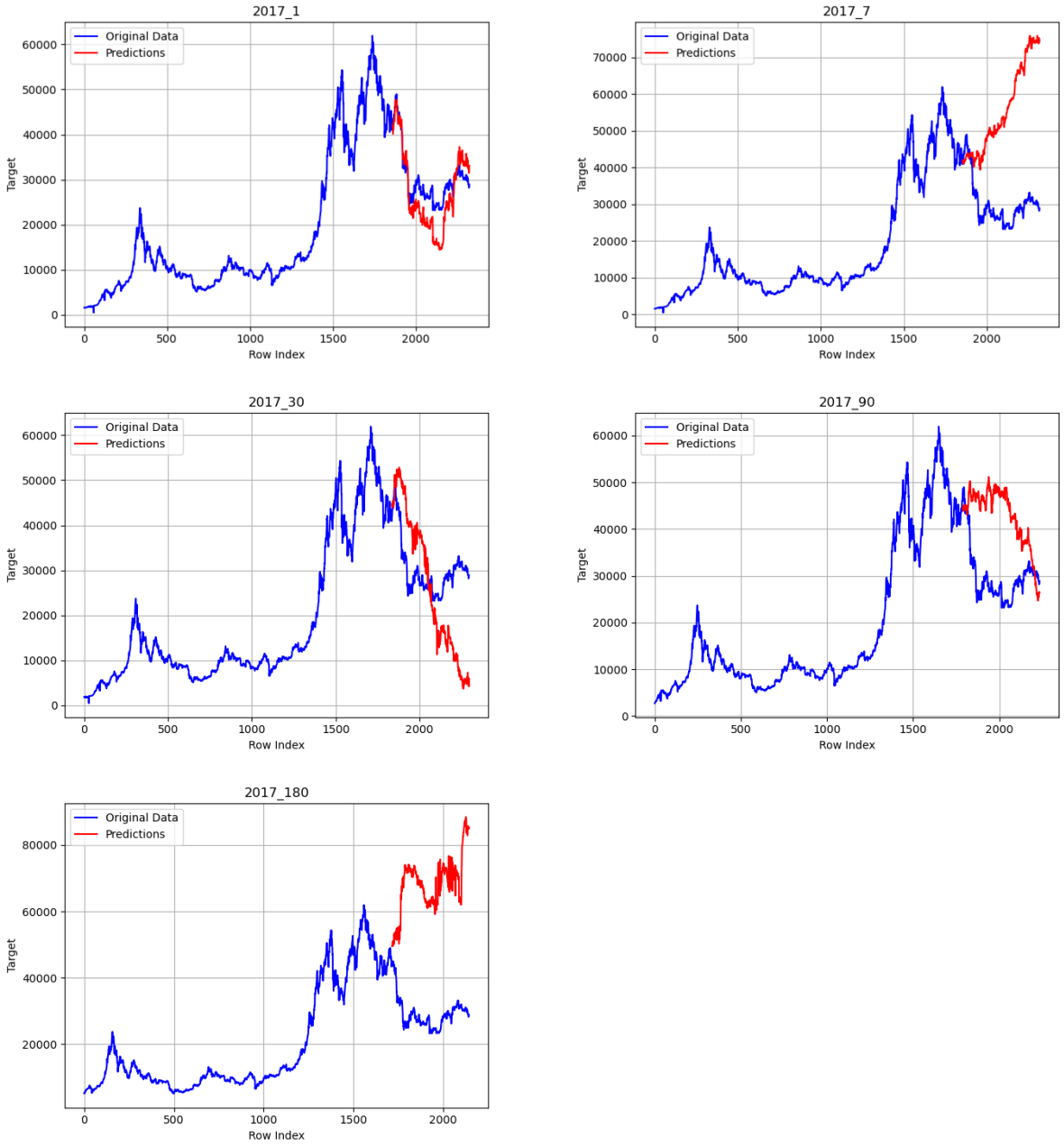

Figure 4.1: Linear Regression Predictions vs Actual Prices for Different Window Sizes in 2017

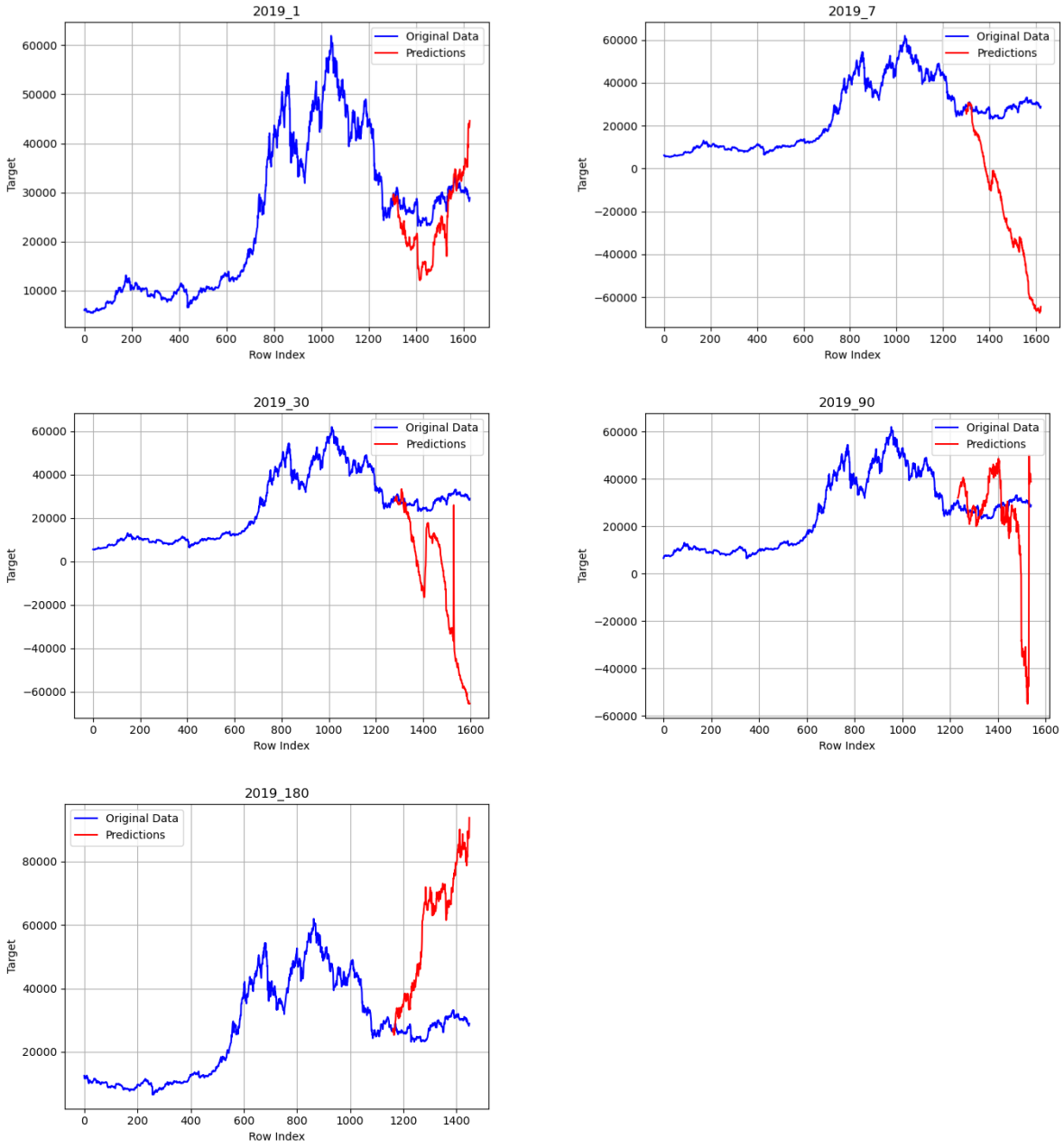

Figure 4.2: Linear Regression Predictions vs Actual Prices for Different Window Sizes in 2019

4.1.2.2 Support Vector Regression

The best hyperparameters found after fine-tuning the SVR models using a grid search approach were: C = 100, epsilon = 0.1, gamma = 'scale', and kernel = 'rbf'.

Dataset	RMSE	MAE	MedAE	MAPE	R-squared
2017_1d	6819.23	5387.73	5077.83	18.48%	-0.90
2017_7d	12072.33	10070.60	9119.78	94.56%	-5.35
2017_30d	7246.34	5959.32	5197.69	30.61%	-1.54
2017_90d	11321.35	9936.23	10801.70	39.39%	-6.87
2017_180d	8440.75	6793.95	5644.62	35.73%	-3.47
2019_1d	3946.26	3286.27	2693.89	4.19%	-2.79
2019_7d	7610.95	5860.91	4312.65	49.07%	-11.24
2019_30d	13104.05	10170.97	7644.24	24.01%	-8.27
2019_90d	9199.74	7965.05	7930.53	29.88%	-44.16
2019_180d	8641.88	7582.59	8301.12	56.72%	-8.36

Table 4.2: Support Vector Regression Metrics

In the results shown in Table 4.2, the SVR model demonstrates very similar behavior to that of linear regression. From these metrics, it is clear that the SVR model generally performs better for the 1-day prediction window compared to longer windows (7, 30, 90, and 180 days). Additionally, in general, the results for the 2019 dataset are poorer than those for the 2017 dataset, the same observation we made using the linear regression model. However, the next-day predictions are better for the 2019 dataset.

The graphical representations in Figures 4.3 and 4.4 demonstrate these observations visually, showing how the model's predictions diverge more significantly from the actual data as the prediction window extends. In general, the SVR model did not produce good results.

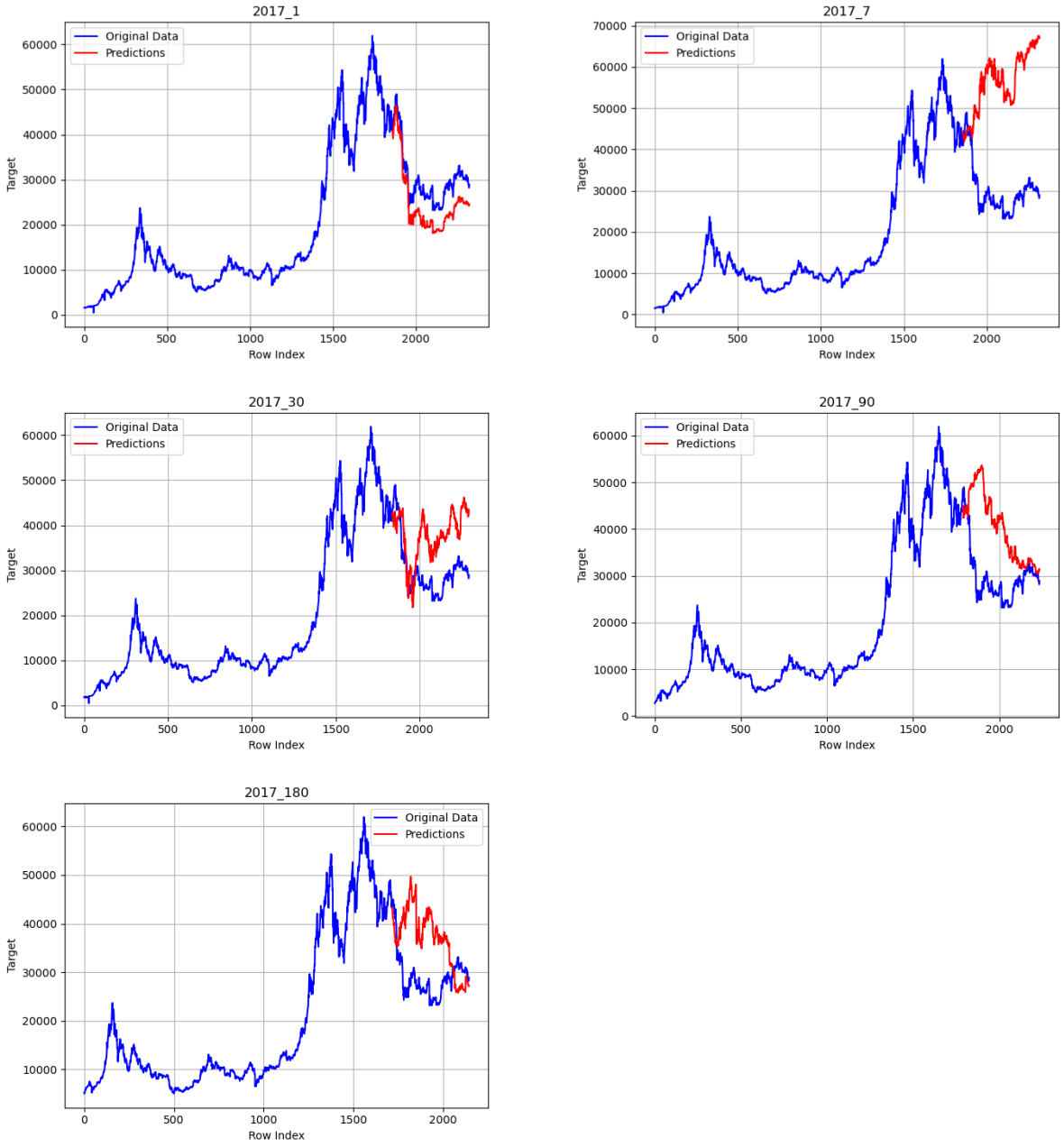

Figure 4.3: SVR Predictions vs Actual Prices for Different Window Sizes in 2017

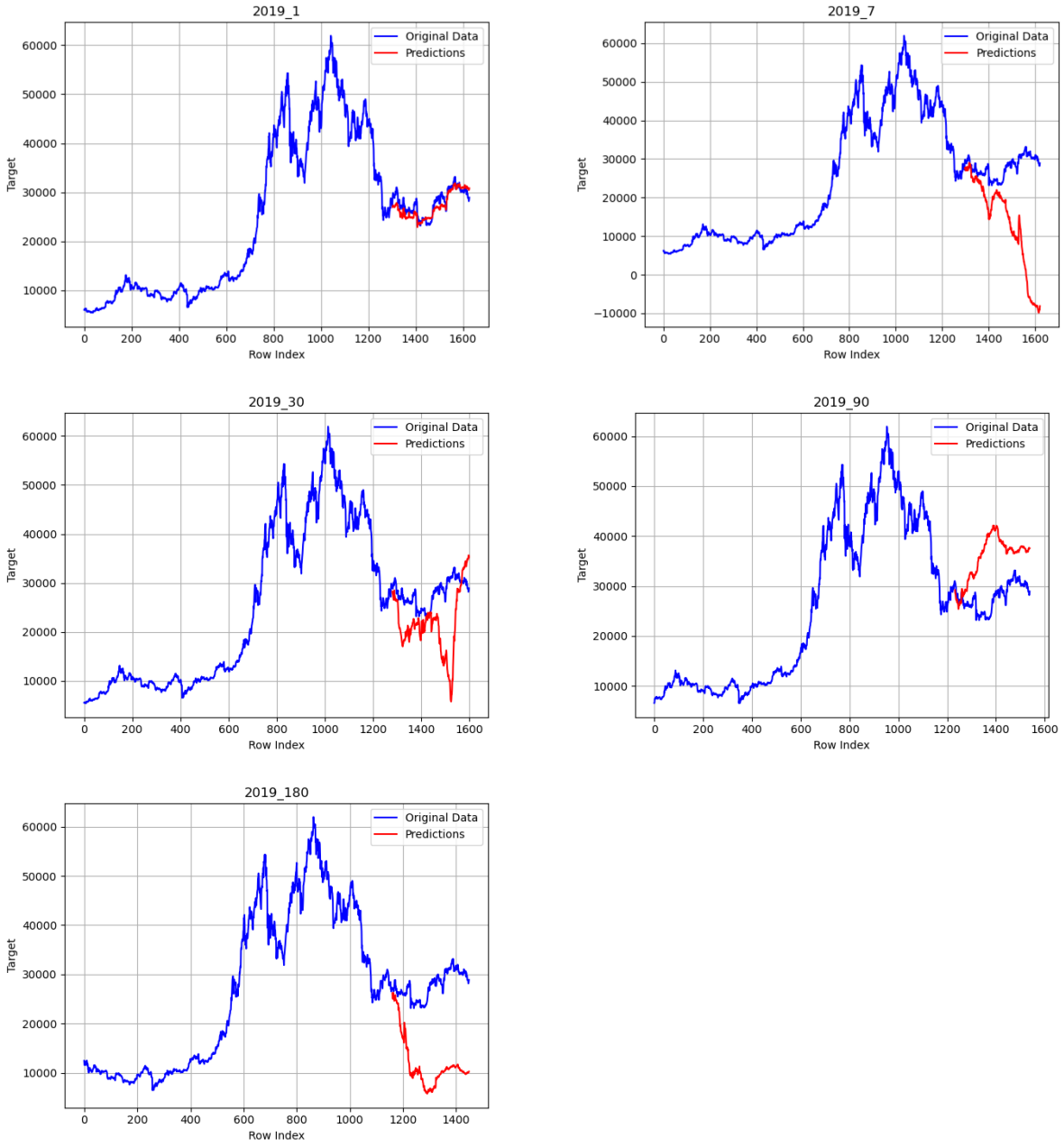

Figure 4.4: SVR Predictions vs Actual Prices for Different Window Sizes in 2019

4.1.2.3 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm was utilized for this analysis, with the best hyperparameters found after fine-tuning being: *algorithm: 'auto'*, *leaf_size: 30*, *n_neighbors: 3*, *p: 1*, and *weights: 'distance'*. Table 4.3 shows the results for both the 2017 and 2019 datasets:

Dataset	RMSE	MAE	Median	MAPE	R-squared
2017_1d	3955.41	3210.55	2955.68	17.84%	0.1981
2017_7d	4722.32	3995.59	3662.87	20.09%	-0.3174
2017_30d	6143.85	4823.21	3906.54	21.92%	-1.2666
2017_90d	8674.72	7270.96	6190.63	51.94%	-3.0902
2017_180d	9417.20	7776.71	6252.56	66.64%	-4.6852
2019_1d	3354.44	2798.92	2623.34	5.96%	0.4165
2019_7d	4185.83	3533.26	3292.53	9.06%	0.1893
2019_30d	5552.74	4454.22	3804.89	10.98%	-0.2416
2019_90d	6740.15	5443.25	4928.78	11.18%	-0.9540
2019_180d	7329.12	6070.03	5263.77	8.38%	-0.9983

Table 4.3: K-Nearest Neighbors Metrics

From the results (Table 4.3 and Figure 4.5), we can observe that the K-Nearest Neighbors (KNN) model generally performs better for the 1-day, 7-day, and 30-day prediction windows for the period 2017, with smaller errors than for the 90-day and 180-day predictions. However, the results are still not satisfactory. As shown in Figure 4.5, the model couldn't identify any real relationship for predictions in the 90-day and 180-day windows.

While the accuracy metrics as Table 4.3 shows for the 2019 dataset may show low error numbers, the graphs (Figure 4.6) reveal that the predictions essentially follow a median line, lacking to capture actual market movements.

Also in the 2017 period for the 1-day, 7-day, and 30-day prediction windows, we can see in Figure 4.5 sudden changes in the predictions. In KNN regression, the predicted value for a data point is determined by the average of the target values of its nearest neighbors. This method can cause sudden shifts in the predictions when the model encounters new data points that are closer to neighbors with significantly different target values.

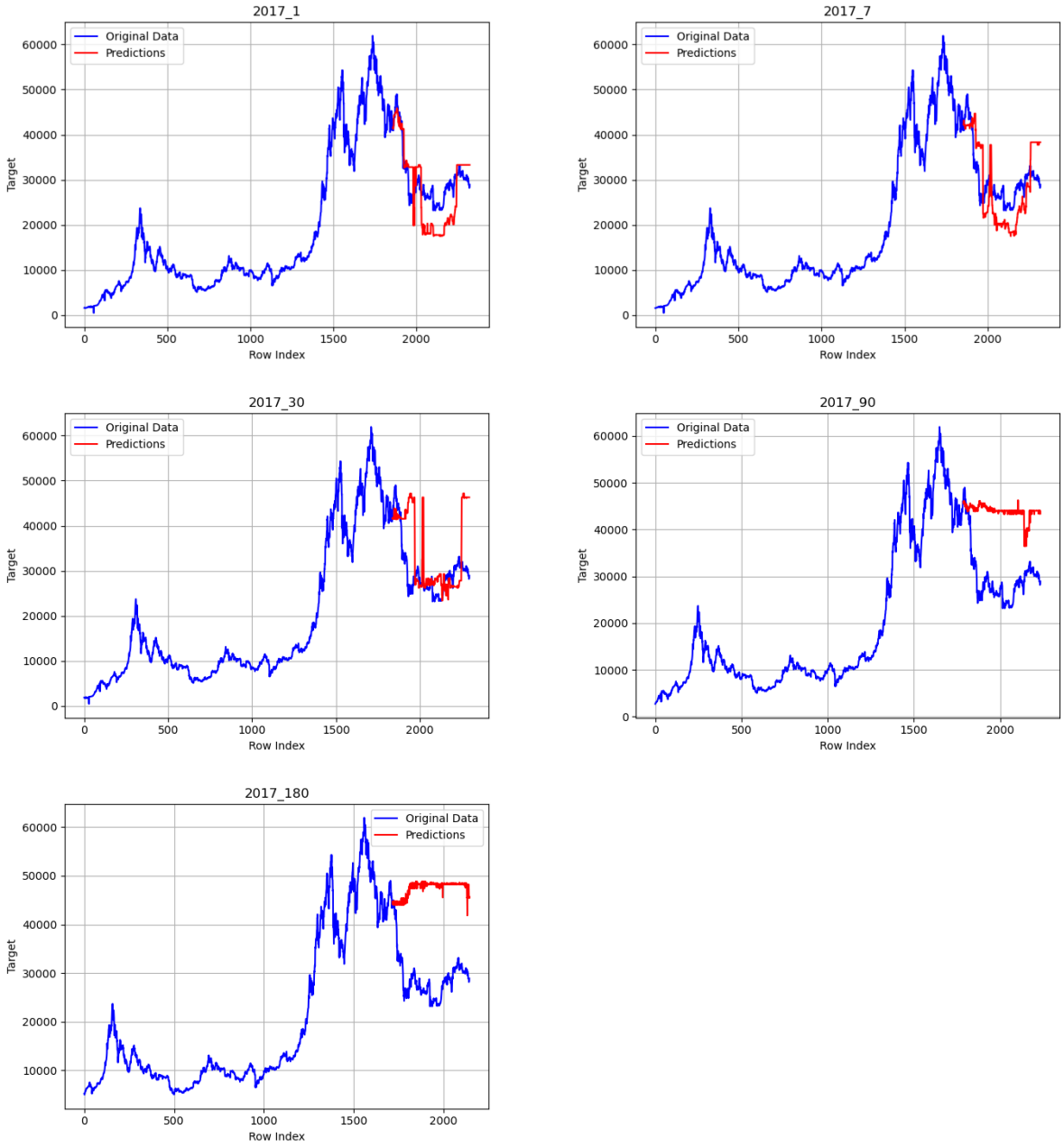

Figure 4.5: KNN Predictions vs Actual Prices for Different Window Sizes in 2017

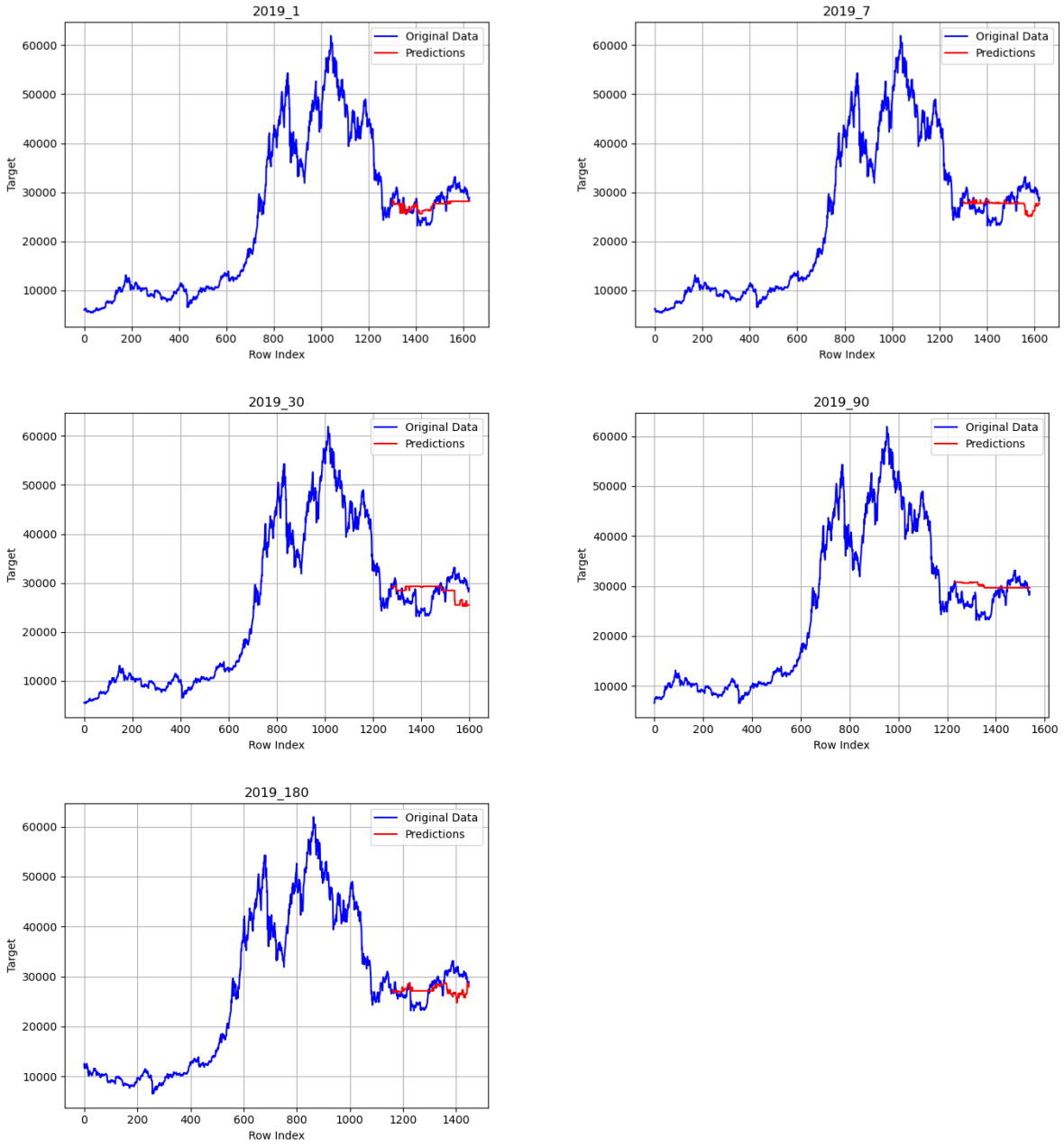

Figure 4.6: KNN Predictions vs Actual Prices for Different Window Sizes in 2019

This and the next regression ML models that we will discuss have the same weaknesses. The models cannot predict prices if the training dataset did not contain data points that indicate that price, as the models will not learn it due to how they work. You can see it in Figure 4.7, where the model predicts the highest point it learned in the training but fails to reach the actual peak. This shows that the ML models could not be able to predict new prices and trends in the cryptocurrency market.

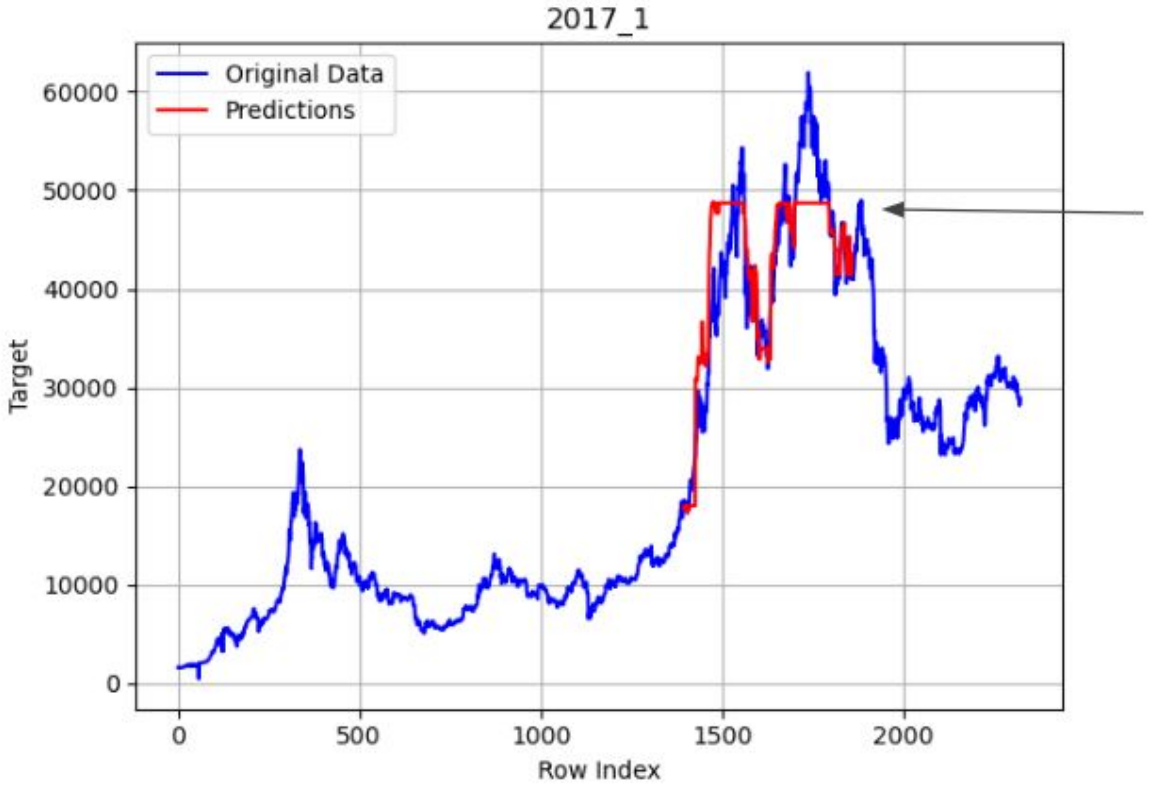

Figure 4.7: ML Model Weakness

4.1.2.4 Decision Trees

The best hyperparameters found after fine-tuning the Decision Tree models using a grid search approach are shown in Table 4.4.

Dataset	Max Depth	Max Features	Min Samples Leaf	Min Samples Split	Splitter
2017_1d	20	None	1	7	best
2017_7d	None	sqrt	1	5	random
2017_30d	20	None	2	2	random
2017_90d	20	sqrt	2	5	random
2017_180d	None	sqrt	2	2	random
2019_1d	10	None	1	5	random
2019_7d	20	log2	1	7	random
2019_30d	None	sqrt	2	2	random
2019_90d	None	sqrt	2	2	random
2019_180d	None	log2	1	2	random

Table 4.4: Decision Tree Best Parameters

Table 4.5 shows the results for both the 2017 and 2019 datasets:

Dataset	RMSE	MAE	MedAE	MAPE	R-squared
2017_1d	4981.89	4272.10	4173.57	41.52%	-0.17
2017_7d	4457.57	3595.02	3025.06	22.25%	0.20
2017_30d	6454.91	5201.57	5144.21	38.97%	-1.17
2017_90d	6792.60	5826.98	5463.53	50.21%	-1.69
2017_180d	8444.61	7208.52	6465.75	62.63%	-3.82
2019_1d	3240.04	2531.43	1954.57	6.28%	0.60
2019_7d	4158.83	3417.66	3100.81	9.47%	-0.02
2019_30d	8931.98	6939.56	3965.87	10.54%	-21.59
2019_90d	6162.90	5082.62	4539.91	9.25%	-1.51
2019_180d	8957.74	7386.61	6485.51	8.47%	-1.79

Table 4.5: Decision Tree Metrics

From a first look at the figures, we can see that the Decision Tree model could not provide any meaningful insights about the price movement, in Figure 4.8, we observe very sudden changes in the predicted prices, with significant variations from one day to the next for the period 2017. This behavior can be attributed to the nature of decision trees, which make predictions based on distinct splits in the data, leading to abrupt changes when a new data point falls into a different leaf of the tree. Additionally, the model shows similar performance issues as seen with other models, particularly for larger prediction windows (90 and 180 days) and the 2019 period dataset (see Figure 4.9). Also, from Table 4.5, we can see that the model achieves better results with

fewer errors and a smaller MAPE of 22.25% for the 7-day prediction compared to the 1-day prediction.

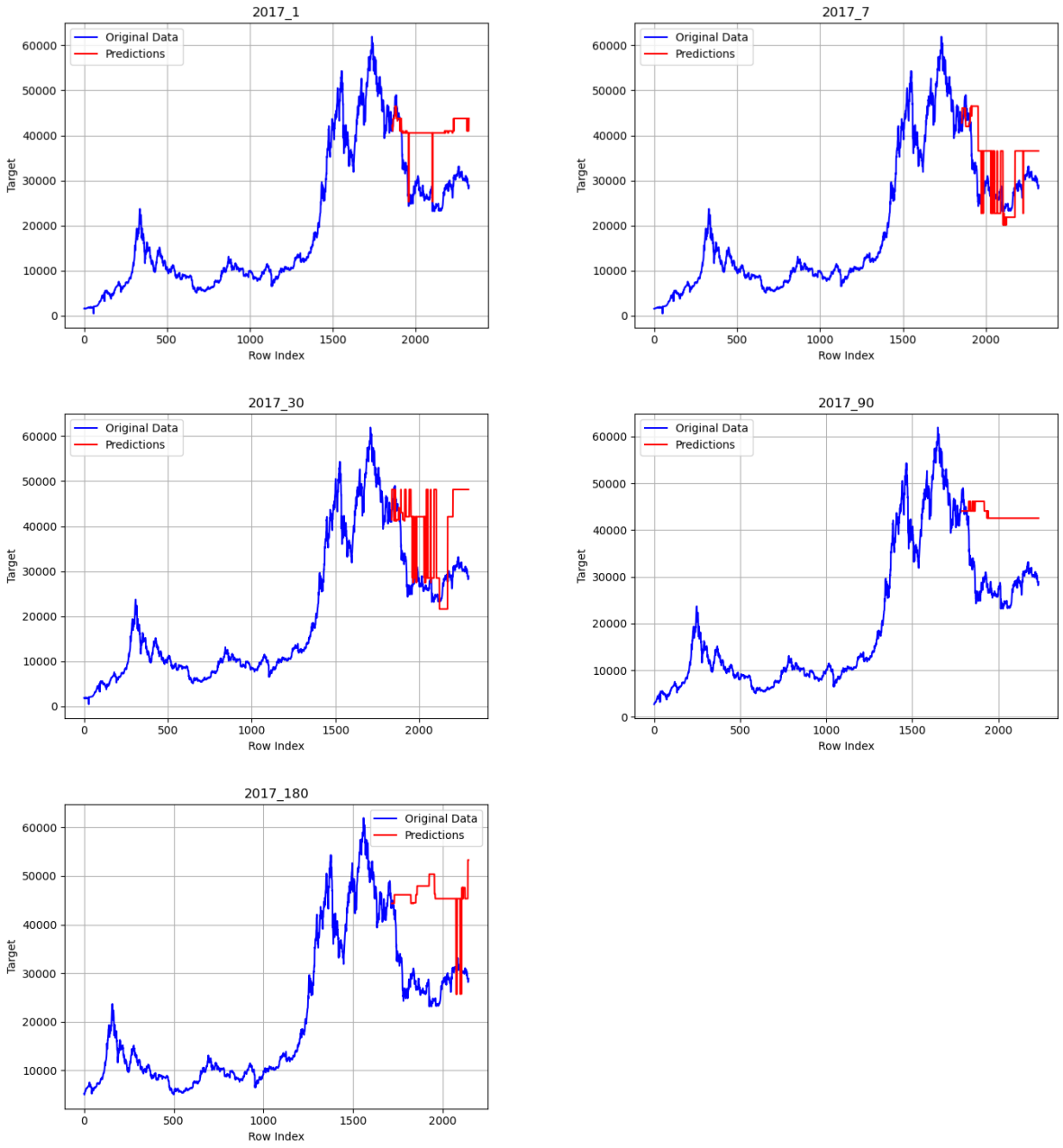

Figure 4.8: Decision Tree Predictions vs Actual Prices for Different Window Sizes in 2017

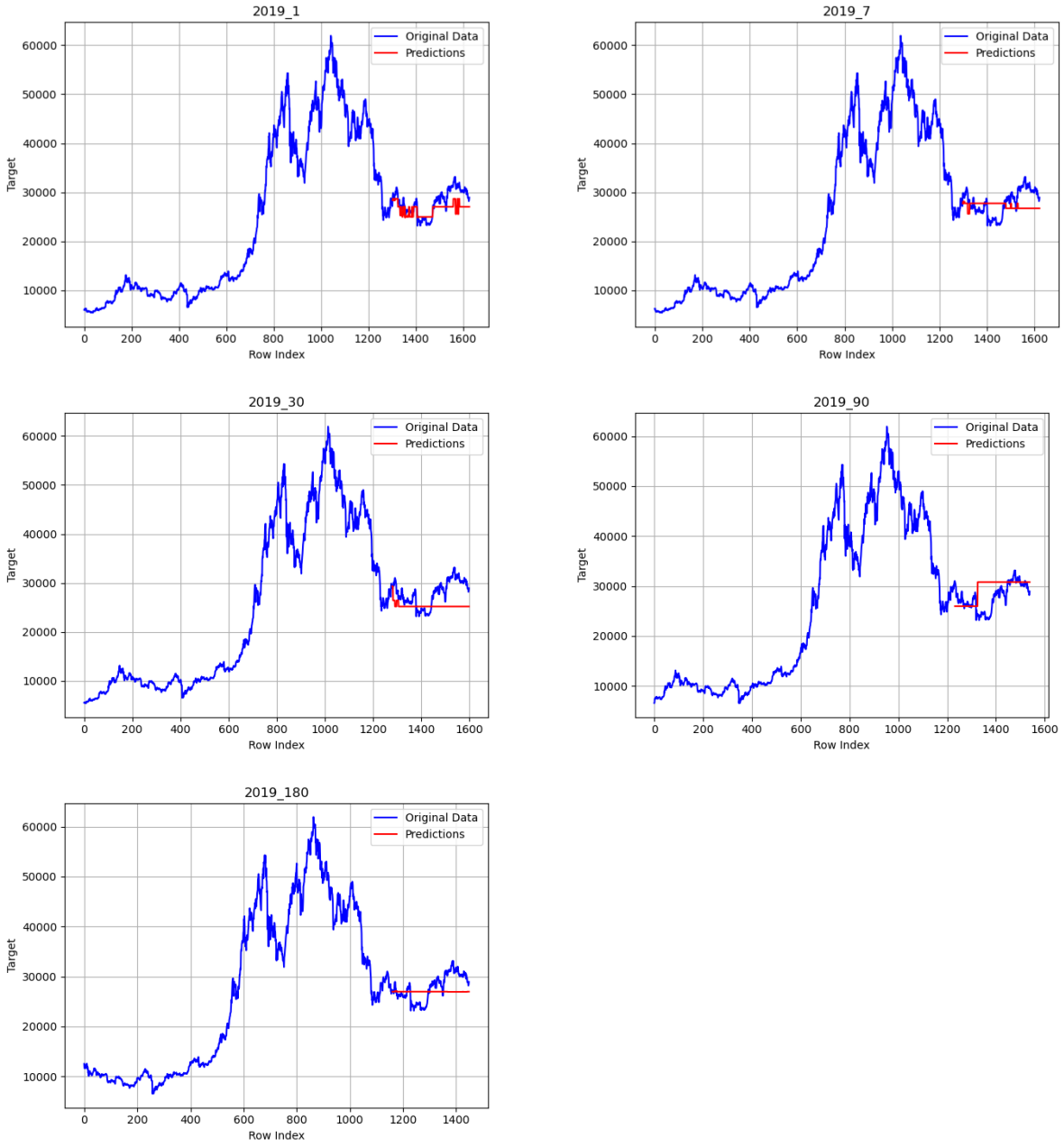

Figure 4.9: Decision Tree Predictions vs Actual Prices for Different Window Sizes in 2019

4.1.2.5 AdaBoost

The best hyperparameters found after fine-tuning the AdaBoost models using a grid search approach are shown in Table 4.6.

Dataset	Learning Rate	Number of Estimators
2017_1d	0.4	200
2017_7d	0.4	200
2017_30d	0.4	200
2017_90d	0.3	150
2017_180d	0.4	100
2019_1d	0.4	200
2019_7d	0.4	200
2019_30d	0.3	200
2019_90d	0.4	200
2019_180d	0.3	200

Table 4.6: AdaBoost Best Parameters

Table 4.5 shows the results for both the 2017 and 2019 datasets:

Dataset	RMSE	MAE	MedAE	MAPE	R-squared
2017_1d	4331.49	3785.33	3768.08	34.89%	0.14
2017_7d	5038.76	4454.52	4432.29	40.85%	-0.19
2017_30d	6429.82	5670.55	5691.26	49.92%	-1.06
2017_90d	11471.58	9426.89	8886.90	56.78%	-6.76
2017_180d	13163.41	12171.75	11668.25	67.11%	-9.08
2019_1d	3302.37	2655.30	2234.01	7.05%	0.44
2019_7d	4161.30	3453.54	3146.23	8.16%	0.21
2019_30d	4173.65	3585.54	3323.88	8.22%	0.24
2019_90d	9157.70	7207.44	5305.40	8.51%	-10.74
2019_180d	8450.25	6849.06	6385.28	8.56%	-4.20

Table 4.7: AdaBoost Metrics

In the case of AdaBoost, the model in 2019 could not make predictions that reflected the actual market movements. Although the metrics appear good, according to Table 4.7, due to how close the predicted line is to the actual values, Figure 4.11 shows that there is no useful information being captured.

For the 2017 period (see Figure 4.10), we observe significant shifts in the predicted prices for the 1-day, 7-day, and 30-day windows. As shown in Table 4.7, the MAPE for the 1-day, 7-day, and 30-day windows are 34.89%, 40.85%, and 49.92%, respectively. However, it seems that it can somewhat predict the direction of the price movements, as they are not entirely off. More specifically, for these three windows, the MAE and MedAE are very close. This indicates that

the differences between the predicted and actual values are somehow symmetric, as shown in the graphical representations in Figure 4.10. For the 90-day and 180-day windows in the 2017 period, the model failed to make any predictions.

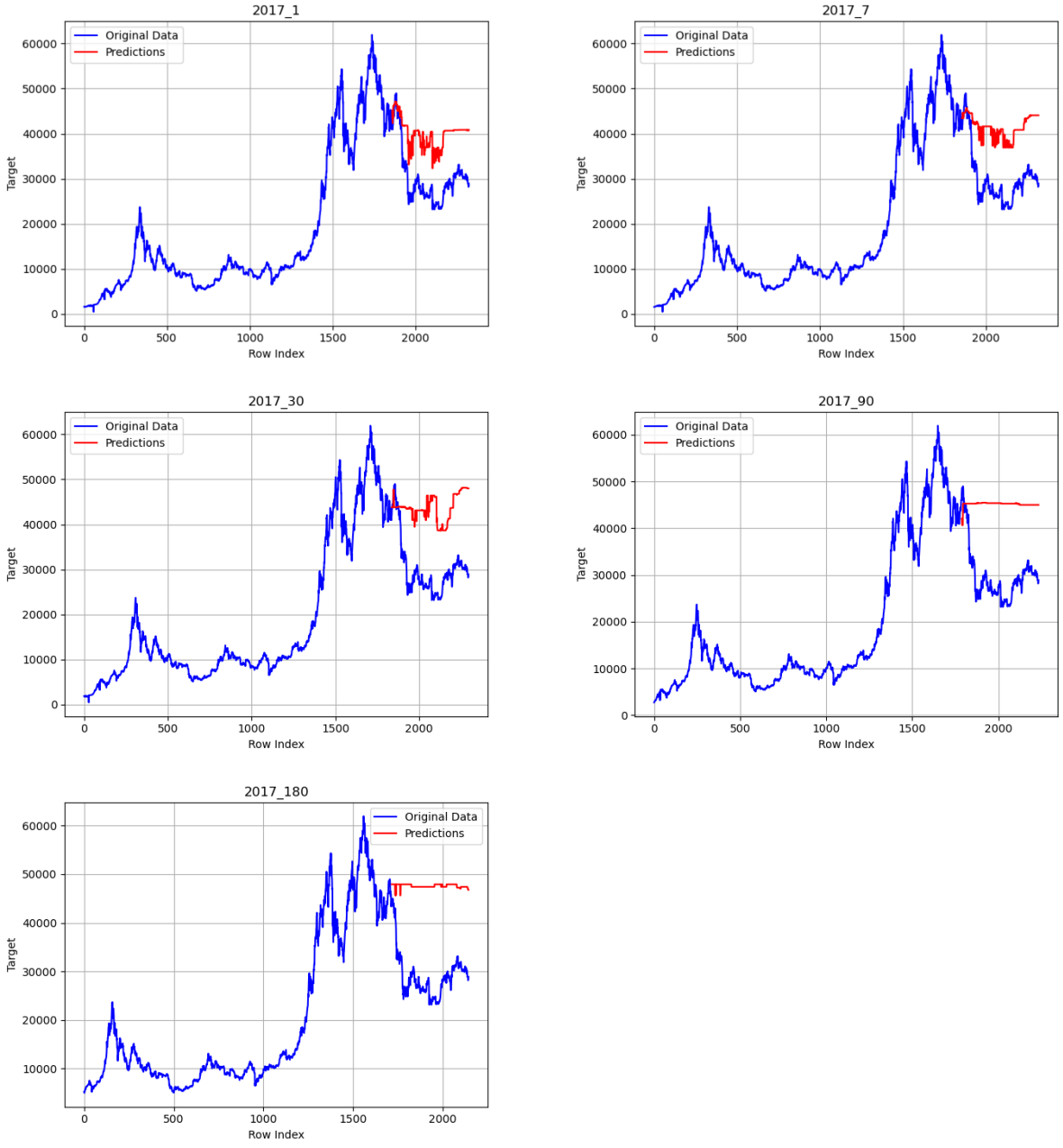

Figure 4.10: AdaBoost Predictions vs Actual Prices for Different Window Sizes in 2017

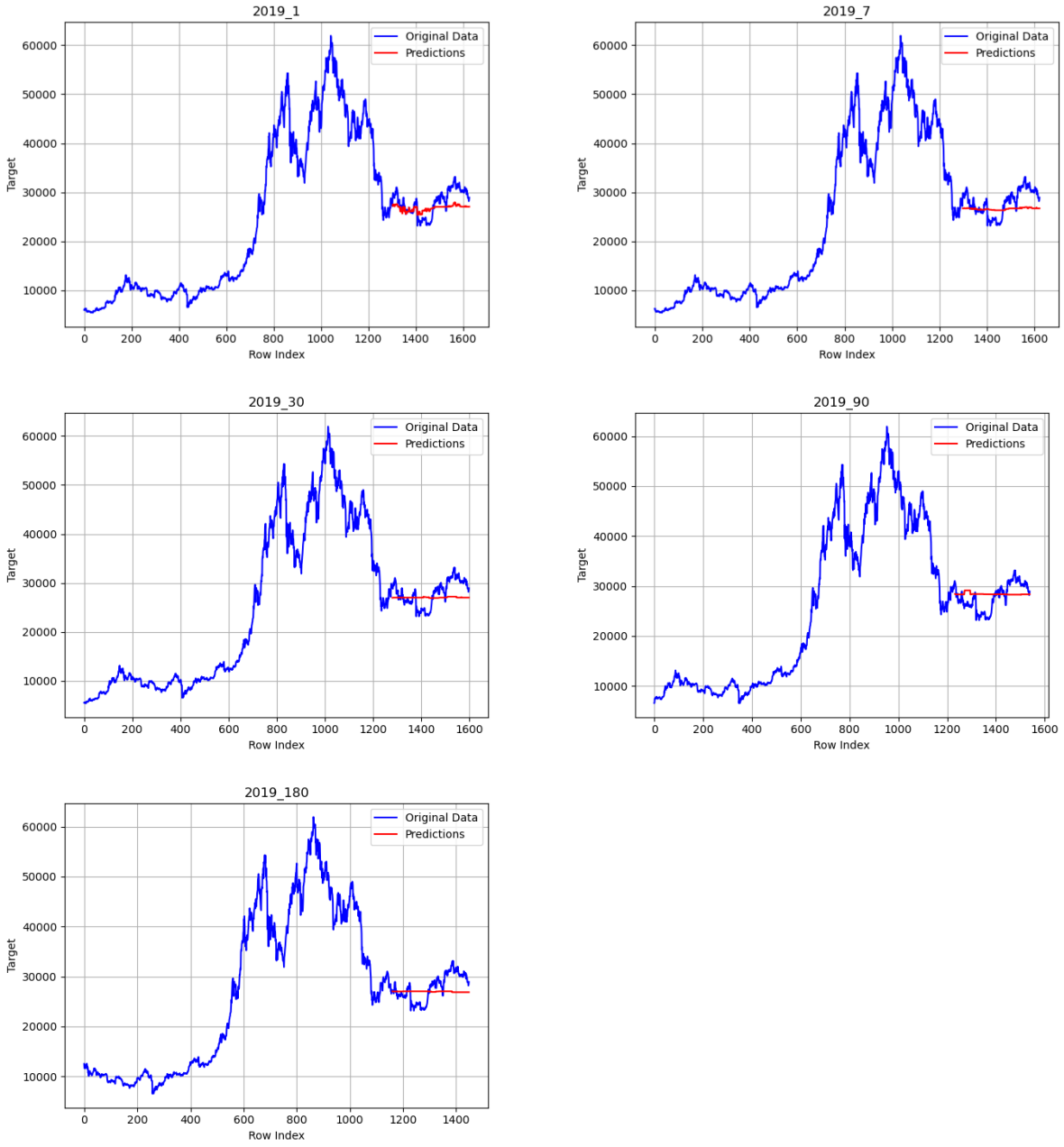

Figure 4.11: AdaBoost Predictions vs Actual Prices for Different Window Sizes in 2019

4.1.2.6 XGBoost

The best hyperparameters found after fine-tuning the XGBoost models using a grid search approach are shown in Table 4.8.

Dataset	colsample_bytree	gamma	lambda	learning_rate	max_depth	min_child_weight
2017_1d	0.5	0	0	0.05	6	3
2017_7d	0.5	0	0.5	0.1	6	1
2017_30d	0.5	0	0	0.1	6	3
2017_90d	0.5	0	0	0.05	6	3
2017_180d	0.5	0	0	0.1	6	1
2019_1d	0.5	0	0	0.05	6	3
2019_7d	0.5	0	0.5	0.1	6	3
2019_30d	0.5	0	10	0.1	6	1
2019_90d	0.5	0	0	0.05	6	3
2019_180d	0.5	0	0	0.05	6	5

Table 4.8: XGBoost Best Parameters

Table 4.9 shows the results for both the 2017 and 2019 datasets:

Dataset	RMSE	MAE	MedAE	MAPE	R-squared
2017_1d	3197.79	2690.59	2384.89	17.05%	0.62
2017_7d	4146.06	3552.08	3395.50	27.77%	0.32
2017_30d	6188.45	5153.15	5132.78	39.93%	-0.85
2017_90d	7814.28	6719.49	6476.93	46.88%	-1.70
2017_180d	8168.04	7085.18	6522.80	68.84%	-4.13
2019_1d	2381.23	1884.87	1501.29	3.92%	0.81
2019_7d	3789.65	3221.36	3076.00	8.64%	0.38
2019_30d	5258.26	4372.67	4459.74	8.82%	-0.41
2019_90d	4260.46	3454.61	2931.59	10.88%	-0.08
2019_180d	4545.46	3741.97	3102.63	10.38%	-3.35

Table 4.9: XGBoost Metrics

Our observations, which can be seen in Figures 4.12 and 4.13 for the periods 2019 and 2017 for the 90-day and 180-day prediction windows, are consistent with the previous tree-based models. However, we observe an exception in the 2019 1-day prediction. The XGBoost algorithm managed to make predictions for the 2019 dataset, but this success is limited to the next day. It seems that the features of the 2019 model helped XGBoost make better predictions compared to the 2017 model with MAPE equal to 3.92% and RMSE of 2381.23, which are significantly smaller than the 17.05% and 3197.79 respectively. Additionally, we can see that the R-squared metric has a value of 0.81, indicating that besides the close predictions, the model captures the variance effectively.

Additionally, for the XGBoost model, we observe similar trends to those of AdaBoost, but the predictions are closer to the actual values for the period 2017 with 1, 7, 30-day prediction window. The errors, as indicated by the percentage, are smaller, with the MAPE dropping to 17.05%, 27.77%, and 39.93%, respectively. Similarly, the RMSE drops to lower values, and we see the same trend in the MAE and MedAE with the AdaBoost algorithm.

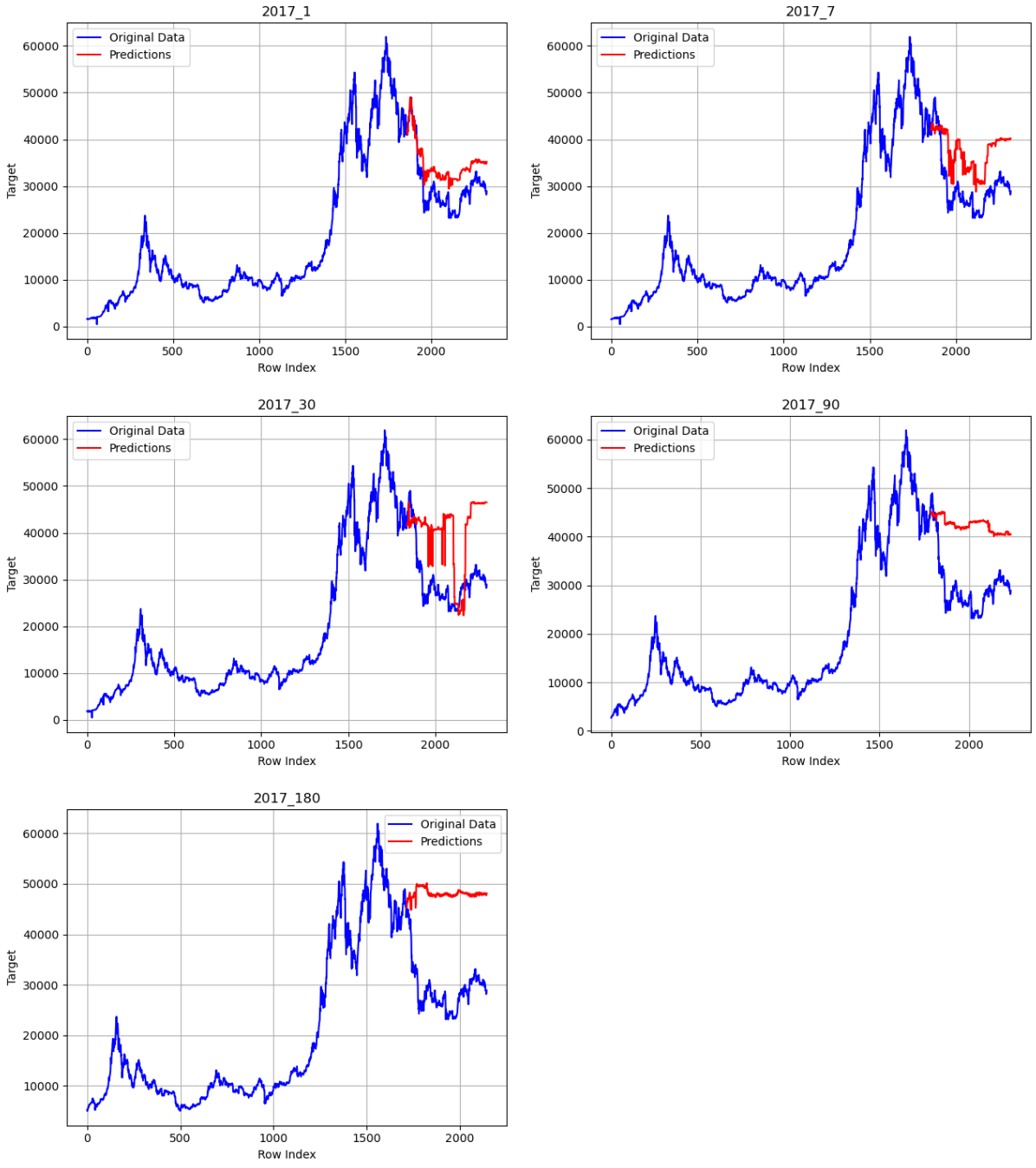

Figure 4.12: XGBoost Predictions vs Actual Prices for Different Window Sizes in 2017

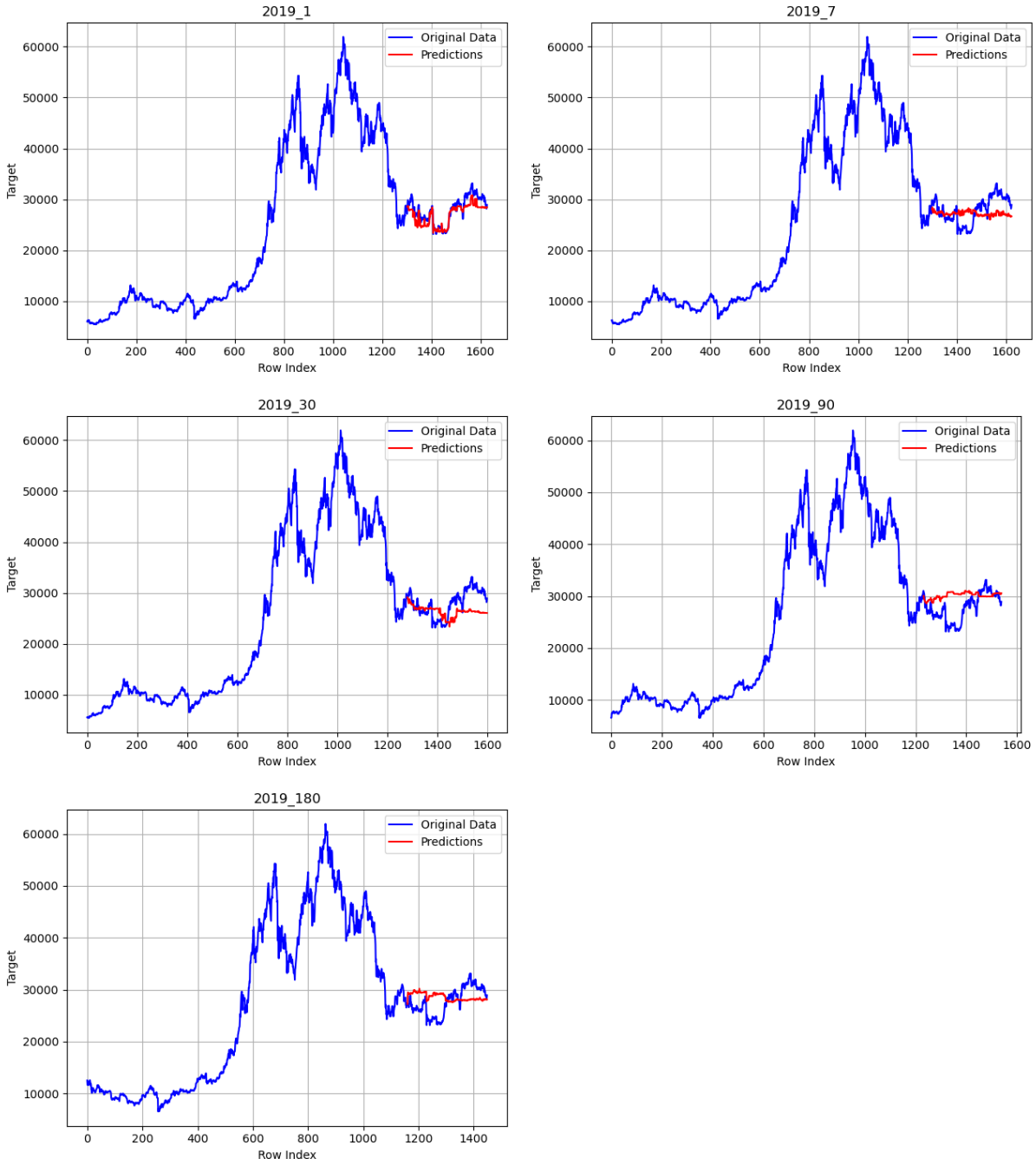

Figure 4.13: XGBoost Predictions vs Actual Prices for Different Window Sizes in 2019

4.1.2.7 Random Forest

The best hyperparameters found after fine-tuning the Random Forest models using a grid search approach are shown in Table 4.10.

Dataset	Max Depth	Max Features	Min Samples Leaf	Min Samples Split	N Estimators
2017_1d	20	sqrt	1	2	150
2017_7d	None	log2	1	2	200
2017_30d	None	log2	1	2	150
2017_90d	30	log2	1	2	200
2017_180d	30	log2	1	2	200
2019_1d	None	sqrt	1	2	150
2019_7d	30	log2	1	2	150
2019_30d	20	log2	1	2	200
2019_90d	None	sqrt	1	2	200
2019_180d	30	log2	1	2	200

Table 4.10: Random Forest Best Parameters

Table 4.11 shows the results for both the 2017 and 2019 datasets:

Dataset	RMSE	MAE	MedAE	MAPE	R-squared
2017_1d	2486.84	1960.08	1529.91	6.56%	0.76
2017_7d	3848.65	3234.33	2900.44	20.89%	0.39
2017_30d	5681.50	4830.81	4625.46	38.75%	-0.59
2017_90d	8791.51	7755.20	7800.03	58.38%	-3.18
2017_180d	10017.37	8883.32	8773.66	62.43%	-4.61
2019_1d	1669.96	1350.90	1240.64	4.98%	0.82
2019_7d	3172.29	2600.86	2339.12	7.34%	0.50
2019_30d	3733.43	3042.89	2740.65	8.07%	0.29
2019_90d	5229.19	4205.95	3449.74	11.62%	-0.35
2019_180d	4484.20	3540.76	2960.00	9.06%	0.16

Table 4.11: Random Forest Metrics

For the 1-day prediction for the 2017 dataset, the model achieves a relatively low MAPE of 6.56%, indicating decent accuracy with an R-squared value of 0.76. However, as the prediction window extends to 7 days and beyond, the errors increase significantly. Due to the nature of how the Random Forest algorithm works, it tends to generalize better predictions, as expected, but fails to maintain accuracy over longer prediction periods. For the 2019 dataset, the model performs even better in terms of metrics, with a lower RMSE (1669.96), MAE (1350.90), and MedAE (1240.64). The MAPE of 4.98% indicates higher accuracy compared to the 2017 dataset. Additionally, the R-squared value of 0.82 suggests that the model explains 82% of the variance in the target variable.

Despite the better metrics for the 2019 dataset, the plot for 2017 seems to show better alignment between the original data and the predictions. The differences can be attributed to variations in data characteristics and distribution between the datasets.

In general, tree-based models rely on the importance of individual features and their interactions to make predictions. Over longer periods, the relevance of certain features may change, and their interactions can become more complex. For that reason, the models might not effectively capture these evolving dynamics with the given dataset.

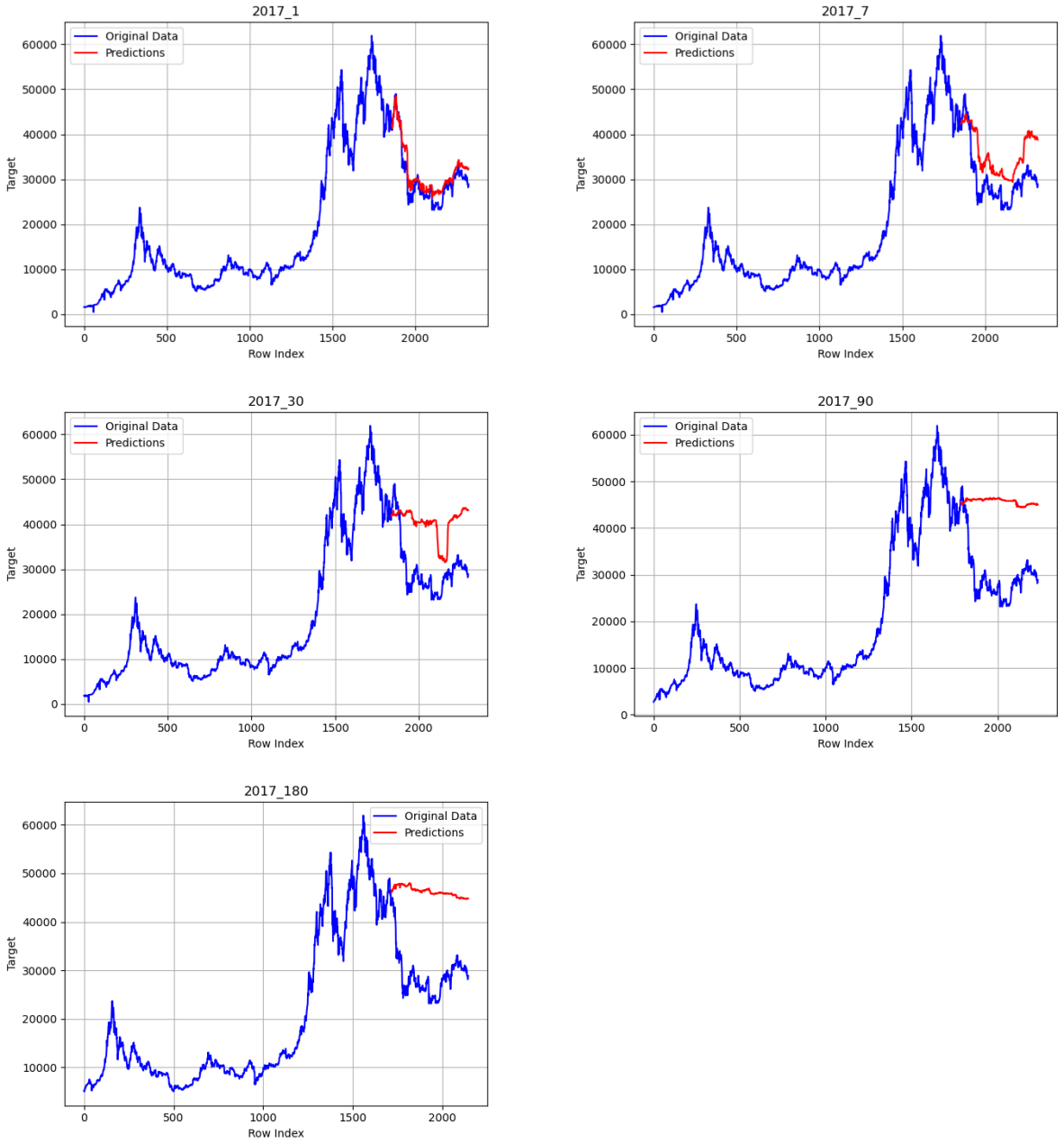

Figure 4.14: Random Forest Predictions vs Actual Prices for Different Window Sizes in 2017

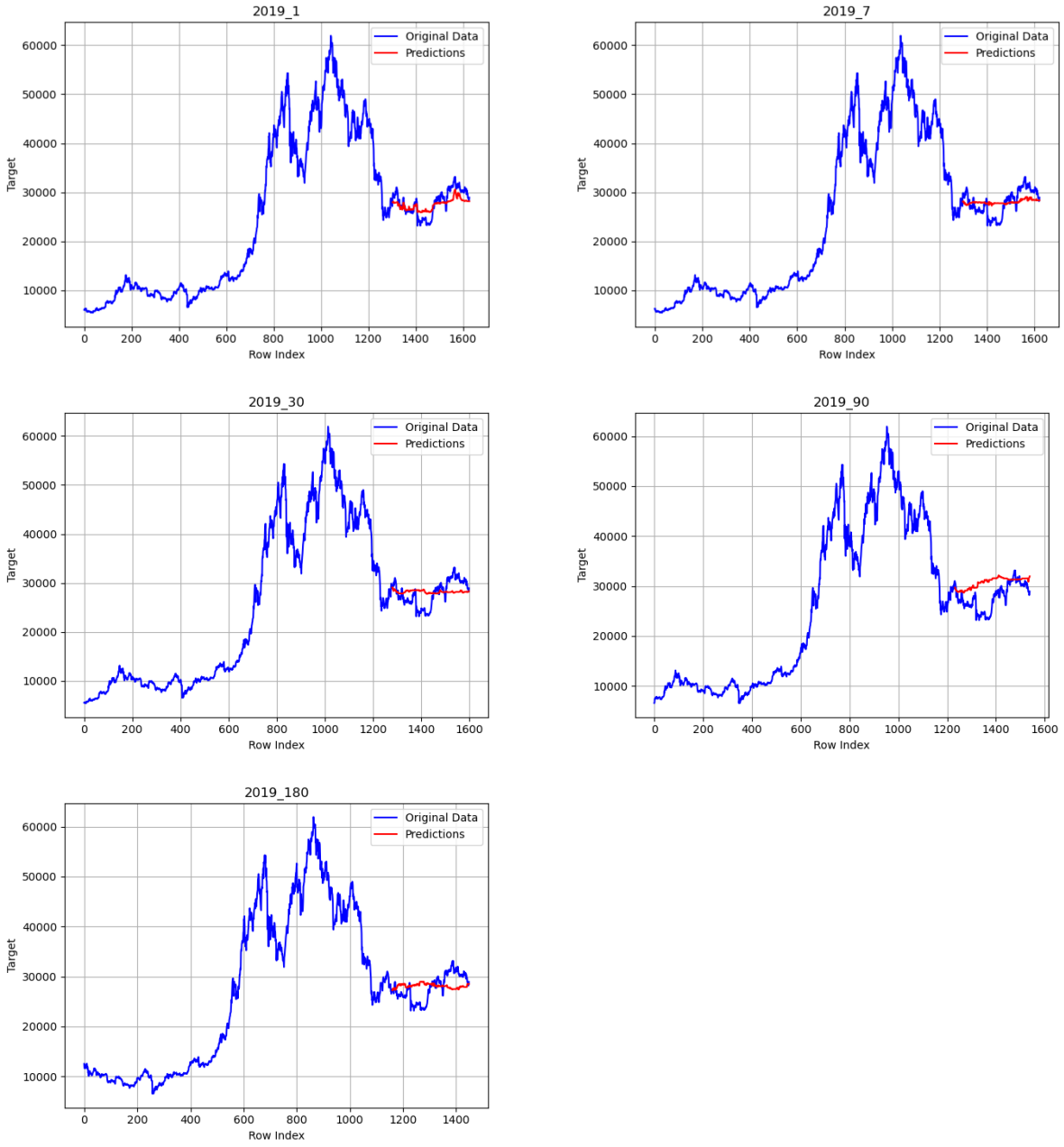

Figure 4.15: Random Forest Predictions vs Actual Prices for Different Window Sizes in 2019

4.2 Deep Learning Algorithms

4.2.1 Methodology

For the deep learning algorithms, we employed a methodology similar to that used for the machine learning models, with some key adaptations. Unlike the machine learning approach, we did not use k-fold cross-validation. Instead, we focused on capturing time series dependencies in the data, ensuring the test set comprised only the most recent period. This approach allowed us to maintain the temporal sequence of the data, which is crucial for time series analysis. The dataset was split into three parts: 70% for training, 15% for validation, and 15% for testing. However, for the LSTM model, the split was adjusted to 70% for training, 10% for validation, and 20% for testing, as this yielded better results. This split allowed the deep learning models to learn from a substantial portion of the data, fine-tune parameters using the validation set, and provide an unbiased evaluation on the test set. The evaluation metrics used were the same as previous.

Also to enhance the training process, we employed the EarlyStopping callback of tensorflow keras's library of Python. This callback stops training once the model's performance on the validation set stops improving, specifically when the validation loss has not decreased for a specific number of consecutive epochs. This technique helps prevent overfitting by stopping the training at the optimal number of epochs, before the model starts to overfit the training data.

4.2.2 Results

4.2.2.1 Convolutional Neural Network

The architecture of the CNN used for the prediction is shown in Table 4.12.

Table 4.12: CNN Model Architecture

Layer (type)	**Output Shape**
Conv1D (filters=32, kernel_size=6, activation='relu')	(None, *num_features* - 5, 32)
MaxPooling1D (pool_size=2)	(None, (*num_features* - 5)/2, 32)
Flatten	(None, 32 * ((*num_features* - 5)/2))
For Windows 30, 90, 180	
Dense (units=50)	(None, 50)
Dense (units=1)	(None, 1)
For Windows 1, 7	
Dense (units=1)	(None, 1)

Optimizer used: Adam optimizer with learning rate equals to 0.001. **Batch size:** 16.

Table 4.13 shows the results for both the 2017 and 2019 datasets:

Table 4.13: CNN Model Metrics

Dataset	RMSE	MAE	MedAE	MAPE	R-squared
2017_1d	1489.94	1234.75	1073.70	4.27%	0.64
2017_7d	2037.62	1630.03	1305.03	5.58%	0.33
2017_30d	5572.49	4859.92	4651.55	14.42%	-3.97
2017_90d	9153.81	8016.79	6554.59	24.02%	-12.29
2017_180d	9041.04	7530.82	6385.66	26.11%	-11.50
2019_1d	7863.47	7511.19	7999.11	20.72%	-6.88
2019_7d	9531.66	9332.25	9277.86	24.87%	-10.56
2019_30d	21969.78	21817.68	21606.06	43.75%	-59.92
2019_90d	22960.33	22891.88	23522.86	44.98%	-63.90
2019_180d	11214.77	10026.63	7639.12	25.62%	-13.92

Based on the evaluation of the CNN model for the 2017 period, several conclusions can be drawn. For short-term predictions, specifically the 1-day and 7-day windows in the 2017 period, the CNN model shows relatively good performance with lower RMSE, MAE, and MedAE values. Despite having relatively low MAPE values of 4.27% and 5.58%, indicating that the average percentage error between predicted and actual values is fairly small, the R-squared values remain low at 0.64 and 0.33 respectively. This suggests that while the model is good at predicting the magnitude of changes, it struggles to capture the overall changes and trends in the data. The low R-squared values indicate that a significant portion of the variation in the actual data is not explained by the model.

As the prediction window increases to 30-day, 90-day, and 180-day, the model's accuracy significantly declines. This can be seen in Table 4.15, where RMSE, MAE and MedAE values are higher, indicating a poor fit to the data. Additionally, the high negative values of R-squared indicate that the model fails to capture any trend in the data. The 2019 dataset results also show very high error values compared to the 2017 dataset, which is probably due to the smaller amount of data.

In general, CNNs are well-suited for capturing local patterns, but they struggle with long-term dependencies. In Figure 4.16, we can see that the predictions are better for smaller window size predictions.

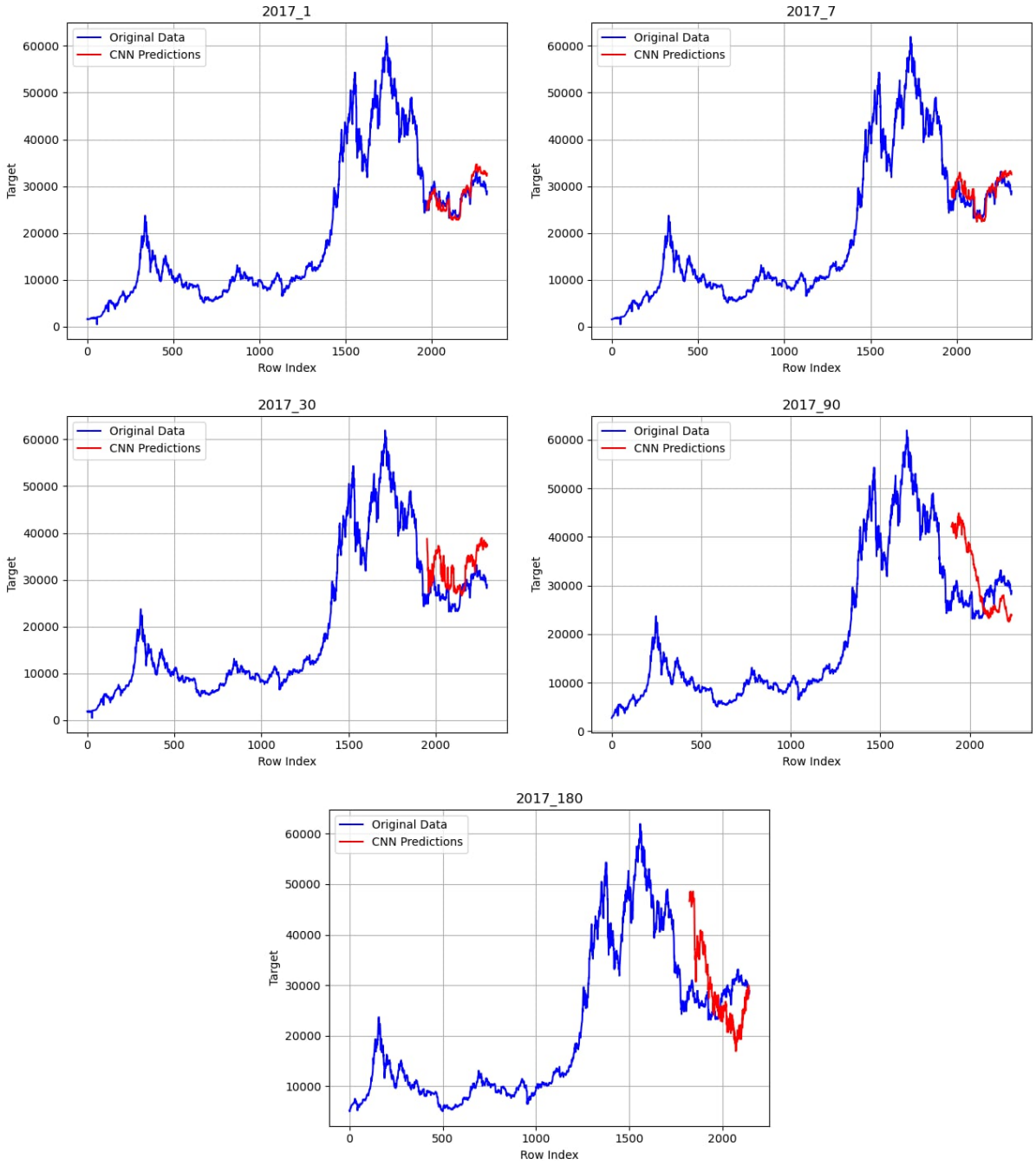

Figure 4.16: CNN Predictions vs Actual Prices for Different Window Sizes in 2017

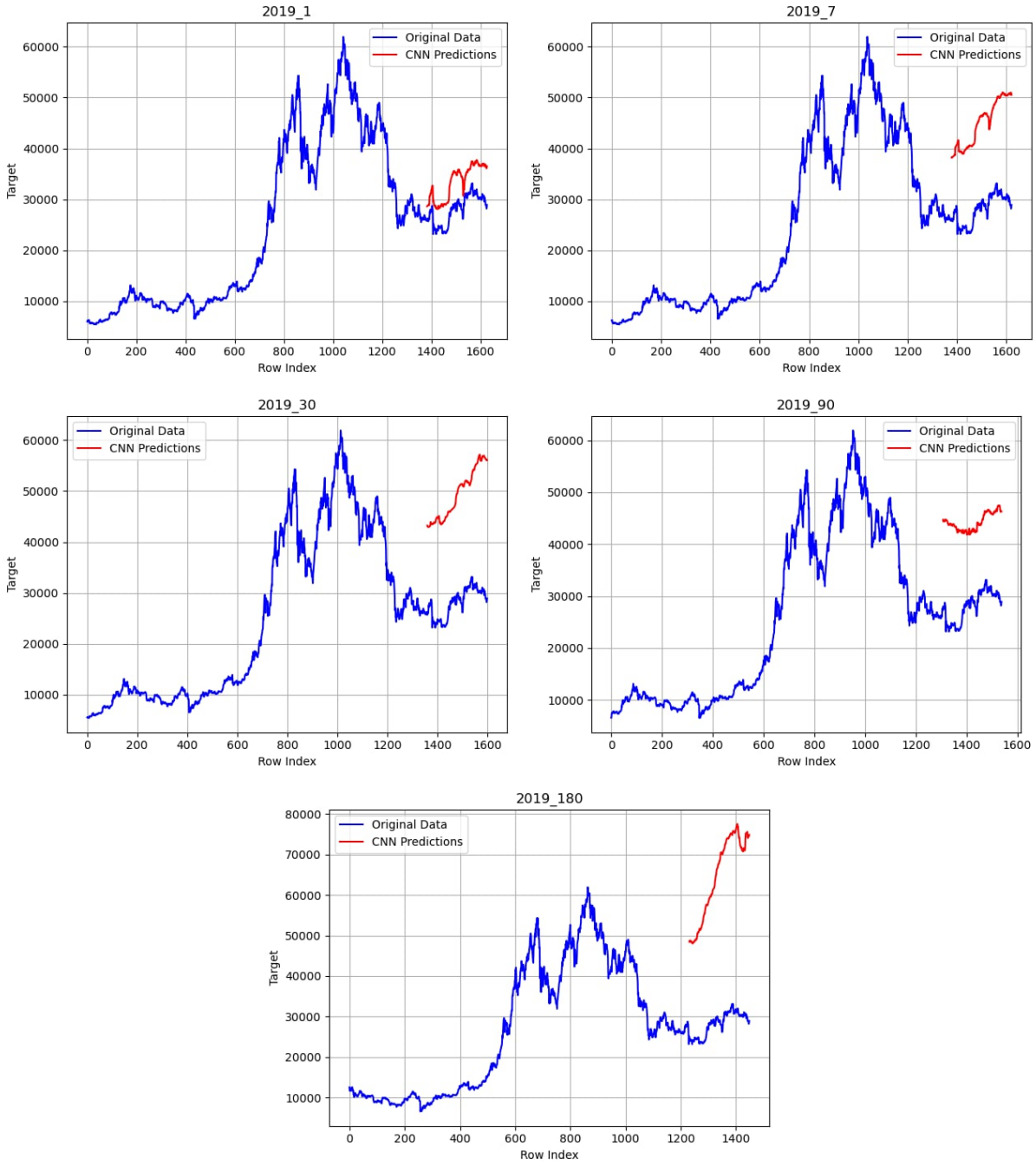

Figure 4.17: CNN Predictions vs Actual Prices for Different Window Sizes in 2019

4.2.2.2 Recurrent Neural Network

The architecture of the RNN used for the prediction is shown in Table 4.14.

Table 4.14: RNN Architecture

Layer (type)
SimpleRNN (100 units)
For Windows 1, 7
Dense (1 unit)
For Windows 30, 90, 180
Dense (100 units, relu)
Dense (1 unit)

Optimizer used: Adam optimizer with learning rate equals to 0.001. **Batch size:** 16.

Table 4.15 shows the results for both the 2017 and 2019 datasets:

Table 4.15: RNN Model Performance Metrics

Dataset	**RMSE**	**MAE**	**MedAE**	**MAPE**	**R-squared**
2017_1d	1853.22	1329.89	818.15	4.66%	0.45
2017_7d	2957.91	2698.26	2533.66	10.97%	-0.41
2017_30d	6613.85	6084.09	6241.99	17.63%	-6.01
2017_90d	4618.85	3748.07	3317.75	11.82%	-2.38
2017_180d	4720.49	4259.07	4518.84	15.04%	-2.41
2019_1d	1220.04	1115.19	1121.33	4.20%	0.81
2019_7d	4585.09	4043.70	3929.21	16.82%	-1.67
2019_30d	5191.41	4252.10	3404.54	13.12%	-2.40
2019_90d	8071.58	7327.55	6261.34	33.87%	-7.02
2019_180d	7743.29	5588.26	3639.36	15.50%	-6.11

As Table 4.15 shows and Figures 4.18, 4.19 illustrate the RNN model shows a reasonable performance for 1-day predictions for both 2017 and 2019 periods, with the best performance being in 2019. More specifically, in the 2019 next day predictions, we observe low values in the errors and a high R-squared value of 0.81.

As the prediction window extends to 7 days, 30 days, 90 days, and 180 days, for both periods, the error metrics (RMSE, MAE, MAPE) increase significantly, and the R-squared values turn negative. This indicates a decline in the model's accuracy and suggests that the model struggles to generalize well over longer prediction periods and to capture the overall trends in the data.

Overall, the 2017 period dataset results in better accuracies compared to the 2019 dataset. This may suggest that the extra data in the 2017 period helps make better long-term predictions despite the differentiated features of the 2019 dataset.

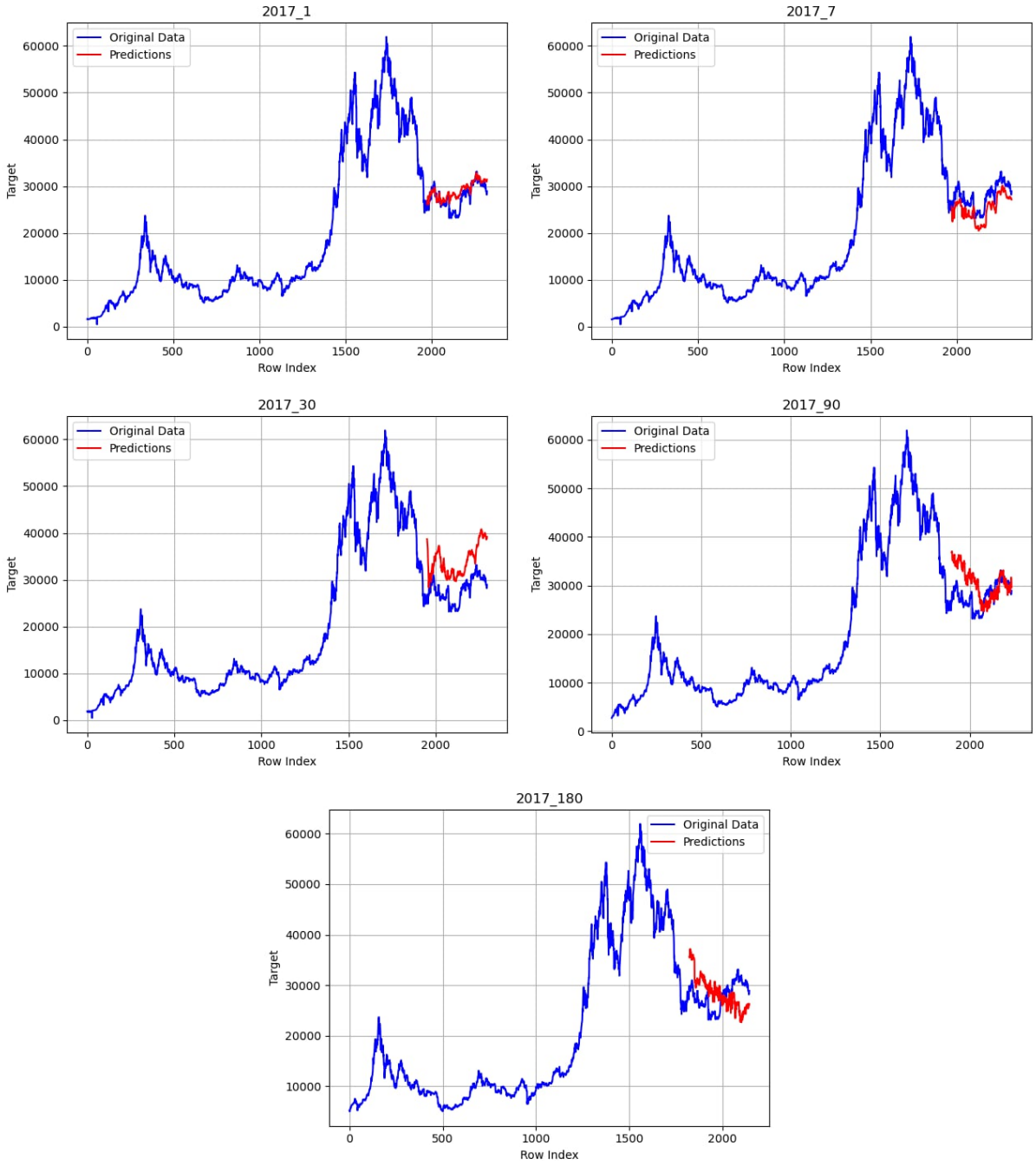

Figure 4.18: RNN Predictions vs Actual Prices for Different Window Sizes in 2017

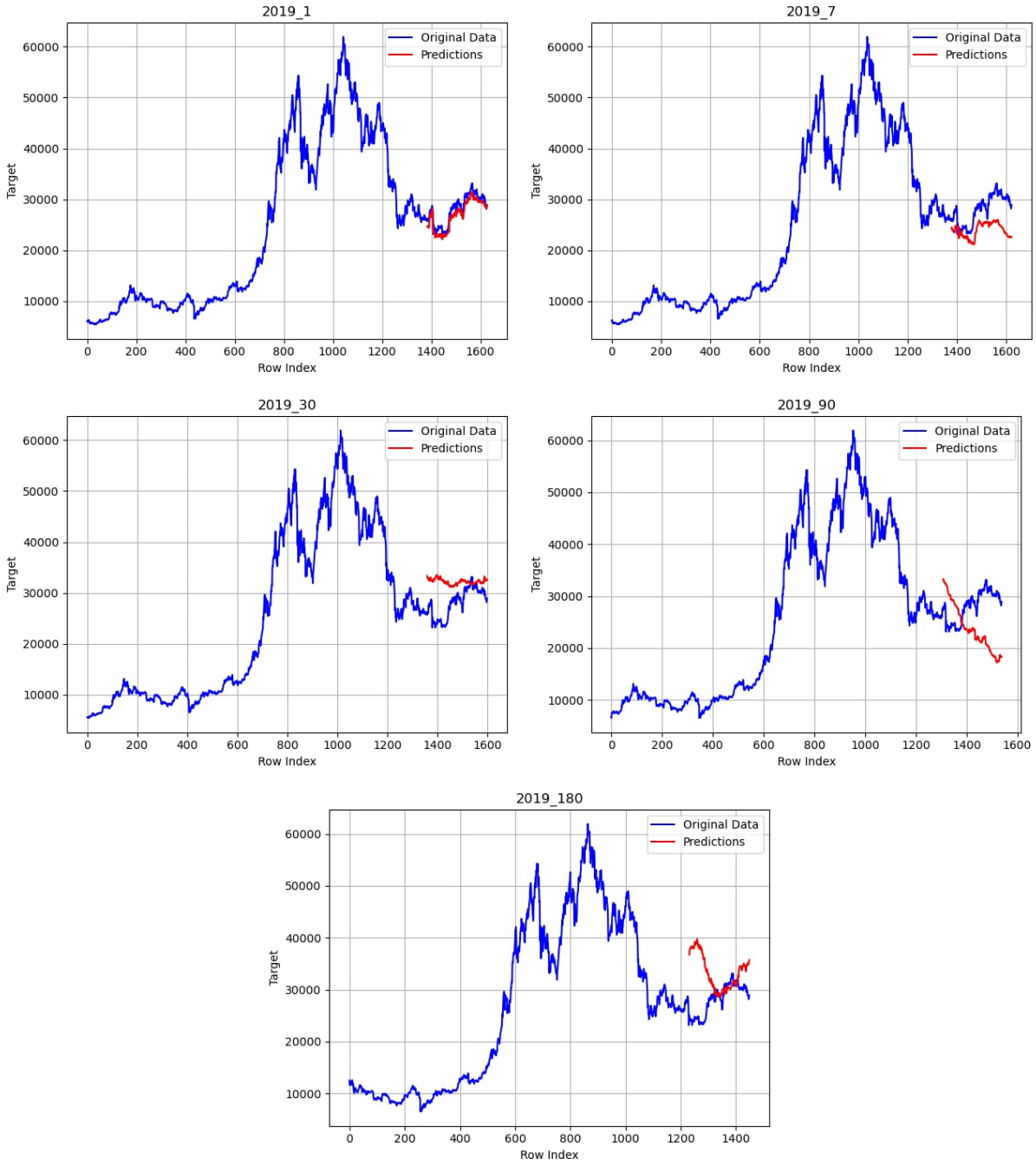

Figure 4.19: RNN Predictions vs Actual Prices for Different Window Sizes in 2019

4.2.2.3 Long Short-term Memory

The architecture of the LSTM used for the prediction is shown in Table 4.16.

Table 4.16: LSTM Model Architecture and Parameters

Dataset	LSTM Neurons	Dense Layers	Optimizer	Batch Size
2017_1d	100	1 Dense (1)	default	16
2017_7d	60	1 Dense (1)	default	16
2017_30d	64	2 Dense (64, 1)	Adam	16
2017_90d	80	2 Dense (64, 1)	Adam	64
2017_180d	80	2 Dense (64, 1)	Adam	64
2019_1d	100	1 Dense (1)	default	16
2019_7d	64	2 Dense (64, 1)	Adam	16
2019_30d	80	2 Dense (64, 1)	default	128
2019_90d	64	2 Dense (64, 1)	Adam	64
2019_180d	80	2 Dense (64, 1)	Adam	128

Table 4.17 shows the results for both the 2017 and 2019 datasets:

Dataset	RMSE	MAE	MedAE	MAPE	R^2
2017_1d	1449.89	1180.62	1024.99	3.99%	0.9431
2017_7d	1752.04	1309.56	1039.41	4.43%	0.9162
2017_30d	8563.63	7723.11	7948.03	20.48%	-1.0601
2017_90d	8155.16	7201.91	6977.08	24.48%	-1.1537
2017_180d	9640.88	8360.15	8352.09	25.81%	-3.8714
2019_1d	742.81	618.11	598.87	2.25%	0.9149
2019_7d	2749.95	2190.83	1826.71	6.96%	-0.1630
2019_30d	3572.97	3066.57	3447.22	9.91%	-0.9436
2019_90d	3964.70	3715.99	4086.89	13.59%	-1.3501
2019_180d	7359.63	6515.08	6940.07	18.76%	-6.8059

Table 4.17: LSTM Evaluation Metrics

First of all, we can see a very high accuracy of the R-squared score in the 2017 prediction window for the next day (0.9431) and after 7 days (0.9162). This means that the model not only minimizes the magnitude error between the predicted and actual values, but also captures the trend of the data, making correct predictions for both increasing and decreasing prices (see Figure 4.19). Additionally, the MAPE is very low, 3.98% for 1 day and 4.39% for 7 days, which means that on average, the predictions will be within 3-4% of the actual price. However, after the 7-day window, the RMSE increases to 8563.63, indicating that the model struggles to find the complex relationships in the data over longer periods.

For the 2019 period, despite having fewer rows, the model produces very good accuracies for the next day, even better than the 2017 1-day prediction, with an RMSE of 742.81, MAE of 618.11,

MedAE of 598.87, and MAPE of 2.25%. Despite the lower values of these metrics compared to the 2017 period, the R-squared value is slightly lower, which means that the predictions for the dataset of the 2017 period capture the variance better.

Accuracies for larger windows in 2017 are not good, with high errors, but it seems that the model still captures that the price will move down.

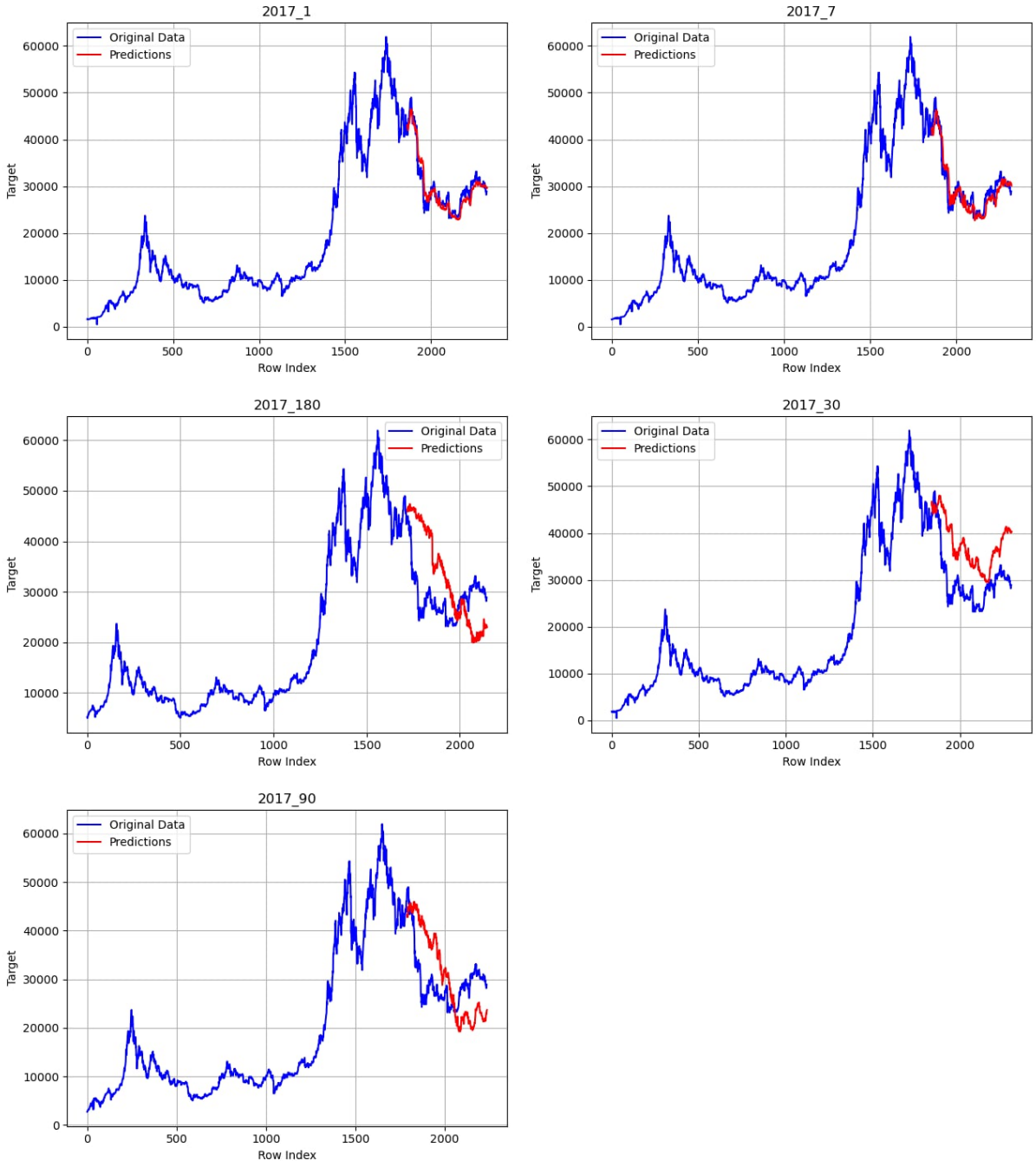

Figure 4.20: LSTM Predictions vs Actual Prices for Different Window Sizes in 2017

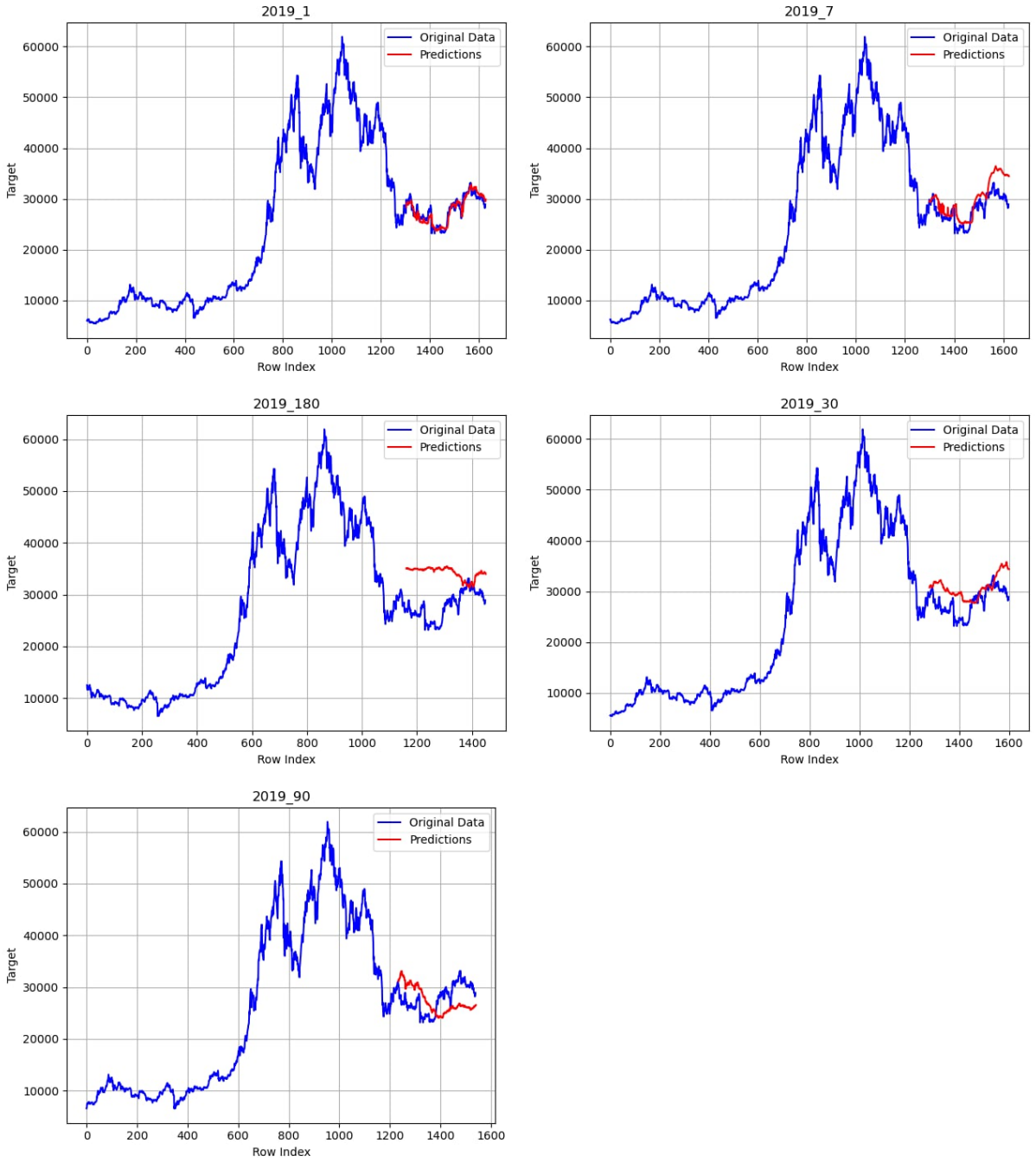

Figure 4.21: LSTM Predictions vs Actual Prices for Different Window Sizes in 2019

4.3 Model Comparison

In this section, we will compare the algorithms. We will start with the machine learning algorithms and focus on the 2017 dataset for the next day, 7 days, and 30 days. This is because, for the 2019 dataset, most models could not make accurate predictions, and for further predictions in 2017 (90 days and 180 days), the models also did not perform well. The comparison will be made using two metrics: RMSE for mean comparison and MedAE for median comparison.

We can see in Figures 4.22, 4.23 that both metrics agree on which model predicts better than the others. The best models seem to be the tree-based models, with Random Forest being the best, followed by XGBoost and then AdaBoost. Random Forest, in particular, uses multiple trees to make the final prediction, which improves accuracy and robustness. Also, KNN appears to perform better than Decision Trees. These observations are consistent for predictions over all days. Additionally, we can see that Linear Regression produces better results for short-term predictions, but for the next 30 days, its performance significantly decreases, resulting in worse results compared to SVM.

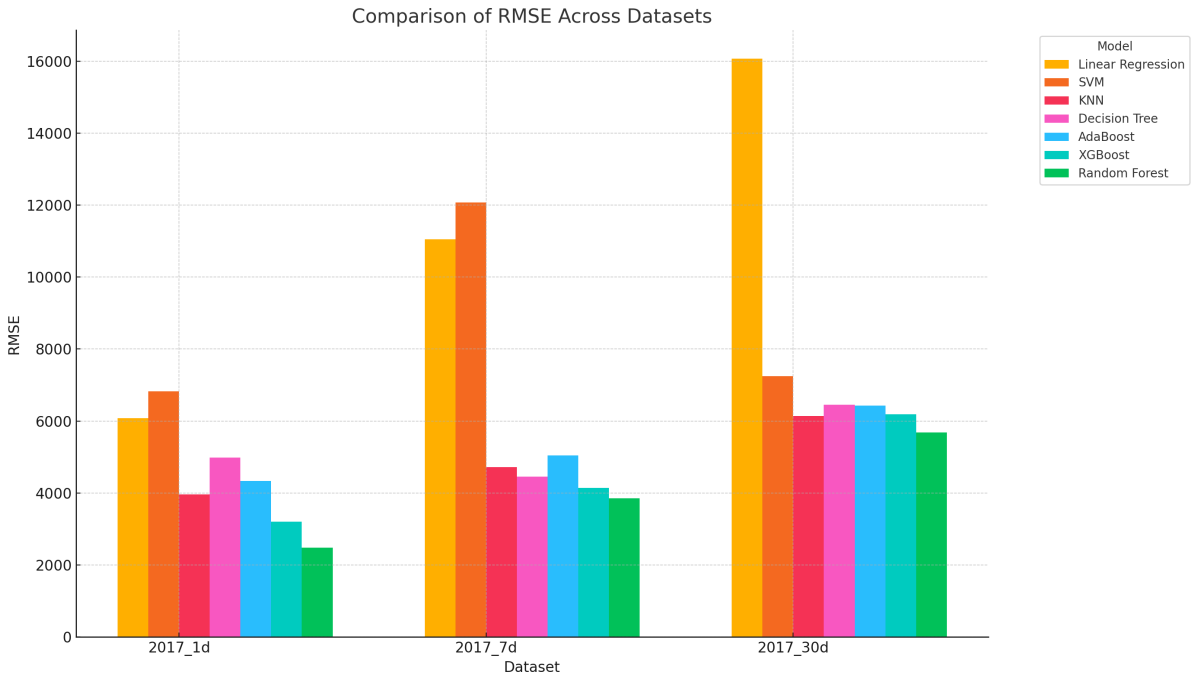

Figure 4.22: Comparison of RMSE Across ML Models

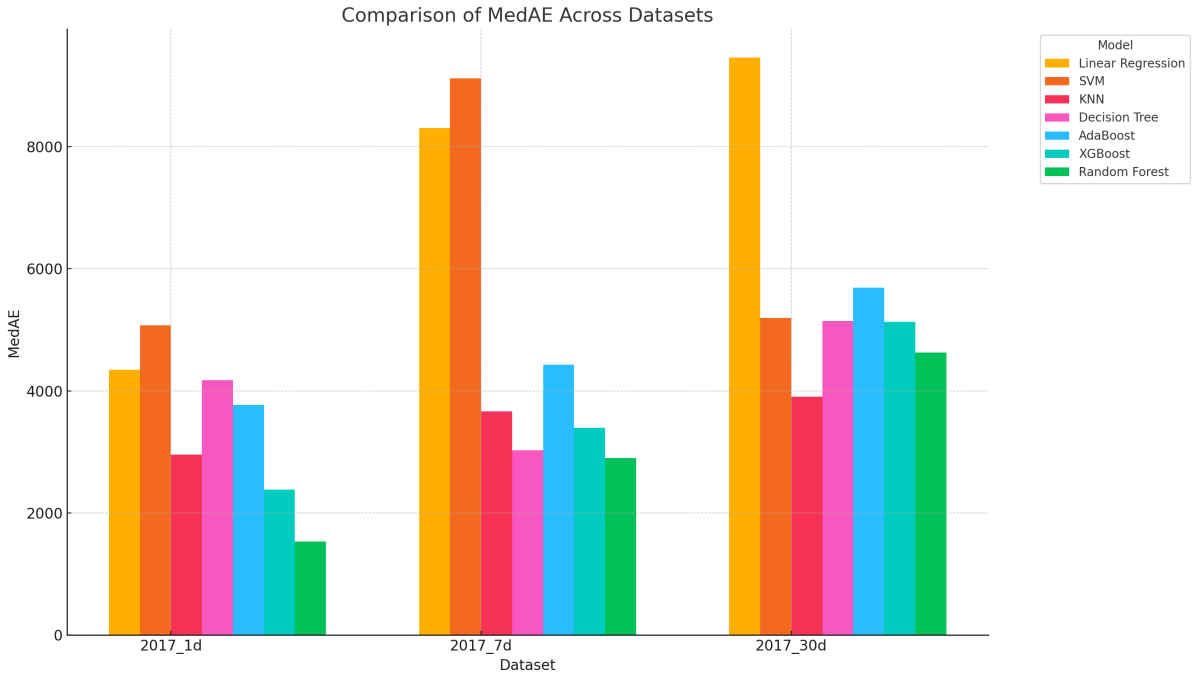

Figure 4.23: Comparison of MedAE Across ML Models

In Figures 4.24 and 4.25, we see the comparison of the three DL models. For the 2017 period, the next day and the 7 days according to the RMSE metric, the LSTM had better performance across all the models. However, for the next day, the RNN's MedAE was lower compared to the LSTM. This suggests that RNN's predictions, while possibly having some large errors (outliers), were generally more consistent with the median of the actual values. For long-term predictions (30, 90, 180 days) for both metrics, we cannot see a clear winner because it varies for different window sizes, and in general, the models couldn't make good predictions for the long term.

For the 2019 dataset, which had more features to analyze but fewer rows, we see very high error values with the CNN model. Overall, the LSTM model produces better results for the 2019 dataset according to both metrics, except for predictions over 180 days, where the MedAE of the RNN was smaller.

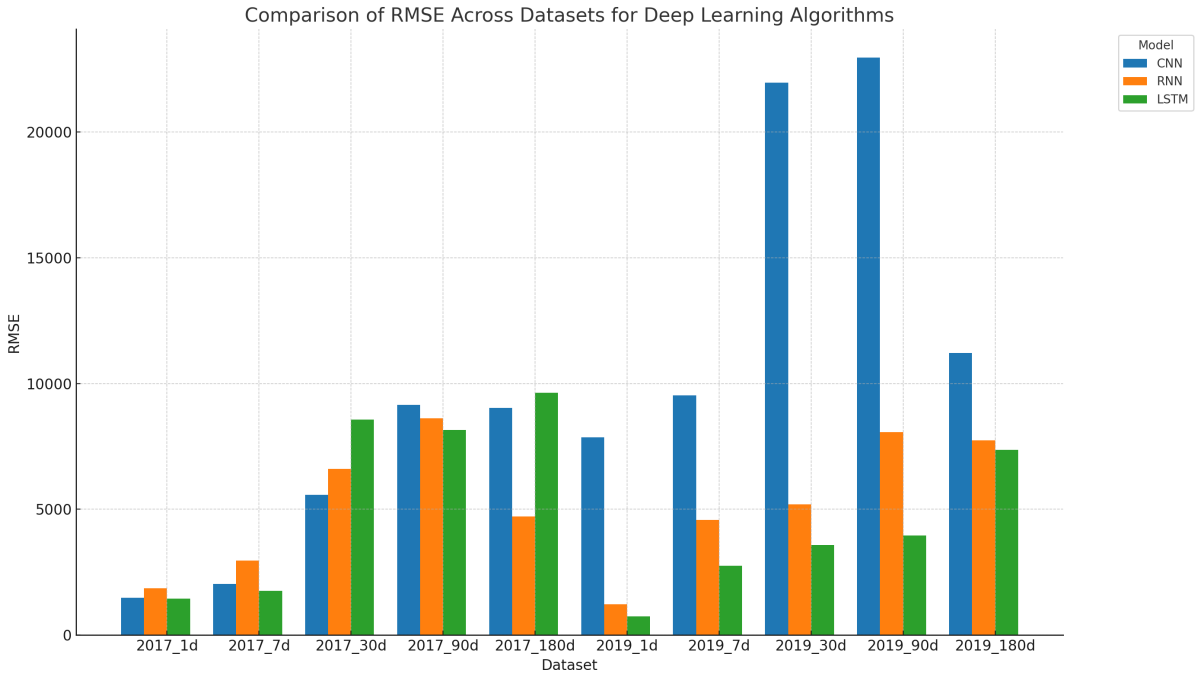

Figure 4.24: Comparison of RMSE Across DL Models

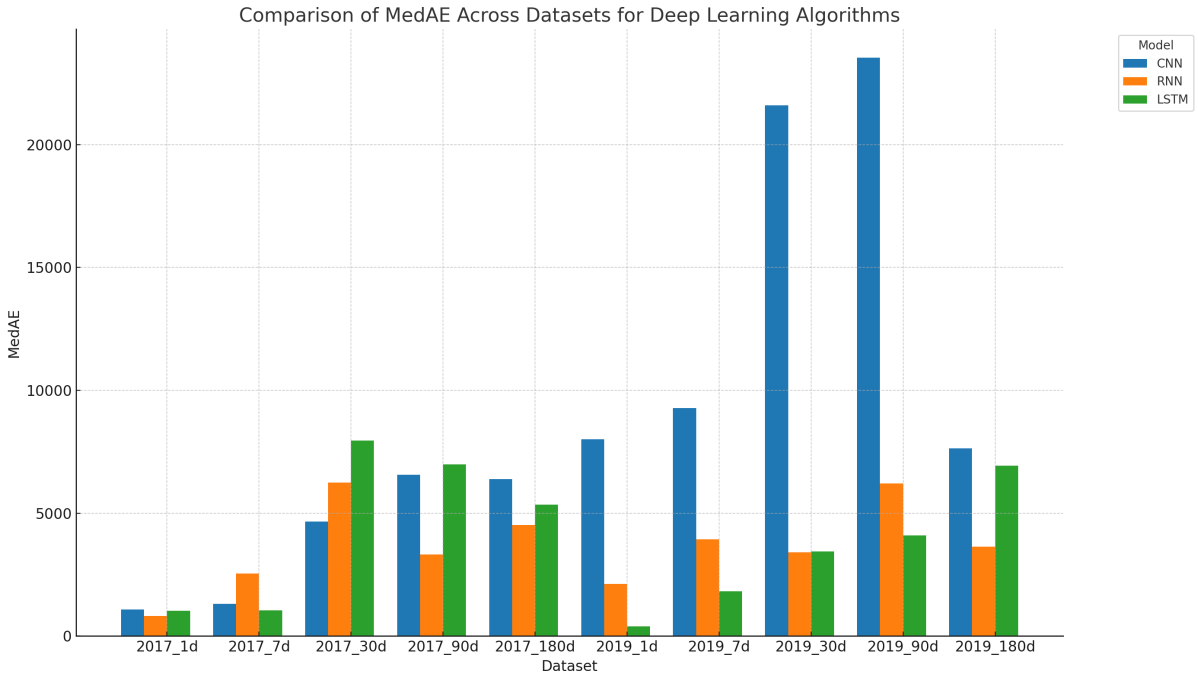

Figure 4.25: Comparison of MedAE Across DL Models

5 Discussion

Contents

5.1 Summary and Final Words 66
5.2 Future work 67

5.1 Summary and Final Words

In summary, we observe that there are indeed indicators that can help identify the price of the Crypto 100 index. As a general comment, we notice that the models struggled to find patterns and relationships when predicting prices far into the future, especially for the 30-day, 90-day, and 180-day windows. This indicates that the models likely struggle to capture very long dependencies. It may also suggest that patterns or dependencies for such distant predictions cannot be found in the data because cryptocurrency prices change too much over that time. Many events can occur during this period that can influence the price. Additionally, we see that the fewer rows in the 2019 period data impacted the accuracy of most models, as they could not make valuable predictions beyond the next day. However, for the next day, most models could make better predictions compared to the 2017 next-day predictions. This suggests that the different features in the 2019 dataset were important in predicting our target.

It is also very important to see the results graphically and not only rely on metrics when dealing with this type of data. For example, in the 2019 dataset, while the metrics of most ML algorithms may show low error numbers, the graphs reveal that the predictions essentially follow a median line, failing to capture actual market movements.

Despite the above, the LSTM model made very good predictions for the next day in the 2019 dataset, with the lowest error values: RMSE of 742.81, MAE of 618.10, MAPE of 2.23%, and an R-squared value of 0.91. Similarly, high accuracies were achieved in the 2017 period for next-day and 7-day predictions. The 7-day predictions are good enough to assist investors in making informed investment decisions, with metrics showing a RMSE of 1752.04, MAE of 1309.56, MedAE of 1039.41, MAPE of 4.43%, and an R-squared value of 0.9162.

Among the ML algorithms, tree-based models generally performed better, with the XGBoost algorithm achieving the best performance. More specifically, the XGBoost model had an RMSE of 2381.23, MAE of 1884.87, MedAE of 1501.29, MAPE of 3.92% and R-squared of 0.81 for the 2019 period next-day prediction.

Overall, it seems that neural networks were better suited for this task.

5.2 Future work

The behavior of cryptocurrencies is influenced by numerous indicators and metrics. Recently, new metrics have shown promise in predicting cryptocurrency prices. These include Cointime Economics metrics such as: Total Coinblocks over time, Liveliness, Vaultedness, Active Supply and Vaulted Supply [49]. Incorporating these metrics could enhance the prediction accuracy of cryptocurrency prices. Unfortunately, during this thesis, we were unable to collect these metrics as we failed to find free sources.

In this thesis, we attempted to predict the Crypto100 index prices for the next day, 7 days, 30 days, 90 days, and 180 days. We found that models struggled with longer-term predictions (30, 90, 180 days). Also, accurate predictions for small future windows are crucial, as even a €200 difference can be significant in cryptocurrency markets, unlike other contexts. Future research could focus on predicting price direction (up or down), which still provides valuable insights, especially for long-term forecasts.

Additionaly, based on the findings of this thesis, it is evident that neural networks were more effective in predicting prices and capturing the general movements of the cryptocurrency market. There are several advanced neural network models specifically designed for time series data that could be explored further. For instance:

- Gated Recurrent Units (GRU) [50]: These models can efficiently capture dependencies in sequential data, making them well-suited for predicting cryptocurrency prices.
- Hybrid Models: Combining models such as CNNs and LSTMs has been shown to enhance predictive performance in various studies. Exploring such hybrid models could yield significant improvements.

Integrating new metrics and leveraging advanced neural network models could significantly improve the accuracy and robustness of cryptocurrency price predictions, providing valuable insights for traders and investors.

BIBLIOGRAPHY

[1] "What is bitcoin." [Online]. Available: https://bitcoin.org/en/

[2] N. Reiff, "Why is bitcoin volatile?" *Investopedia*, January 2024. [Online]. Available: https://www.investopedia.com/articles/investing/052014/why-bitcoins-value-so-volatile.asp

[3] "Machine learning: What it is, tutorial, definition, types." [Online]. Available: https://www.javatpoint.com/machine-learning

[4] "What is cix100? full step guide." 2020. [Online]. Available: https://medium.com/@CryptoIndex/what-is-cix100-full-step-guide-179d7618df26

[5] A. A. Awan, "An introduction to shap values and machine learning interpretability," June 2023. [Online]. Available: https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability

[6] D. Maulud and A. Mohsin Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, pp. 140–147, 12 2020. [Online]. Available: https://www.researchgate.net/publication/348111996_A_Review_on_Linear_Regression_Comprehensive_in_Machine_Learning

[7] R. Curry, "Adaboost: Explained!" Medium, 2022. [Online]. Available: https://medium.com/@curryrowan/adaboost-explained-92408a6713da

[8] L. Rokach and O. Maimon, *Decision Trees*, 01 2005, vol. 6, pp. 165–192. [Online]. Available: https://www.researchgate.net/publication/225237661_Decision_Trees

[9] A. Singh. (2024, February) Knn algorithm: Introduction to k-nearest neighbors algorithm for regression. Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

[10] A. Cutler, D. Cutler, and J. Stevens, *Random Forests*, 01 2011, vol. 45, pp. 157–176. [Online]. Available: https://www.researchgate.net/publication/236952762_Random_Forests

[11] D. Basak, S. Pal, and D. Patranabis, "Support vector regression," *Neural Information Processing – Letters and Reviews*, vol. 11, November 2007. [Online]. Available: https://www.researchgate.net/publication/228537532_Support_Vector_Regression

[12] P. Sonawane. (2023, December) Xgboost — how does this work. Medium. [Online]. Available: https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb

[13] S. Shipra, “What is lstm? introduction to long short-term memory,” *Analytics Vidhya*, Jan 04 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[14] K. O’Shea and R. Nash. (2015) An introduction to convolutional neural networks. [Online]. Available: https://doi.org/10.48550/arXiv.1511.08458

[15] S. Das, A. Tariq, T. Santos, S. S. Kantareddy, and I. Banerjee, *Recurrent Neural Networks (RNNs): Architectures, Training Tricks, and Introduction to Influential Research*. Springer US, 2023, pp. 117–138. [Online]. Available: https://doi.org/10.1007/978-1-0716-3195-9_4

[16] B. Becher, “Understanding blockchain technology,” March 29 2024. [Online]. Available: https://bitcoin.org/en/

[17] “What is ethereum?” [Online]. Available: https://ethereum.org/en/what-is-ethereum

[18] “What is litecoin?” [Online]. Available: https://litecoin.org

[19] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system.” [Online]. Available: https://www.bitcoin.org/bitcoin.pdf

[20] S. Nevil, “What is proof of work (pow) in blockchain?” 2024. [Online]. Available: https://www.investopedia.com/terms/p/proof-work.asp

[21] “What is market cap?” [Online]. Available: https://www.coinbase.com/en-gb/learn/crypto-basics/what-is-market-cap

[22] I. Sarker, “Machine learning: Algorithms, real-world applications and research directions,” *SN COMPUT. SCI.*, vol. 2, 2021, applications of Machine Learning. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x

[23] O. Yenigün, “About the importance of data in machine learning,” *Medium*, Sep 2022. [Online]. Available: https://medium.com/@okanyenigun/about-the-importance-of-data-in-machine-learning-ffa66657ee77

[24] E. F. Morales and H. J. Escalante, “Chapter 6 - a brief introduction to supervised, unsupervised, and reinforcement learning,” in *Biosignal Processing and Classification Using Computational Learning and Intelligence*, A. A. Torres-García, C. A. Reyes-García, L. Villaseñor-Pineda, and O. Mendoza-Montoya, Eds. Academic Press, 2022, pp. 111–129. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128201251000178

[25] J. Terra. (2023) Regression vs. classification in machine learning for beginners. [Online]. Available: https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article

[26] B. W. Artem Oppermann. (2023, December) What is deep learning and how does it work? Built In National. [Online]. Available: https://builtin.com/machine-learning/deep-learning

[27] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, December 1943. [Online]. Available: https://doi.org/10.1007/BF02478259

[28] S. Kilicarslan, K. Adem, and M. Celik, "An overview of the activation functions used in deep learning algorithms," *Journal of New Results in Science*, vol. 10, pp. 75–88, 2021. [Online]. Available: https://doi.org/10.54187/jnrs.1011739

[29] D. Saul, "Lstm recurrent neural networks — how to teach a network to remember the past," *Towards Data Science*, Feb 6 2022. [Online]. Available: https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e

[30] [Online]. Available: https://www.upgrad.com/blog/basic-cnn-architecture

[31] X. Li, Y. Wang, S. Basu, K. Kumbier, and B. Yu, "A debiased mdi feature importance measure for random forests," 2019. [Online]. Available: https://arxiv.org/abs/1906.10845

[32] Y. Zohar, "Permutation importance (pi): Explain machine learning predictions," 2024. [Online]. Available: https://www.aporia.com/learn/feature-importance/explain-ml-models-with-permutation-importance

[33] S. Hiregoudar, "Ways to evaluate regression models," aug 2020. [Online]. Available: https://towardsdatascience.com/ways-to-evaluate-regression-models-77a3ff45ba70

[34] [Online]. Available: https://www.coingecko.com/en/api/documentation

[35] [Online]. Available: https://coinmarketcap.com/

[36] [Online]. Available: https://www.investing.com/economic-calendar/cpi-733

[37] [Online]. Available: https://fred.stlouisfed.org/series/FEDFUNDS

[38] [Online]. Available: https://www.cmegroup.com/markets/interest-rates/cme-fedwatch-tool.html

[39] [Online]. Available: https://data.ecb.europa.eu/data/datasets/ICP/ICP.M.U2.N.000000.4.ANR

[40] [Online]. Available: https://www.policyuncertainty.com/global_monthly.html

[41] [Online]. Available: https://production.dataviz.cnn.io/index/fearandgreed/graphdata

[42] [Online]. Available: https://alternative.me/crypto/fear-and-greed-index/

[43] [Online]. Available: https://lunarcrush.com/

[44] [Online]. Available: https://www.augmento.ai/

[45] [Online]. Available: https://coinmetrics.io/

[46] Api reference — scikit-learn 1.4.2 documentation. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html

[47] E. Polydorou, "Github repository for methodology code," 2023. [Online]. Available: https://github.com/epolyd01/Thesis-Machine-Learning-Analysis

[48] S. Pandian. (2023) K-fold cross validation technique and its essentials. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/

[49] Dilution-proof. (2023) Reviewing cointime economics. [Online]. Available: https://dilutionproof.medium.com/reviewing-cointime-economics-d153020d18a3

[50] S. Kostadinov, "Understanding gru networks," 2017. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be