A QUANTITATIVE COMPARISON OF REGRESSION MODELS ON TIMELY EVOLVING DATASETS

(ZAMANLA DEĞIŞEN DATALARDA REGRESYON MODELLERININ NICEL KARŞILAŞTIRMASI)

by

Mithat Sinan Ergen, B.S.

Thesis

Submitted in Partial Fulfillment

of the requirements

for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

JUNE 2019

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Keziban Orman for her unlimited support and guidance with this thesis and my friends for their moral support.

Table of Contents

ACKNOWLEDGEMENTS iii

Table of Contents iv

List of Figures vii

List of Tables xi

ABSTRACT xii

ÖZET xv

1 INTRODUCTION 1

1.1 Literature Review 3

1.2 Objective and Contributions 6

2 METHODOLOGY 8

2.1 Data Set 8

2.1.1 Stock Market Data Set 8

2.1.2 Cryptocurrency Exchange Data Set 10

2.1.3 Weather Data Sets 14

2.2 Regression Algorithms' Comparison Framework 17

2.2.1 CARET Package 21

2.2.2 Major Parameters' Values 21

2.2.3 Model Building Parameters 22

2.2.4 Hypotheses about Data 24

2.3 Regression Models 26

2.3.1 Linear Models 26

2.3.2 Tree-Based Models 29

2.3.3 Neural Network-Based Models 32

2.3.4 Other Algorithms 35

2.4 Evaluation 38

2.5 Comparison Metrics 38

2.5.1 Mean Absolute Error(MAE) 39

2.5.2 Mean Squared Error(MSE) 39

2.5.3 Root Mean Square Error(RMSE) 39

2.5.4 R-Squared Error(R^2) 40

2.5.5 Execution Time 40

3 EXPERIMENTS AND RESULTS 41

3.1 Statistical Analysis Evaluation 42

3.2 Metric Evaluation 45

3.3 Data Comparison and Evaluation 47

3.4 Regular vs Random Training and Testing 49

3.5 Algorithm Comparison and Evaluation 52

3.6 Summary 54

4 CONCLUSION 69

REFERENCES 70

BIOGRAPHICAL SKETCH 76

List of Figures

2.1 End Value of XU100 Index 9

2.2 Visual Representation of the Correlation Matrix of XU100 Data . . . 9

2.3 End Value of Bitcoin Prices 11

2.4 Visual Representation of the Correlation Matrix of Bitcoin Price Data 12

2.5 End Value of Ethereum Prices 13

2.6 Visual Representation of the Correlation Matrix of Ethereum Data . 14

2.7 Temperature of Hungary Weather 15

2.8 Visual Representation of the Correlation Matrix of Weather Hungary Data 17

2.9 Temperature of Madrid Weather 19

2.10 Visual Representation of the Correlation Matrix of Spain Weather Data 20

3.1 R Squared Values of Hungary Weather Data with Train Regular and 0.7 Training Ratio 46

3.2 R Squared Values of Ethereum Data with Train Regular and 0.7 Training Ratio 47

3.3 R Squared Values of Bitcoin Data with Train Regular and 0.7 Training Ratio 48

3.4 R Squared Values of Madrid Weather Data with Train Regular and 0.7 Training Ratio 49

3.5 RMSE Values of Ethereum Data with Same Data Selection and Prediction Process Type and Training Ratio 50

3.6 RMSE Values of Hungary Weather Data with Same Data Selection and Prediction Process Type and Training Ratio 51

3.7 RMSE Values of XU100 Data with Same Data Selection and Prediction Process Type and Training Ratio 52

3.8 RMSE Values of Madrid Weather Data with Same Data Selection and Prediction Process Type and Training Ratio 53

3.9 MAE of Models for Ethereum Data 54

3.10 MAE of Models for Bitcoin Price Data 55

3.11 MAE of Models for Hungary Weather Data 56

3.12 MAE of Models for Madrid Weather Data 57

3.13 MAE of Models for XU100 Data 57

3.14 MAE of Models for Train Regular on Madrid Weather Data 58

3.15 MAE of Models for Train Random on Madrid Weather Data 58

3.16 MAE of Models for Test Regular on Madrid Weather Data 59

3.17 MAE of Models for Test Random on Madrid Weather Data 59

3.18 MAE of Models for Train Regular on Bitcoin Price Data 60

3.19 MAE of Models for Train Random on Bitcoin Price Data 60

3.20 MAE of Models for Test Regular on Bitcoin Price Data 61

3.21 MAE of Models for Test Random on Bitcoin Price Data 61

3.22 MAE of Models for Train Regular on Hungary Weather Data 62

3.23 MAE of Models for Train Random on Hungary Weather Data 62

3.24 MAE of Models for Test Regular on Hungary Weather Data 63

3.25 MAE of Models for Test Random on Hungary Weather Data 63

3.26 MAE of Models for Train Regular on Ethereum Price Data 64

3.27 MAE of Models for Train Random on Ethereum Price Data 64

3.28 MAE of Models for Test Regular on Ethereum Price Data 65

3.29 MAE of Models for Test Random on Ethereum Price Data 65

3.30 TestRegular on Stock Market Data 66

3.31 TestRandom on Stock Market Data 66

3.32 TestRegular on Ethereum Price Data 67

3.33 TestRandom on Ethereum Price Data 67

3.34 Execution Time of Models for Different Data 68

List of Tables

2.1 Correlation Matrix of XU100 Data 10

2.2 Correlation Matrix of Bitcoin Price 12

2.3 Correlation Matrix of Ethereum Data 13

2.4 Correlation Matrix of Spain Weather Data 16

2.5 Correlation Matrix of Spain Weather Data 18

3.1 ANOVA Results 43

3.2 MANOVA Results 44

3.3 Tukey post-hoc Test Results 45

ABSTRACT

Forecasting is considered as an important task in various domains. It has mass effect on several real-world systems. Until recent years, this task has often been realized by statistical approaches. But recently, machine-learning algorithms have been used for such predictions. Although the performance of machine learning algorithms has been proven for tasks like image processing and natural language processing(NLP), one should also prove their accuracy in forecasting and ask the question: "Are machine learning techniques working accurate enough to totally abandon statistical methods in forecasting?"

In order to find an answer to this question, Spyros Makridakis started a competition in 1982 under the name "M-Competitions". In this competition, 111 sub-samples selected from 1001 time series were used to program 15 different forecasting methods with certain variations. The forecasting performances were compared and evaluated. As a result, it was concluded that complex methods did not give better results than simple statistical methods. Similar results were achieved in the second competition held in 1993. Later in 2000, M-3 competition was organized with more researchers and more methods than before. A total of 3003 time series, which were gathered from many different domains including finance and industry fields, were used for evaluation. Five different error metrics were used for the evaluation, and the results of the competition were then announced in (Makridakis & Hibon, 2000). In the M4 competition held in 2018, machine learning forecasting methods were used in addition to the other algorithms from the previous competitions. At the end of this competition, simple machine learning methods showed lower performance than statistical methods. Different researchers have conducted various studies with the data used in M competitions. It is possible to show the work of (Ahmed et al., 2010) as an example. Ahmad and his colleagues made a comprehensive comparison project in 2010. They measured the forecasting performance of the most important machine learning methods and statistical methods on time series data. After this study, they concluded that the data used in the forecasting was highly influential to the results.

We have prepared this thesis with similar motivations. Our aim in this project includes using data from different domains to run various forecasting algorithms. Then evaluate the performance of these algorithms and report the results. However, our research does not include fine parameter tuning for each algorithm to increase their accuracy but a more general performance evaluation. Thus, this research can give different results depending on the change in various parameters. In this context, we have collected data in the stock market, cryptocurrency and weather domains. We put together the daily data of the Istanbul Stock Exchange with the necessary permits from Borsa İstanbul in a single file, simplified the data set and made it ready for use. We created datasets with Bitcoin and Ethereum cryptocurrencies' daily closing values. We used two different weather data which are from Madrid and Hungary. On these data, we ran 17 different forecasting algorithms including statistical and machine learning methods. Then we evaluated their performance according to the error metrics we determined. In terms of error metrics we used mean absolute error (MAE), mean squared error (MSE), root mean square error (RMSE), R-squared error (R2) and execution time of the algorithm.

In our study, other than using data from different domains, we also applied different experimental parameters. The first was to separate the main data at different rates while creating training and test sets. First, we used 15% of all data as training set and the rest as test set. We then increased this ratio gradually and finally set 99% of the main data as training set. With this experiment, we examined the effect of the size of the training / test sets on the outcome. As the second parameter, we created regular or random selection from the main data when creating the training set. In this way, we examined the effect of regular or random selection on the result. Also, we ran the forecasting algorithms both on the training set and the test set and compared the error rates.

While evaluating the results, we investigated the parameters that have effect on the result by statistical analysis method. As a result of this analysis, it was found that the parameters that had the most effect on the result were the test type, the algorithm and the data used, and the ratio of the training data did not have much effect. In the graphical evaluation, we observed that statistical algorithms made

better forecasting than machine learning algorithms on the data we used in this research. Particularly, we saw that LSTM, with some exceptions, had the lowest performance in many cases. But we observed that linear models generally had high performance. When we compared the execution times, we observed that linear models worked much faster than machine learning algorithms.

The largest data we used in this research, which is the Hungarian weather data, contains 96453 lines. The fact that it is not large enough for some algorithms to perform may have affected the result. Therefore, the same experiments can be repeated with very large data to expand the findings in this study. However, the result we obtained from the data we used in this research was that the statistical methods are faster and more efficient than the machine learning methods when forecasting with temporal data.

ÖZET

Günümüzde birçok farklı alanda yapılan çalışmalarda, veriden yola çıkarak tahmin yapmanın önemli bir etkisi vardır. Bu etki günlük hayattaki problemler için de geçerlidir. Uzun bir süre, bu tahminler istatistiksel yaklaşımlar kullanılarak yapılmıştır. Ancak yakın zamanda makine öğrenmesi teknikleri, yaptıkları düşük hata oranlı tahminlerle bu işlemlerde sıklıkla kullanılmaya başlamıştır. Resim ve dil işleme alanlarında makine öğrenmesi metotlarıyla ilgili yapılan birçok çalışma bu tekniklerin performansını kanıtlar niteliktedir. Ancak makine öğrenmesi teknikleri tahmin işlemlerinde istatistiksel metotları terk etmeye yetecek düzeyde çalışmakta mıdır?

Spyros Makridakis bu soruya bir yanıt bulmak adına 1982 yılında "M-Competitions" ismi altında bir yarışma başlatmıştır. Bu yarışmada 1001 zaman serisi içinden seçilmiş 111 alt örnek üzerinde çalışılarak 15 farklı metot, çeşitli varyasyonlarla tahmin yapmak üzere programlandı ve bu metotların tahmin performansları karşılaştırıldı. Sonuç olarak karmaşık metotların basit istatistiksel metotlardan daha iyi sonuç vermediği ortaya çıktı. 1993 yılında ikincisi düzenlenen yarışmada da benzer sonuçlar elde edildi. Daha sonra 2000 yılında M-3 yarışması düzenlendi ve bu yarışmaya önceki yarışmalardan daha fazla araştırmacı, daha fazla metot ile katıldı. Değerlendirme için finans ve endüstri alanlarını da kapsayan birçok farklı alandan derlenen toplamda 3003 zaman serisi kullanıldı. Tahminlerin değerlendirme kriterleri için beş farklı hata metriği kullanıldı ve daha sonra yarışmanın sonuçları (Makridakis & Hibon, 2000) yayınında açıklandı. 2018 yılında yapılan M4 yarışmasında ise makine öğrenmesi tahmin metotları kullanılan algoritmalara eklendi. Bu yarışma sonunda sade makine öğrenmesi metotları istatistiksel metotlara oranla düşük performans gösterdiler. M yarışmalarında kullanılan verilerle farklı araştırmacılar çeşitli çalışmalar yapmışlardır. Bunlara (Ahmed et al., 2010)'in yaptığı çalışmayı örnek gösterebiliriz. Ahmad ve çalışma arkadaşları 2010 yılında geniş kapsamlı bir karşılaştırma projesi yaptılar. Dönemin

en önemli makine öğrenmesi metotlarının ve istatistiksel bazı metotların zaman serisi verisindeki tahmin performansını ölçtüler. Bu çalışma sonrasında kullanılan verinin sonuçları fazlasıyla değiştirdiği sonucuna vardılar.

Biz de benzer motivasyonlarla bu tezi hazırladık. Projedeki amacımız; farklı alanlardan veriler kullanarak farklı tahmin algoritmaları çalıştırmak ve daha sonra bu algoritmaların performanslarını değerlendirmektir. Ancak, burada amacımız tek tek algoritma performanslarını yükseğe çekecek parametre ayarları yapmaksızın genel geçer bir performans değerlendirme deneyi tasarlamaktır. Bu deney çeşitli parametrelerin değişimine göre farklı sonuçlar verebilir. Bu bağlamda borsa, kriptopara ve iklim alanlarında veriler topladık. İstanbul borsasının günlük verilerini Borsa İstanbul'dan gerekli izinleri alarak tek bir dosyada birleştirdik, veri setini sadeleştirdik ve kullanıma hazır hale getirdik. Bitcoin ve Ethereum kriptoparalarının günlük değerlerinin olduğu bir veri seti oluşturduk. İklim verisi olarak da Madrid ve Macaristan olmak üzere iki farklı veri kullandık. Bu veriler üzerinde, içinde istatistiksel ve makine öğrenmesi metotları bulunan 17 farklı tahmin algoritmasını çalıştırdık ve belirlemiş olduğumuz hata metriklerine göre performanslarını değerlendirdik. Hata metrikleri olarak mean absolute error(MAE), mean squared error(MSE), root mean square error(RMSE), r-squared error(R2) ve tahmin algoritmasının çalışma süresini kullandık.

Çalışmamızda farklı veriler kullanmamızın haricinde farklı deney parametreleri de uyguladık. Bunlardan ilki eğitme ve test kümelerini oluştururken ana veriyi farklı oranlarla parçalamaktı. İlk olarak tüm verinin %15'ini eğitme kümesi, kalanını test kümesi olarak oluşturduk. Daha sonra bu oranı kademeli olarak arttırarak en sonda %99 oranında eğitme kümesi olacak şekilde ayarladık. Bu deneyle eğitme/test kümelerinin büyüklük oranının sonuca olan etkisini inceledik. İkinci parametre olarak eğitme kümesini oluştururken ana veriden sıralı olarak veya rastgele seçim yapılmasını sağladık. Böylece sıralı veya rastgele seçimin sonuca olan etkisini incelemiş olduk. Tahmin algoritmalarını hem eğitme kümesi üzerinde hem de test kümesi üzerinde çalıştırıp hata oranlarını kıyasladık.

Sonuçları değerlendirirken istatistiksel analiz yöntemiyle sonuca etkisi olan parametreleri araştırdık. Bu analiz sonucunda sonuca en çok etkisi olan parametrelerin test tipi, kullanılan algoritma ve veri olduğu, eğitme verisi oranının ise fazla etkisi olmadığı ortaya çıktı. Grafiksel değerlendirmede ise bu çalışmada kullandığımız veriler üzerinde istatistiksel yöntemlerin makine öğrenmesi yöntemlerine kıyasla daha iyi tahmin yaptığını gözlemledik. Özellikle LSTM'in bazı istisnalar hariç birçok durumda en düşük performansı gösterdiğini gördük. Doğrusal modellerin ise genel olarak yüksek performans gösterdiğini gözlemledik. Çalışma süreleri kıyaslandığında ise yine doğrusal modellerin makine öğrenmesi algoritmalarına göre çok daha hızlı çalıştığı gözlemlendi.

Bu çalışmada kullandığımız verilerden en büyüğü olan Macaristan iklim verisi 96453 satır içeriyor. Verilerin bazı algoritmaların performans göstermesi için yeterince büyük olmaması sonucu etkilemiş olabilir. Dolayısıyla bu çalışmadaki bulguları genişletmek adına çok büyük verilerle aynı deneyler tekrarlanabilir. Ancak bu çalışmada kullandığımız veriler üzerinden ulaştığımız sonuç, zamansal verilerde tahmin yaparken istatistiksel metotların makine öğrenmesi metotlarına oranla hem daha performanslı hem de daha hızlı çalıştığıdır.

1 INTRODUCTION

In time series analysis, forecasting is considered as a crucial task . It helps making decisions about the studied system. In its most basic form, forecasting can be defined as predicting the future action of the studied system by using all available historical data. We encounter these kinds of dynamic systems whose components vary as time goes by in many different domains such as stock markets, weather, health etc. Traditionally, forecasting task is realized by using statistical methods. However, right now, we come across that machine learning (ML) methods start to be used very commonly for this important task (Ahmed et al., 2010).

Recent studies reveal the distinctive performance of ML methods on different types of data. We encounter important results of ML methods' application on real-world data. For instance, in internal medicine domain, especially in cardiovascular medicine physicians identify problems and abnormalities by observing diagnostic images. Since cardiovascular diseases are in the world's most important causes of death, these images must be examined carefully to detect the problem. That is why; ML methods are used to interpret them. They considerably increase the reliability of the diagnostics. Experiments that were run with several learning algorithms prove that ML methods could achieve a similar performance to doctors (Šajn & Kukar, 2011). Another considerable domain that ML methods exhibit high performance is natural language processing (NLP). Wee Meng Soon et al. proposed a ML solution to determine if given two expressions allude to the same thing in the world (Soon et al., 2001).This process is called coreference resolution and in NLP systems, it is an essential task. They performed ML by using a learning algorithm on a modest corpus of training data with coreference chains of noun phrases and they obtained encouraging results. Dynamic systems make use of ML methods as much as other domains. N. Justesen et al. apply ML to show how to use deep learning to play video games (Justesen et al., 2017). They present different genres of video games and observe how deep learning techniques should differ from one genre to another. For basic games, like arcade games, these techniques have better than human performance but there are still many open challenges in more complex games. Conferences like the IEEE Conference on Computational Intelligence in Games organize competitions for game environments.

Although we encounter that ML methods' usage increases accuracy and consequently improves the performance of some systems as we explained above, there is only limited objective evidence to prove their performance as a forecasting method. We do not come across enough benchmarks to compare the performance of ML methods with their alternatives in the literature.

Recently, Makridakis et al. aimed at performance evaluation of different methods on forecasting process(Makridakis et al., 2018b). They evaluate the accuracy and computational requirements of the machine learning methods by comparing them to those of statistical methods on a subset of 1045 monthly series from the 3003 of the M3 Competition. They use eight common statistical methods and eight prominent ML ones. They build the forecasting model according to the first n-18 observations where n is the length of the series and then they produce 18 forecasts. They also record the computational complexity of the methods along with their accuracy. They use mean absolute percentage error (sMAPE) and the mean absolute scaled error (MASE) as accuracy measures. As a result, it is observed that the computational requirements of the machine learning methods are considerably greater than those of statistical methods and the latter has higher accuracy and makes better forecasting. This surprising result raises questions if the state-of-art methods' performance is as good as the traditional methods' performance in forecasting tasks on different data. In this study, we focus on this question. We aim to perform a quantitative comparison of several algorithms on timely evolving data from different domains such as stock market, weather and cryptocurrency. This research differs from Makridakis' work mainly in terms of data used and forecasting models used. We create and use a new data, which is Borsa Istanbul stock market data, that had not been used before in similar researches. We do not use fully statistical methods such as ARIMA instead, we use more linear models, make multivariate instead of univariate forecasting and most importantly, we do not focus only on making forecasts but we aim to see the effect of various parameters that are explained further in Chapter 2 to the model's performance.

1.1 Literature Review

In this part, we explain various previous studies that aim at comparing forecasting methods. Perhaps one of the most prominent research on this topic was proposed by (Ahmed et al., 2010). In their work, they presented a large scale comparison for the most important ML models for time series forecasting. They used the monthly M3 time series competition data. M3 time series competition data includes data from different domains and types such as micro, industry, macro and finance. They have different time intervals like yearly, quarterly, monthly and custom. In this study, Ahmad and his colleagues chose to use the monthly data. The models they used included multilayer perception, Bayesian neural networks, CART regression trees, K-nearest neighbor regression, support vector regression, generalized regression neural networks(kernel regression), radial basis functions and Gaussian processes. They observed significant differences between different methods. They deducted multilayer perceptron and Gaussian processes give the best results in their experiments. But it was concluded that since there were many parameters to consider while performing a comparison between statistical models and machine learning methods, the results were mixed. They stated that it would be interesting to see when a different time series data is used in this comparison. They expect to have a different ranking in the accuracy of the models when the data changes.

A different comparison research was performed by (Alon et al., 2001). They used an aggregate retail sales data set with strong trend and seasonal patterns. They defined the task of forecasting the patterns in a data set as a long-standing issue in time-series analysis and thus investigated the best model to forecast these patterns. They used artificial neural networks(ANN) and traditional methods for regression which they believed were accurate for modeling trend and seasonal fluctuations. The error metric used in this research was mean absolute percentage error(MAPE) because of its popularity in the forecasting literature. As a conclusion they declare that the ANN is the best across different forecasting periods. It is followed by Box-Jenkins method. They observed that ANN outperformed the traditional statistical methods where economic conditions were relatively volatile. However, when the conditions became more stable, statistical methods performed with lower error rates. Thus, it was assumed that ANN had lower error rates on unstable, volatile data than traditional statistical methods.

Since there were various researches on comparison between different forecasting models, it was necessary to separate accurate studies from the others. That is why, a study was performed by (Adya & Collopy, 1998). In this work, they identified 11 guidelines to evaluate the performance of artificial neural networks in forecasting. They made a research to determine the previous work done on comparing different regression models. They filtered the researches and ended up with 48 studies between 1988 and 1994. They filtered the studies according to two questions which interrogate if the study appropriately evaluates the predictive capabilities of the proposed network and if the study implements the neural network in a way that it performs well. They call these two questions effectiveness of validation and effectiveness of implementation. As a result, they concluded that only 11 out of 48 studies met all of the criteria for effectiveness of validation and implementation. As a result, they claimed that when neural networks are effectively implemented and validated, they have a high potential for forecasting and prediction.

A similar study was performed by (Laurent et al., 2011). They claimed that much of the work of finding new algorithms to use in tasks such as indexing, classification, regression and segmentation had had two types of experimental flaws. They defined these flaws as (1) implementation bias and (2) data bias. They claim that the existence of these flaws made the studies lose their generalizability to real world problems. Thus, the "improvement" that was offered by the new algorithms was artificial and was not reproducible for different data sets. To prove their point, they reviewed more than 360 papers, included a subset of 57 papers and tested them on 50 real world, highly diverse data sets. They demonstrated that many of the results in the literature had very little generalizability to real world problems. We should note that this was not a study to criticize the papers but to increase the effectiveness of the researches done. For that purpose, they give certain suggestions for researchers which involve testing algorithms on a wide range of data sets, designing the experiments implementation bias-free, using different similarity measures when comparing and finally sharing all data and code to allow reproducibility.

In order to evaluate the reliability and accuracy of various forecasting methods, Makridakis Competitions(M-Competitions) were started in the year 1982 by Spyros Makridakis. In 1982, a subsample of 111 out of 1001 time series were used to test 15 methods with different variations. It was concluded that statistically complicated methods did not necessarily provide better forecasts than simpler ones. The accuracy measures influence the results considerably. A second competition was held in 1993 which had a larger scale compared to the first one. The results were similar to the first competition. In the year 2000, M-3 Competition was held, with the inclusion of more methods and researchers than the ones in M-Competition and M-2 Competition. 3003 time series from different domains like finance and industry were used for evaluation. Five measures were used for evaluation of the forecasting performance which are sMAPE, average ranking, median symmetric absolute percentage error, percentage better and median relative absolute error. The results were discussed in the article (Makridakis & Hibon, 2000). After the M3 Competition has done in the year 2006, Artificial Neural Network and Computational Intelligence Forecasting Competition(NN3) was introduced. They were a replication of M3 Competition with a neural network and computational intelligence(CI) extension to demonstrate the progress made in the passing years until the last Makridakis Competition (Crone et al., 2011). The competition showed that ensembles of CI approaches performed with lower error rates than combinations of statistical methods. Moreover, for the most complex subset of short and seasonal series, neural networks outperformed all the other methods. This result highlighted the ability of neural networks to handle complex data.

The latest Makridakis Competition was held in the year 2018 (Makridakis et al., 2018a). It was called M4 Competition. It referred to the continuation of three previous competitions. The main goal of M4 was to repeat the results of the competitions before and extend their scope. They first, considerably increased the number of series used, second, included ML forecasting methods and third evaluated both point forecasts and prediction intervals. At the end of the competition, they concluded that the most accurate methods were generally combinations of mostly statistical approaches. A surprising result was that a hybrid approach of both statistical and ML features which averaged almost 10% more accurate than the combination benchmark. But in general, pure ML methods had low performance with none of them passing the combination benchmark.

1.2 Objective and Contributions

Our main objective is to perform a quantitative comparison on timely evolving data sets by using different methods including not only traditional statistical methods but also state-of-art ML methods when training and forecasting. We use a tool called CARET which stands for Classification and Regression Training to fit the models. We do not try to fully optimize the parameters of each forecasting model, it is not the goal of this research. Instead, we let CARET tune the parameters and focus on evaluating the performance of different parameters. These parameters include using different percentages of the whole data as training data, selecting data randomly or regularly for training(data selection type) and making predictions on training data or test data(prediction process type). We use financial and weather data to test the algorithms. The methods and the data sets used in this research are explained in detail further in this paper in Chapter 2. The main contributions of this work include;

1. A data set of Borsa Istanbul's XU100 index from the year 2005 to 2018 formed and preprocessed by us .

2. A quantitative comparison of statistical forecasting models and state-of-art machine learning models.

   (a) Statistical Analysis: by using ANOVA and MANOVA

   (b) Error Metrics

   (c) Complexity: measured with execution time

3. Testing the forecasting models with certain parameters:

   (a) Data Ratio: percentage of data used for training the model

   (b) Prediction Process Type: testing the forecasting models on the test set or the training set

   (c) Data Selection Type: picking the training set's elements in random or in regular order from the data.

4. Data Comparison: checking if the nature of the data influences the result.

We aim to achieve our goals by using 5 different data from 3 domains to test the forecasting models. We then evaluate their performance according to statistical analysis with ANOVA and MANOVA and also according to our error metrics which are explained in Chapter 2. We look for cases of overfitting and underfitting, examine their possible causes and solutions. In Chapter 2, we present the data sets used in this research in detail. Then, we explain the details about the CARET tool that we used for managing model building and testing tasks. In the following part, we explain the regression algorithms used in this research with their properties. At last, We present our evaluation criteria and our comparison metrics. In Chapter 3, we show the results and evaluate them. We first perform statistical analysis, then we observe the error metrics for the algorithms. We investigate the effect of using different data, the dependencies to the random or regular data selection when model building , different prediction process type and we make a summary of the results. Finally, in Chapter 4 we make a general summary of the research and explain the future work that can be done.

2 METHODOLOGY

In this part, we present the methodology that we proposed to compare the accuracy of different algorithms in forecasting. We first define the data used in the experiments, second we explain our experimental methodology by giving the detail about how the training and testing processes are performed. We then introduce the algorithms used in this work to make forecasts. It is followed by explaining how we evaluate statistically and empirically the results of our experiments. Then we introduce error metrics that we have used to compare the algorithms quantitative performance and their speed.

2.1 Data Set

We concentrate on 5 different data sets from 3 different domains; weather, stock market and cryptocurrency. All of these systems have a nature of evaluation over time. Hence, they are appropriate for the main purpose of this study. We explain each of the data set in the following part. To better understand the data, we show the trends of forecasting area visually and Pearson Correlation Coefficient (PCC) values between different attributes of each data set.

2.1.1 Stock Market Data Set

Borsa Istanbul is the sole exchange entity of Turkey combining the former Istanbul Stock Exchange(ISE) and the Derivatives Exchange of Turkey under one organization. Borsa Istanbul calculates different quality indices for the markets so that investors can follow the movements in the market. For the Equity Market, a total of 324 indices are calculated, 54 of which are instantaneous and 270 at the end of the session. In this work, we extract Istanbul stock exchange (BIST100) data set which is one of these indices. BIST100 Index is used as the basic index for Borsa Istanbul Equity Market. In order to measure the common performance of 100 shares with the highest market value and transaction volume, traded on Borsa Istanbul markets. The fact that BIST100 index is one of the most basic indices of Borsa Istanbul has been effective in selecting this index in our study.

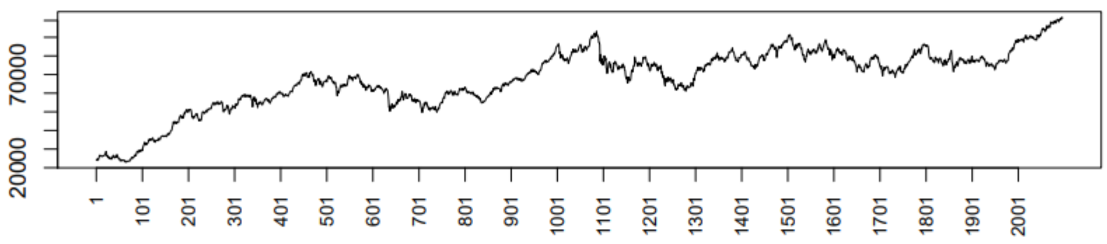

Figure 2.1: End Value of XU100 Index

We collected the raw stock market data between 2005 and 2018 for Borsa Istanbul from their official data store by taking necessary permissions. Our data includes 2097 entries. It consists of the XU100 index during trading days. This index represents 100 prominent stocks from the market, which have the highest market value and highest trading volume. We collected, preprocessed and prepared this data to use for this research. The data includes the fields of [date], [index code], [price index

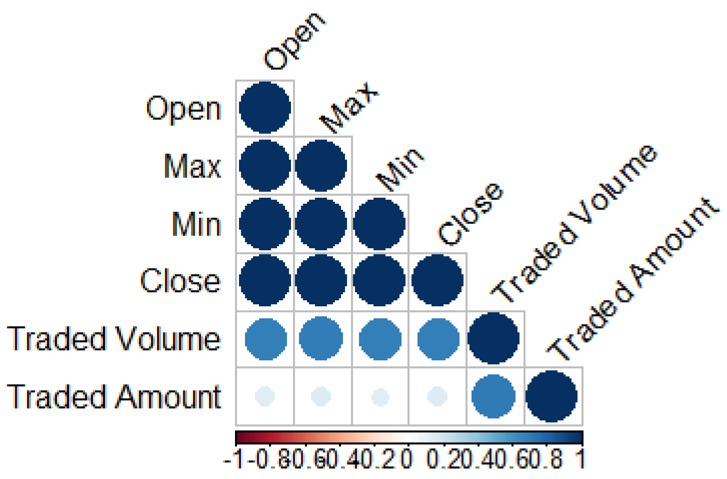

Figure 2.2: Visual Representation of the Correlation Matrix of XU100 Data

opening value], [price index end value], [price index min value], [price index max value], [traded volume] and [traded amount]. Among them, price index opening value shows the value of the index for the beginning of the trading day while price index end value corresponds to the value of the closing of trading day. We show the trend of end value changes between December 2006 and June 2017 in figure 2.1. Accordingly, we can say that in eleven year, end value exhibit stationary and very gently non linear increasing trend. Its value increases five times from 20000 to 100000 in about eleven years. In fact, we examine other attributes, i.e. open, min, max values. They have similar trends with end value.

Table 2.1: Correlation Matrix of XU100 Data

	Open	Max	Min	End	Traded Volume	Traded Amount
Open	1	0.999	0.999	0.998	0.688	0.139
Max	0.999	1	0.999	0.999	0.694	0.147
Min	0.999	0.999	1	0.999	0.683	0.133
End	0.998	0.999	0.999	1	0.689	0.141
Traded Volume	0.688	0.694	0.683	0.689	1	0.702
Traded Amount	0.139	0.147	0.133	0.141	0.702	1

PCC numerical results for each couple of attributes are represented in table 2.1 under the form of symmetric correlation matrix. We also show visual version of PCC results in figure 2.2. These results show us that there is a very high correlation between open, max, min and end values. Nevertheless, traded volume and amount seem non related to other four attributes.

2.1.2 Cryptocurrency Exchange Data Set

Over the last decade, blockchain technology has long gained attention in the industry. Nowadays scientific community considers the blockchain technology as a big revolution. The blockchain technology has a wide spectrum of practical applications in various domains with financial and non-financial approaches.

However, economists and the people working in the finance industry address on its financial benefits. They try to integrate the blockchain technology in the current banking systems especially to the digital payment systems. Moreover, they attempt to use this technology in the digital currency and adapt it to current systems. All these working environments can be considered as a strong indicator of the fact that the blockchain technology has great potential to be the new engine of progression in digital economy. Whereas the blockchain is surely more than that compared to the industry, there is a big gap in the scientific community(Atzori, 2015). During the past few years, the blockchain technology became one of the most popular research subjects. The researchers work on how to integrate the blockchain technology and the working mechanism into other fields(Pilkington, 2016).

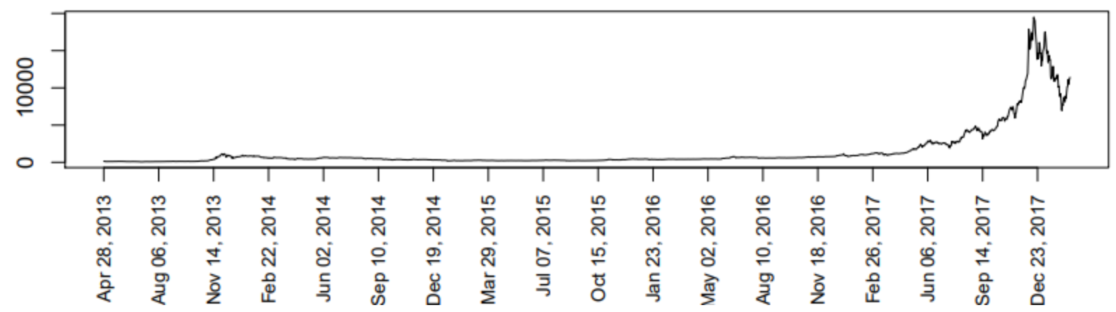

Figure 2.3: End Value of Bitcoin Prices

A blockchain is a distributed database that stores the logs of all transactions. It can be also defined as a public ledger that contains all the transactions. These transactions are, indeed, the digital events that have been performed and shared between users. Due to the blockchain technology mechanism, each executed transaction should absolutely be verified by consensus of a majority of the participants in the system before storing in the public ledger permanently. Once a transaction is verified, it can not be undone or modified anymore. Thus, the blockchain technology provides a log for every transaction performed. Again based on the mechanism, each participant can have the database of the transactions which makes to manipulate the system highly difficult. From a data analysis point of view, the blockchain is a data management technology where the data is stored in a public ledger where this ledger can be stored in every participant. Anyone can be a participant of this network, called a node of the network. This public ledger mechanism provides the secure and anonymous data management with data integrity without any third party organization in control of the transactions.

Table 2.2: Correlation Matrix of Bitcoin Price

	Open	High	Low	Close	Volume	Market Cap
Open	1	0.998	0.997	0.996	0.943	0.999
High	0.998	1	0.998	0.999	0.946	0.998
Low	0.997	0.998	1	0.998	0.932	0.997
Close	0.996	0.999	0.998	1	0.940	0.996
Volume	0.943	0.946	0.932	0.940	1	0.944
Market Cap	0.999	0.998	0.997	0.996	0.944	1

The first cryptocurrency data set is on bitcoin from an anonymous exchange. The data shows the bitcoin value (USD) and relevant coin information timestamped from 28 April 2013 to 20 March 2018. There are 1760 daily records. We have following attributes for each record: [Date],[Open],[High],[Low],[Close],[Volume],[Market Cap]. The change in Close feature according to Date is given in figure 2.3. According to it, we see that there is a very stable trend until June, 2017. Then, we observe a very sudden exponential increase until October, 2017. It is followed by sudden decrease.

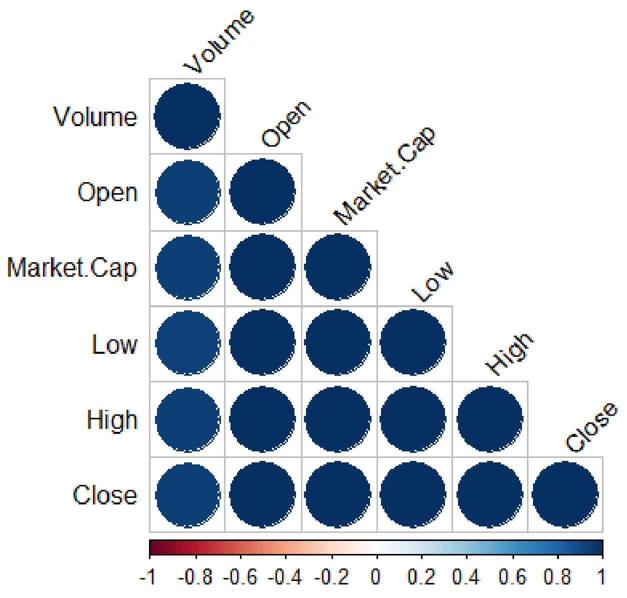

Figure 2.4: Visual Representation of the Correlation Matrix of Bitcoin Price Data

PCC results are given in table 2.2 and in figure 2.4. We can see that the coefficients are very close to 1.0 for every couple of attributes. Thus, there is a highly correlated relationship between features. Dark blue color on the graph represents a high relationship.

The second cryptocurrency data set is on Ethereum from an anonymous exchange. It is the second well known coin in blockchain market. The data shows the Ethereum value(USD) and relevant coin information timestamped from 8 August 2015 to 20 February 2018. There are 928 daily records. We have following attributes in this data set: [Date], [Open], [High], [Low], [Close], [Volume], [Market Cap].

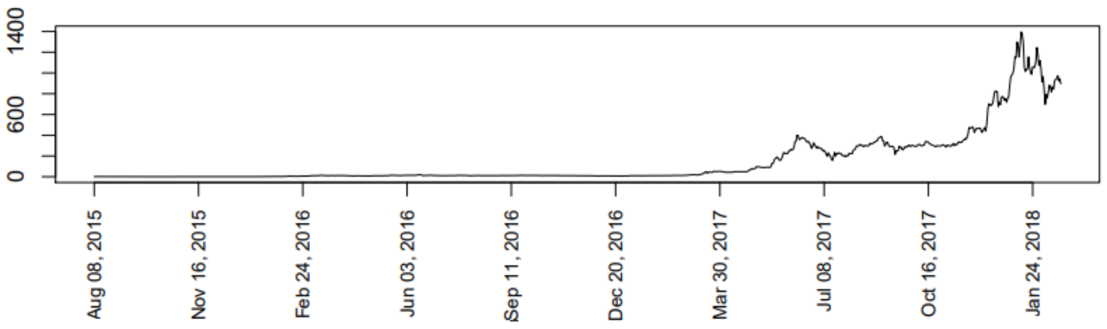

Figure 2.5: End Value of Ethereum Prices

We show the 928 days trend of end value of Ethereum prices in figure 2.5. As we observed in bitcoin data set, the values stays stable until March, 2017. Then there is an step-by-step sudden increases. We observe the main increase between October, 2017 and June, 2018. After June, 2018, a decrease start to be observed.

Table 2.3: Correlation Matrix of Ethereum Data

	Volume	Low	Close	High	Market Cap	Open
Volume	1	0.897	0.913	0.923	0.914	0.913
Low	0.897	1	0.998	0.996	0.995	0.995
Close	0.913	0.998	1	0.998	0.995	0.995
High	0.923	0.996	0.998	1	0.998	0.998
Market Cap	0.914	0.995	0.995	0.998	1	0.999
Open	0.913	0.995	0.995	0.998	0.999	1

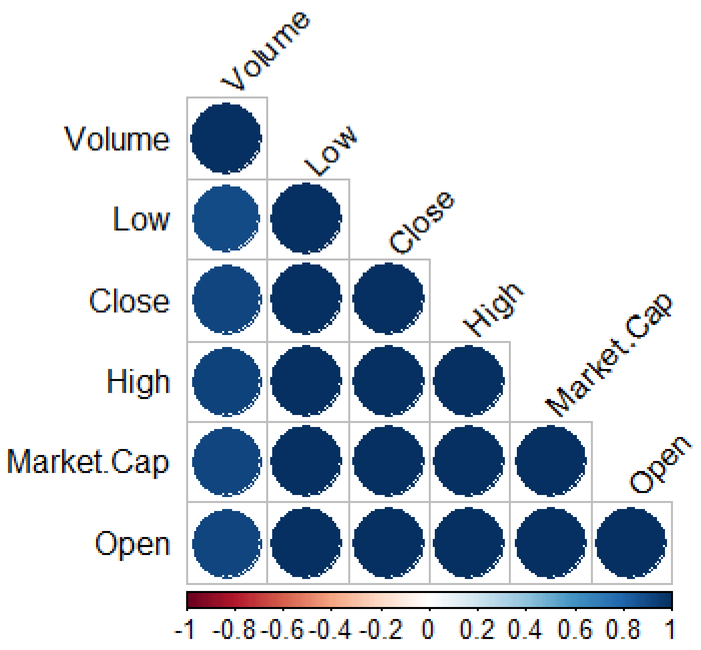

Figure 2.6: Visual Representation of the Correlation Matrix of Ethereum Data

PCC results are given in table 2.3 and figure 2.6. Here again, we observe a very high correlation between every couple of attributes as it is for bitcoin data set. It means there is a strong relation for every attributes.

2.1.3 Weather Data Sets

To compare the methods, the third domain we used is historical weather information data. Weather data is commonly used in data science studies since they involve a real-world forecasting activity. The performance of the forecasting models may be compared to the performance of the forecasting of the physical weather forecasting tools and calculations. The data is formed by putting together many features of the weather including detailed features related to temperature, humidity, wind and pressure. In this study, we used two different weather data sets to compare our forecasting models.

The first weather data set is from Szeged city from Hungary, timestamped from 1

April 2006 to 9 September 2016. We have 96453 hourly records. Each record has following attributes: [Formatted Date], [Summary], [Precip Type], [Temperature (C)], [Apparent Temperature(C)], [Humidity], [Wind Speed (km/h)], [Wind Bearing (degrees)], [Visibility (km)], [Loud Cover], [Pressure (millibars)], [Daily Summary]. One can see the trend of mean temperature for Szeged city in figure 2.7. It seems

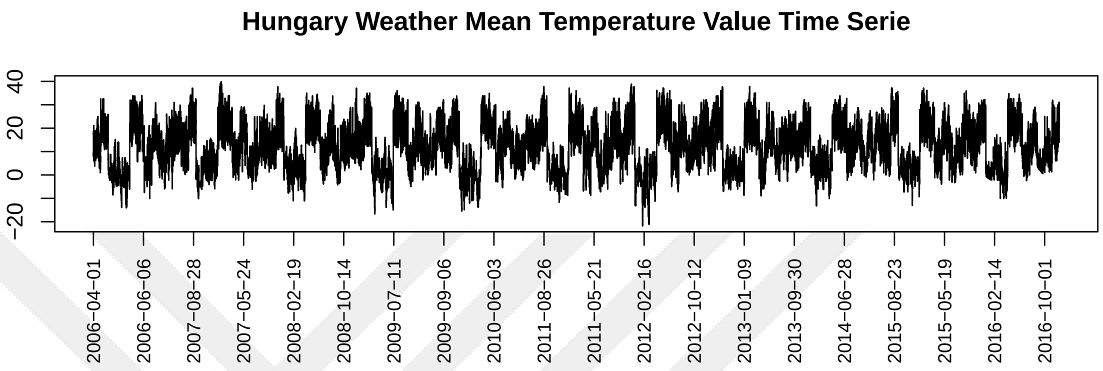

Figure 2.7: Temperature of Hungary Weather

this value is rather noisy, seems like having a stationary signal with daily changes. Because the data is collected per hour, we observe sudden but small changes in the data.

PCC results are given in table 2.4 and in figure 2.8. Accordingly, we observe that there is no obvious relation between any features except [Apparent Temperature] and [Temperature]. Indeed this is an expected fact.

The second weather data set is from Madrid city from Spain timestamped from 1 January 1997 to 31 December 2015. There are 6812 daily records. We have following attributes in this data set: [CET], [Max TemperatureC], [Mean TemperatureC], [Min TemperatureC], [Dew PointC], [MeanDew PointC], [Min DewpointC], [Max Humidity], [Mean Humidity], [Min Humidity], [Max Sea Level PressurehPa], [Mean Sea Level PressurehPa], [Min Sea Level PressurehPa], [Max VisibilityKm], [Mean VisibilityKm], [Min VisibilitykM], [Max Wind SpeedKm/h], [Mean Wind SpeedKm/h], [Max Gust SpeedKm/h], [Precipitationmm], [CloudCover], [Events], [WindDirDegrees].

The 6812 daily trends of mean temperature for Madrid city can be seen in figure 2.9. As it is observed for Szeged weather, we encounter again a stationary signal.

Table 2.4: Correlation Matrix of Spain Weather Data

	Humidity	Visibility	Pressure	Wind Speed	Wind Bearing	Apparent Temperature	Temperature
Humidity	1	0.103	0.007	-0.017	-0.002	-0.604	-0.634
Visibility	0.103	1	-0.001	-0.003	-0.012	-0.092	-0.634
Pressure	0.007	-0.001	1	0.004	-0.009	-0.003	-0.008
Wind Speed	-0.017	-0.003	0.004	1	0.012	-0.010	-0.003
Wind Bearing	-0.002	-0.012	-0.009	0.012	1	0.030	0.031
Apparent Temperature	-0.604	-0.092	-0.003	-0.010	0.030	1	0.992
Temperature	-0.634	-0.092	-0.008	-0.003	0.031	0.992	1

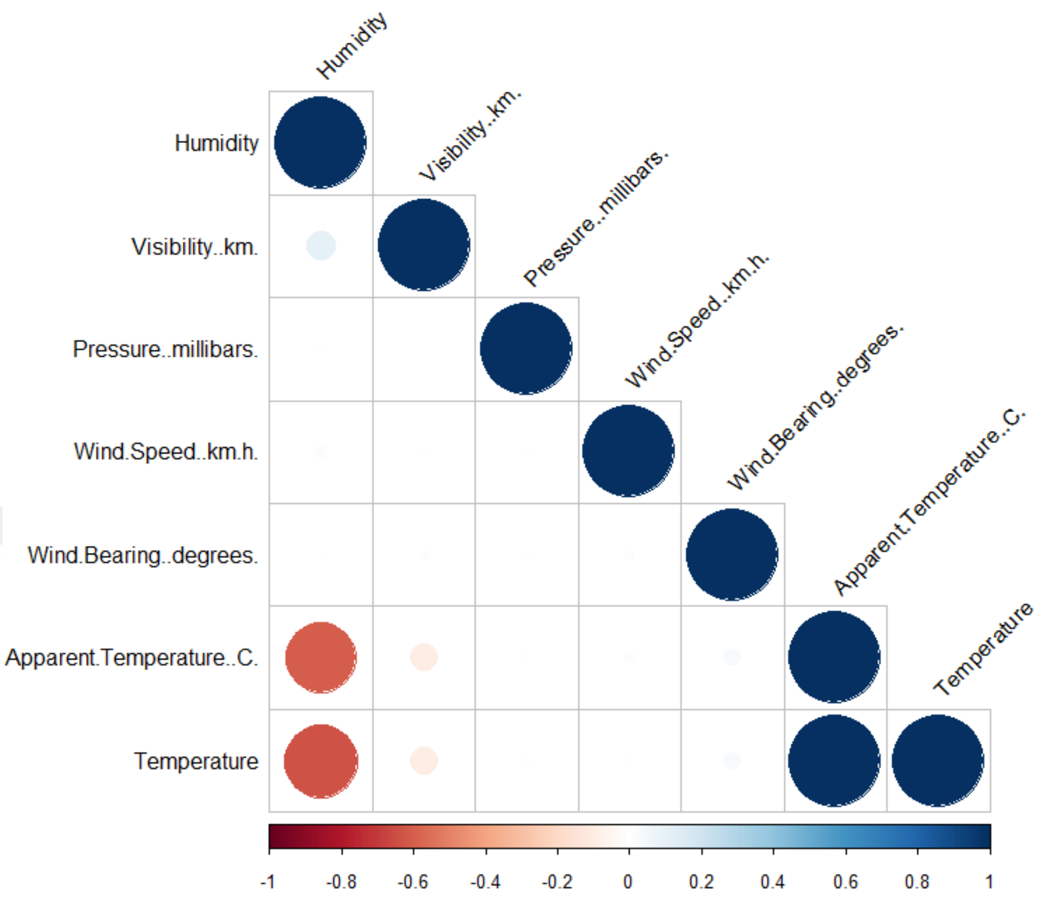

Figure 2.8: Visual Representation of the Correlation Matrix of Weather Hungary Data

However, this signal is less noisy. Hence we see clearly the increase and decrease points in an annual period.

PCC results are given in table 2.5 and in figure 2.10. Accordingly, we observe a correlation between few of the features such as between the ”min” and ”max” values of same features, i.e. max and min humidity.

2.2 Regression Algorithms’ Comparison Framework

In this section we concentrate on the experimental set-up that creates the major part of this work. The main purpose of this study is to evaluate the quantitative performance of different regression algorithms by using objective performance

Table 2.5: Correlation Matrix of Spain Weather Data

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
MaxHumidity(1)	1.000	0.730	0.098	0.083	0.252	0.034	-0.043	-0.388	-0.154	0.160	-0.061	-0.718	-0.606
MinHumidity(2)	0.730	1.000	-0.030	-0.019	0.362	-0.024	-0.210	-0.499	-0.092	0.231	-0.081	-0.758	-0.479
MaxSea.Level(3)	0.098	-0.030	1.000	0.914	-0.056	0.004	-0.038	0.039	-0.427	-0.199	-0.265	-0.083	-0.316
Pressure(4)	0.083	-0.019	0.914	1.000	-0.073	-0.015	-0.203	-0.025	-0.485	-0.145	-0.225	-0.047	-0.264
SeaLevelPressure(5)	0.252	0.362	-0.056	-0.073	1.000	0.008	-0.050	-0.066	-0.040	0.023	-0.068	-0.285	-0.170
CloudCover(6)	0.034	-0.024	0.004	-0.015	0.008	1.000	0.037	0.023	-0.027	-0.034	-0.016	0.003	-0.056
WindDirDegrees(7)	-0.043	-0.210	-0.038	-0.203	-0.050	0.037	1.000	0.386	0.129	0.043	0.092	0.084	0.080
MaxVisibility(8)	-0.388	-0.499	0.039	-0.025	-0.066	0.023	0.386	1.000	0.018	-0.112	-0.019	0.350	0.197
MaxWindSpeed(9)	-0.154	-0.092	-0.427	-0.485	-0.040	-0.027	0.129	0.018	1.000	0.002	0.131	0.032	0.218
MinDewpoint(10)	0.160	0.231	-0.199	-0.145	0.023	-0.034	0.043	-0.112	0.002	1.000	0.809	0.326	0.560
DewPoint(11)	-0.061	-0.081	-0.265	-0.225	-0.068	-0.016	0.092	-0.019	0.131	0.809	1.000	0.583	0.774
MaxTemperature(12)	-0.718	-0.758	-0.083	-0.047	-0.285	0.003	0.084	0.350	0.032	0.326	0.583	1.000	0.858
MinTemperature(13)	-0.606	-0.479	-0.316	-0.264	-0.170	-0.056	0.080	0.197	0.218	0.560	0.774	0.858	1.000

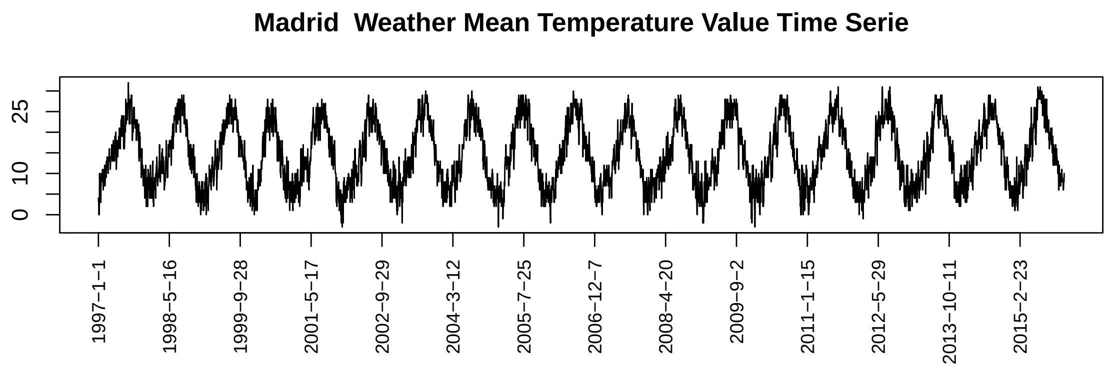

Figure 2.9: Temperature of Madrid Weather

evaluation metrics on different data which have evolution over time. The main issue in this evaluation is creating the environment for the experiments. We compare several algorithms by using various regression evaluation metrics. Each algorithm has a different prediction strategy. The performance of each algorithm depends on its input parameters. One can concentrate on optimizing the prediction. For this purpose, feature selection, algorithms' parameter regulation or data preparation can be applied. However, these operations are out of the scope of this work. We want to see the overall limits of the algorithms depending on our predefined experimental parameters. These parameters can be listed as follows;

1. Data
2. Algorithm's Category
3. Evaluation Metric, i.e. Error Metric
4. Model Building Parameters
   (a) Data Ratio: percentage of data amount saved for model built
   (b) Prediction Process Type: train vs test
   (c) Data Selection Type: random vs regular

We create experiment environment by chaining the values of these parameters. We run each algorithm belonging to different algorithm category. Then we calculate evaluation metric result for each data. When building the model, we use different experimental parameters. Here, we want to underline that these are not algorithm

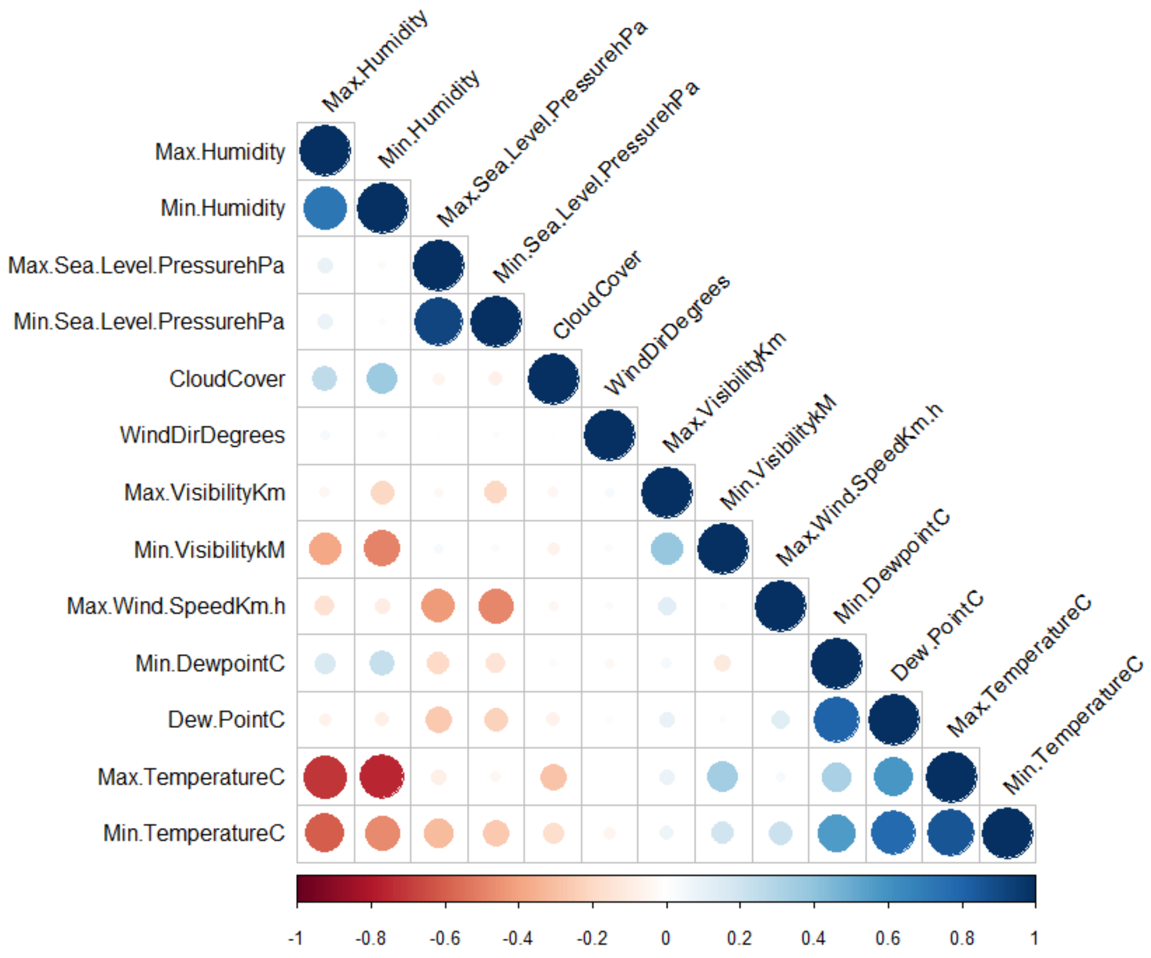

Figure 2.10: Visual Representation of the Correlation Matrix of Spain Weather Data

dependent parameters. For any algorithm, to build the model, we use different prediction process type, different data selection type and different data ratio. Among those parameters, the details of data are explained in previous section. The exhaustive details of algorithms' category and evaluation metrics will be explained in the following sections. In this section, we concentrate on CARET, the platform on which we run the algorithms. Then, we give different values of first three major parameters that are listed above in our experimental set-up. It is followed by explaining model building parameters by giving the hypotheses that we expect to happen as the results of the experiments. Finally, we explain the hypotheses of different data usage on algorithm's performance.

2.2.1 CARET Package

The main purpose in our research is not to optimize the performance of the algorithms for each data set separately since this would require testing them with a vast number of different parameters and comparing the results each time. For example, in order to optimize the k-nearest neighbor model for the Ethereum data, we have to run it multiple times with different values of k parameter. There are already many researches which look into the influence of parameters to the result in this model(Batista & Silva, 2009) and the optimization of each model is not in the scope of this particular research. That is why; we make use of a package called CARET which stands for "Classification and Regression Training"(Kuhn, 2008). It contains a set of functions to ease the process for creating predictive models. It has tools to split and preprocess the data. It can also perform tasks like feature selection, model tuning using resampling and variable importance estimation. It is open-source and is available on CRAN for use. We mainly use caret to build our prediction models because it automatically tests each model with multiple parameters to optimize the results. Since CARET runs the models with multiple parameters, the execution time of each algorithm is expected to be higher. The code used to build our experiments is shared in the References part of this research Github (2018).

2.2.2 Major Parameters' Values

The major parameters, *data*, *algorithms' category* and *evaluation metrics, i.e. error metrics*, of experiments are explained in detail in the previous and following sections respectively. Here, we mention different algorithm categories and error metrics.

Since CARET package contains around 175 different algorithms, we limit our experiments on well-known regression algorithms. We concentrate on 17 different algorithms and separate them into 4 different categories; (1) Linear Models, (2) Tree-Based Models, (3) Neural Network-Based Models and (4) Other Algorithms. When creating these categories, our main intuition is to create logical classes that represent the main model building strategies that the algorithms use. Rather than evaluating each algorithm separately, we compare algorithms' categories. We want to evaluate the strategies that they use. These algorithms are explained in detail later on.

We use 5 different error metrics to compare the algorithms' performance. The error metrics we use for our research are (1) root mean square error(RMSE), (2) mean absolute error(MAE), (3) R squared, (4) mean squared error(MSE) and (5) execution time. In our experiments, we do not expect that the comparison of the algorithms will depend on the error metrics. Although we do not expect that the order of algorithms' performance will not be affected by those metrics, different metrics can reveal the difference of the algorithms at different scale. That is why; their usage can be complementary to each other. The details of each error metric is explained later on.

2.2.3 Model Building Parameters

At this section, we concentrate on the parameters that we use when building the models; *data ratio which is the percentage of data amount saved for model creation*, *prediction process type* and *data selection type for model creation.* We also explain basic hypotheses that we propose as the expected result of the effect of different values of those parameters.

Our first model-related parameter is the data ratio parameter. It is the percentage amount of the total data we are going to use for model creation, a.k.a, training. It starts from the value of 0.15 which refers to 15% of the total data. So if the value of this ratio is 0.15, the model will be trained with 15% of the total data and it will be tested on 85% of the data. After 0.15 value, we use 0.2 then we increase the ratio's value by 0.1 each time we run an algorithm until the value of the ratio reaches 0.99. This results in running the algorithm multiple times and recording the results at each time. According to the results we may decide how the training data set size affects the prediction accuracy and what is the optimized ratio for each algorithm on our particular data sets. Our first hypothesis is as follows;

H_1: We expect that the increase of the data that is used for model built will increase the performance of all algorithms.

We propose this hypothesis because the more data that the algorithms use at their training step, the more robust models they may create. Especially, for neural network based algorithms, we already know that they improve their model by the feed of new data income to the system. Indeed proving our hypothesis demands statistical evidence of this parameter effect on algorithms' result error metrics. These details will be explained in the Results section.

Our second model-related parameter is the prediction process type which means from which process we calculate the error metric results. Naturally, we use two different prediction process types: train and test. This means, we calculate the error metrics during the building of the model and also during the model validation. During model creation, we use 10-fold cross-validation. If an algorithm has an over fitting tendency by its nature, we expect that its training performance will be very high but its test performance will be relatively low. If an algorithm learns well from the data, we will observe an equilibrium at its training and test performance. Our hypothesis related to this parameter is given as follows;

H_2: If an algorithm overfits, its training and test performances will be quite different. Hence prediction process type parameter will be an indicator for overfitting algorithm detection.

Indeed, observing train and test performance of the algorithms together will be a natural proof of this hypothesis. In fact, this expectation should be seen as a claim rather than a real hypothesis.

Our third model-based parameter is the data selection type. It is the data order that we use for model built. We use two different data selection types; regular and random. In the regular selection, when we set the ratio variable, we obtain that ratio of the data set in order. For example, if the ratio variable is set to 0.2, we have the first 20% of the data in order as training set and the remaining 80% as the test set. But in the random selection we split the data randomly. For example, if the ratio variable is set to 0.5, the training set contains a random 50% portion of the whole data. These two different training methods help us understand how the time order in the data affects the performance of the algorithms. Our hypothesis related to this parameter is given as follow;

H_3: We expect to have similar error ratios after using random or regular data selection.

For H_3, we also want to underline that, here we compare regression methods. Most of these methods are not developed for time series analysis naturally. Indeed, they can be used for this purpose but our assumption is that the order of data does not have effect on the performance of regression algorithms. We aim to prove this hypothesis by observing the error metrics for the same data and for the same training ratio when their training set is picked regularly and randomly.

2.2.4 Hypotheses about Data

Before we start to deepen the algorithms and error metrics, we want to propose different hypotheses based on the observed nature of each data type. Although all data sets we used here have a nature of evolution over time, we see that their evolution has different trends. This causes different expectation for prediction independent from the algorithms that we use. The difference of data can create a cross impact on the algorithms' performance. In this section, we want to underline the possible cross impact due to the data difference.

For weather data, in general, we observe similar temperature values for each year in the same period. While, for the stock market data and the cryptocurrency data, the fluctuations in the closing price value make the data more unpredictable. There is no pre-defined pattern that we can observe at first sight. Moreover, the closing value does not depend on seasonality.

H_4: We expect to have similar error ratios on stationary data like weather data when we use random or regular data selection.

Firstly, since we apply different methods on these data, we find it logical to have varying results for each data as they differ in certain properties as we explained in previous section. Picking the training set in regular order from the whole data would result in the algorithm having a narrow look at the data. It might cause overfitting for all the data sets, especially if we have a low training/test ratio. It would not have accurate predictions in the data without seasonality since it does not have prior knowledge of the existence of a vast interval of predictions. But, with adequate ratio of the data given as training set, it may achieve good results in predicting the output of stationary data, which is the weather data.

H_5: We expect to have lower error ratio with stationary data after using random data selection and higher with regular data selection.

We will prove this hypothesis by testing the models with random and regular training test types on the weather data which is, by nature, acting similarly in specific times of the year.

Secondly, if we build the training set by picking the elements randomly from the whole data set, we may be able to overcome the overfitting problem in all the data sets since the training set would make the model be aware of outlying cases in the data. Nevertheless, it may well be possible that we encounter overfitting due to the fact that random picking elements from the data set does not guarantee picking the ones that would help the predictions.

H_6: We expect to decrease the overfitting problem when we use random data selection.

We compare the results of training randomly and regularly to see if training randomly helps solving the overfitting problem in our case.

Lastly, we expect that neural network based models have higher computational complexity than the other models.

H_7: Neural network based models have higher computational complexity than the other models.

We compare the execution times of the algorithms in order to prove this hypothesis.

2.3 Regression Models

In this part, we define the regression models used in this research and explain their general properties. We used 17 different algorithms which have different modelling strategies. For the sake of comprehensibility, we separate those algorithms into four different categories. Each category represents the main idea behind the algorithms it contains. Each algorithm's abbreviation in the CARET package is written in parantheses in the titles.

2.3.1 Linear Models

This section describes the linear models used in this research. Linear models can be defined as models which describe a continuous response variable in terms of one or more predictor variables in a function.

2.3.1.1 Linear Regression(lm)

Linear Regression is a statistical method to observe the relationships between dependent and independent variables. In its most basic form, there is one independent variable, x, which is also called explanatory or predictor, and the dependent variable, y, which is also called response or outcome, is explained as a lineaire function of x. This model is called “simple linear regression”. In general form, y is explained by several different dependent variables. This time, model is called “multiple linear regression”.

The model builds an equation of a straight line using these two different types of variables and assume that the relationship between them fall along a straight line. When the data points do not fall exactly on a straight line the equation is modified by adding in a statistical error variable to fit the model(Douglas C. Montgomery, 2012).

It is a very simple and easy to understand model with good interpretability and requires relatively low resources. But while it is a powerful and widely used method, it has certain drawbacks which limit its application and sometimes require thorough preprocessing of the data in order to obtain logical results. The base of the model itself is a limiting factor since it assumes that the data is normally distributed when it is generally not. Also since the real-life data is not supposed to be 100% completely linear the linear regression model is very prone to outliers. One can find the general formula of linear regression in the following equation. Here, when building the model, β parameters are estimated. There different methods for such estimation. One of the most common estimation method is least-square method. The other parameter, ϵ, reflects the statistical error of the model. More clearly, it is the error of estimated values.

$$y = \beta_0 + \beta_1 x + \epsilon \tag{2.1}$$

Linear regression is used in pattern recognition(Bishop, 2006), astronomy(Isobe et al., 1990), stock markets and many other domains which provide linear data.

2.3.1.2 Generalized Linear Model(glm)

The standard linear model is not capable of handling nonnormal responses, y, such as counts or proportions. This fact inspires the urge to develop a new model which can represent categorical, binary and other response types. These models are called generalized linear models (Faraway, 2006). To form the model, a vector of observations y having n components is assumed to be a realization of a random variable Y whose components are independently distributed with means. The vector is then specified as in terms of a small number of unknown parameters β_1,...,β_p inferred from the data. Thus, the equation can be represented as follows:

$$\mu = X * \beta \tag{2.2}$$

where X is the model matrix and β is the vector of parameters(P. McCullagh, 1991).

Generalized Linear Model can deal with categorical predictors and it is relatively easy to build. The equation provides a clear understanding of how each of the predictors have an effect on the outcome. Even if there are some advantages over the simple linear regression model, as far as a linear model is concerned, it is not possible to completely avoid outliers.

Bayesian Generalized Linear Model(bayesglm) is similar to the generalized linear model but it adopts the Bayesian perspective in the modeling.

2.3.1.3 Ridge Regression(ridge)

Ridge Regression algorithm is first introduced in 1970 by Arthur E. Hoerl and Robert W. Kennard (Hoerl & Kennard, 1970). It is a technique to analyze multiple regression data which have the problem of multicollinearity. Multicollinearity means having almost linear relationship among independent variables. If variables in a regression problem have multicollinearity, the first task is to determine if it causes a problem because it is possible to have division by zero which causes the calculations to fail. So, in case of multicollinearity, least squares estimates are unbiased when their variances are relatively large which raises the possibility of them to be far from the true value. To reduce the standard errors, ridge regression blends in a degree of bias to the regression estimates in order to stabilize the estimated values.

Ridge regression uses standardization of the variables by subtracting their means and dividing them by their standard deviations. Ridge regression procedures are performed on standardized variables and in the end, they are adjusted back into their original scale.

Ridge regression is very advantageous to use when there is multicollinearity in the data set since it uses bias to deal with the problem. Also it uses penalties which are calculated according to the estimates value and this helps to choose the important features in the data set and increases their influence.

Bayesian Ridge Regression(bridge) that we use in this paper is very similar to the original ridge regression algorithm. The difference is that

we assume each regression coefficient has expectation zero and variance 1/k.

2.3.1.4 Non-Convex Penalized Quantile Regression(rqnc)

Quantile regression is first introduced in a seminar paper in 1978 by Roger Koenker and Gilbert Bassett Jr (Koenker & Bassett, 1978). It is useful for studying connections between a response variable and a set of covariates and it is mostly used efficiently on heterogeneous data. Quantile regression methods estimate models for the conditional median function and other conditional quantile functions (Koenker, 2005). It is capable of making an accurate statistical analysis of the stochastic relationships between random variables. It is possible to use penalization in quantile regression to increase its efficiency. There are three mostly used penalty function types which are L1 or Lasso penalty(Tibshirani, 1994), nonconvex penalty(SCAD)(Fan & Li, 2001) and two-stage adaptive penalty(Stucky & van de Geer, 2015). L1 penalty has a low computational cost because of its convex structure and has a relatively high performance. SCAD, on the other hand, resembles L1 penalty but it changes into a quadratic function thus becoming non-convex. In our research we use non-convex penalization which is SCAD on our quantile regression algorithm.

Quantile regression allows the user to understand the relationships between different variables outside of the mean of the data which makes it possible to understand the outcomes which are not normally distributed and have nonlinear relationships with predictor variables. It is used in ecology(Cade & Noon, 2003), researches on education(Eide & Showalter, 1998) and in investigating public-private sector wage differentials in Germany(Melly, 2005).

2.3.2 Tree-Based Models

This section describes the tree-based models used in this research. Tree-based models are supervised learning models that can be used for both classification and regression problems. They are highly flexible and they make it possible to find out complex non-linear relationships in the data. In our research, we use 3 different tree-based models.

2.3.2.1 CART(rpart)

Classification and regression trees are machine-learning methods for constructing prediction models from data (Loh, 2011). The data space is recursively partitioned and each partition is fit using a simple prediction model. As a result, it is possible to build a decision tree to represent the partitioning. In classification with CART, dependent variables which have a limited number of unordered values are used and the error is measured in terms of misclassification cost. Regression trees are also for dependent variables which have continuous or ordered discrete values and the prediction error is generally measured by the squared difference between the actual and the predicted values. CART is similar to a regular decision tree algorithm so its main elements include rules based on a variable used to split the data in different branches, rules to stop the branching process and the prediction of the target variable.

There are some techniques that are used to increase the accuracy of CART. Bagging **treebag** builds n classification trees by using bootstrap sampling of the training data (Breiman, 1996). Bootstrap is a statistical method for estimating a quantity from a data sample. It separates the data into n samples and calculates the target value in each sample, then it takes the average of the sum of target values in each sample and uses the result as the estimate. After bagging uses bootstrap sampling to build the classification trees, it combines the predictions to create a meta-prediction. We use the regular CART and bagged CART in our project.

CART has certain advantages and drawbacks. It is not highly affected by the outliers in the input variables. It can use the same variables multiple times in different parts of the tree which may reveal complex interdependencies between variables. It can be used with different prediction methods to pick the input set of variables and it does not require a specifically distributed data. The most important drawback of CART is complexity. If the tree is large and has many branches, it becomes very time-consuming and complex to build the branches and construct the tree.

2.3.2.2 Random Forest(rf)

Random Forests combine many tree predictors together so that each tree depends on the values in a random vector which was sampled independently and with the same distribution for all trees in the forest (Breiman, 2001). The accuracy of a random forest depends on the strength of the individual tree classifiers and a measure of the dependence between them because when random forest is making a prediction, it takes the average of all the estimates of the individual decision trees in the model.

Random Forest can be used for both regression and classification tasks. It is easy to use and build because it does not require fine adjustments, parameters are fairly simple to understand. Overfitting problem is not very common in Random Forest but the main limitation of the algorithm is the large number of trees which make the algorithm run slow so it may not be effective to use it for real-time predictions.

Random Forest is commonly used in various fields such as banking, medicine and stock markets. Banks use it to find the customers who will be eligible to the new services the bank will offer in the future, it is also used for fraud detection. In stock markets, it is used to predict a stock's behavior in the future. In medicine, it is used to observe the patient's medical history to make predictions about the future diseases the patient may have.

2.3.2.3 Stochastic Gradient Boosting(gbm)

Gradient boosting creates additive regression models by sequentially fitting a simple parameterized function to current "pseudo"-residuals by least-squares at each iteration(Friedman, 2002). The "pseudo"-residuals refer to the grade of the loss function being minimized with respect to the model values at each training data point. In 1996, Breiman proved that putting randomness into function estimation could increase the performance. The same idea of randomness is put into the gradient boosting algorithm by drawing a subsample of the training data from the full training data set with no replacement at each iteration. Instead of the full sample, this subsample is used to fit the base learner and computations are made according to this subsample. This procedure is called stochastic gradient boosting.

Gradient boosting is commonly used in anomaly detection with supervised learning on highly unbalanced data such as DNA sequences and credit card transactions. Since the main task of gradient boosting is to optimize an objective function, it is possible to use this algorithm in almost all objective functions but they tend to be more sensitive to overfitting when it is dealing with noisy data and the training process is often longer.

2.3.3 Neural Network-Based Models

This section describes the neural network-based models used in this research. A neural network tries to replicate the human brain's functioning with diverse mathematical models. That is why, it can be briefly described as a network of neurons organised in layers.

2.3.3.1 Neural Network(nnet)

A neural network is a compilation of basic processing elements that are named units or nodes connected together to function just like an animal neuron(Gurney, 2004). Nodes function just like neurons in the human brain, they fire when they encounter sufficient amount of stimuli. In a node, the input data is combined with a set of coefficients or weights which increases or decreases the importance of the input for the training process. Then the product formed with the input data and the weight go to the node's activation function to decide if that signal should go forward in the network and have a part in the output.

Neural networks can work on almost any data. They do not require prior knowledge on the data. They also have high fault tolerance so failure of a few cells in the network does not prevent it from creating output. In spite of all these advantages, neural networks require powerful processors to run efficiently. Also, it is not easy to explain the behavior of the network and how the system infers certain outputs.

Neural networks are used in both classification and regression problems in various domains. Credit card fraud detection(Ghosh & Reilly, 1994), handwriting recognition(Graves & Schmidhuber, 2009), stock market prediction(Kimoto et al., 1990) are good examples of the real life application of neural networks.

2.3.3.2 Single-Layer Perceptron(mlp)

The term perceptron is first introduced in 1957 by Frank Rosenblatt in his research in Cornell Aeronautical Laboratory (ROSENBLATT, 1957). He mentions the growing attention on the possibility of creating a device with human-like functions like perception, recognition, concept formation and the ability to generalize from experience. He explains the requirements to construct such a system and calls this system which operates according to his principles a perceptron.

A single-layer perceptron is a network of n number of artificial neurons in parallel. Each artificial neuron in the layer produces one network output and it is often linked to all the external inputs. Each input is given a weight and a sum is calculated by taking into account this weight variable. The output node has a threshold value t. If summed input is greater or equal to t, then the output y is 1, otherwise the output y is 0. The inputs are separated into 2 categories which involve the inputs that return 1 as output and the inputs that return 0 as output. It then draws a line, if the points are linearly separable, to separate the points that cause a fire and the points that do not cause a fire.

SLP is generally used to solve relatively basic problems and it is fast to build and run with low computational complexity. It cannot be used for data which is not linearly separable and this raises the famous XOR function problem in SLP. Since XOR function outputs cannot be represented on a 1 dimension surface, it is not possible to solve the XOR problem with SLP. This explains the limitation of the algorithm.

2.3.3.3 Multi-Layer Perceptron(mlpML)

Multi-layer Perceptron is a nonparametric technique which uses backpropagation procedure in estimation tasks (Werbos, 1974). Backpropagation is a gradient descent-based procedure to calculate the weights in a multilayer feedforward neural network (Rumelhart et al., 1988). A feedforward neural network contains multiple neurons which are arranged in layers. Nodes are connected to each other and all the connections have an assigned weight. There are three types of nodes which are input nodes that bring information to the network, hidden nodes that have no direct bond with the outside of the network that transfer the input nodes to the output nodes and finally the output nodes that make the computations and display the output as a result. A feedforward network works in only forward direction, it starts from the input nodes, goes through the hidden nodes and ends in the output nodes without performing any loops. But in MLP, the model is trained by using an algorithm called backpropagation. It starts with random weights assigned to each node and for every input in the training data set, the output is observed and compared to the desired output. The output is then propagated back to the previous layer and the weights are then modified according to the error. The process is then repeated until a predetermined value of error is obtained. Backpropagation is sometimes defined as learning from mistakes since it repeats the same process by trying to decrease the error of a certain input.

MLP is used in many domains such as data compression(Blanchet, 1990), speech recognition(Bourlard & Morgan, 1990) and hand-written character recognition(Basu et al., 2005).

2.3.3.4 Long Short-Term Memory(lstm)

Long Short-Term Memory model is introduced by Sepp Hochreiter and Jürgen Schmidhuber in their publication in 1997 entitled ”Long short-term memory” (Hochreiter & Schmidhuber, 1997). He observed that with conventional Back-Propagation Through Time or Real-Time Recurrent Learning, error signals that go backwards in time are inclined to either blow up or vanish. In order to solve this problem, he introduced Long Short-Term Memory(LSTM) as a novel recurrent network architecture.

LSTM is actually a type of recurrent neural network(RNN) but it includes additional units such as a memory cell and forget gates that allow a better control over the gradient flow. Memory cells can hold information for an extended period of time and gates can decide when the information enters the memory and how it is handled after it enters the memory.

LSTM has been used in various researches in different topics such as machine translation(sequence to sequence learning) (Sutskever et al., 2014), image captioning (Vinyals et al., 2014), question answering (Wang & Nyberg, 2015) and hand writing generation (Graves, 2013).

2.3.4 Other Algorithms

This section describes the other models used in this research that do not go into the categories described above.

2.3.4.1 Support Vector Machine(svm)

Support Vector Machine algorithm was introduced at the Computational Learning Theory(COLT) conference in 1992 by Vladimir Vapnik and his colleagues, it includes algorithms shaped by the usage of kernels and it is preferred for classification and regression tasks. It puts together many features already invented before and forms the maximal margin classifier. These features include, the usage of kernels, hyperplanes, optimization techniques which were used in pattern recognition and usage of slack variables to deal with noise in the data. [6] It maps the input vectors into high dimensional feature space Z by using non-linear mapping. A linear decision surface is then constructed with particular properties in this space which ensure high generalization ability of the network. The main problem in this approach is how to find a separating hyperplane that would generalize well the data which is called the optimal hyperplane. It is a linear decision function that maximizes the two classes' vector margins. To construct these optimized hyperplanes, a small part of the training data is used which is referred to as support vectors (Cortes & Vapnik, 1995). If it is not possible to divide the training data without error, the goal should be to separate the data by minimizing the number of errors. In order to do so, the subset of training errors is excluded from the training set. After the exclusion, it is possible to divide the remaining training set without errors. This is called the soft margin hyperplane.

It is also possible to perform regression with SVM. A m-dimensional feature space is filled with mapped values of input by using some non linear mapping and from this, a linear model is built in the feature space. The estimation quality is given by the loss function called insensitive loss function which was proposed by Vapnik. The regression is performed via linear regression in the high-dimension feature space using the insensitive loss. As mentioned before, SVMs use functions called kernels for regression and classification problems. The task of the kernel function is to transform the input data into the required form for the model. There are many types of kernel functions like linear, nonlinear, polynomial, radial basis and sigmoid. In our research we use two different SVM kernel functions which are linear and radial basis. Linear kernel is generally used for linear data and nonlinear kernels such as radial basis are used for nonlinear data.

SVMs are used in many different domains and they have relatively higher performance in text classification problems. If an appropriate kernel function is found, it is possible to solve highly complex problems with SVMs but unfortunately it takes a lot of effort to find a suitable kernel function for the model. While SVMs scale relatively well to high dimensional data, they suffer from long training time for large data sets and the interpretation of the final model is complicated. The real life application of SVMs include handwriting recognition, breast cancer diagnosis and intrusion detection.

2.3.4.2 k-Nearest Neighbor(knn)

The nearest neighbor algorithm is a basic nonparametric decisions procedure that is generally used to determine the class of unclassified test samples by assigning them the class of the nearest sample in the training set (Cover & Hart, 1967). If we have an unclassified x and the number of neighbors parameter k is 1(1-NN), then the rule decides that x should have the class of the nearest neighbor. If the number of neighbors increase, then the algorithm finds the closest k neighbors and decides which class the test variable should be assigned to. When the neighbor parameter is very large the effect of the distance factor decreases significantly. On the other hand, if k is too small, classification will be highly affected by the outliers. So the main problematic of this algorithm is to decide the k parameter. Three different distance functions are used to calculate the distance between data points: Euclidean, Manhattan and Minkowski. In case of categorical variables, Hamming distance is used.

$$EuclideanDistance = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{2.3}$$

$$ManhattanDistance = \sum_{i=1}^{k} \mid x_i - y_i \mid \tag{2.4}$$

$$MinkowskiDistance = (\sum_{i=1}^{k} (\mid x_i - y_i \mid)^q)^{\frac{1}{q}} \tag{2.5}$$

While k-NN is generally used for classification, it can also be used for regression tasks (Hastie & Tibshirani, 1996). It calculates the average of the numerical target of the K nearest neighbors. For example, if a graph shows the height in centimeters and the age of a person in years and we are trying to find the age of a test point, k-NN simply finds out the closest points to the test point and assumes that they should have a similar age as the test point. This is called feature similarity.

k-NN is simple to use and implement. It gives relatively good results in classification tasks if the k value is chosen correctly but computational complexity may be high depending on which distance measure is used in the algorithm. It can be used for regression, but it makes predictions in the interval of the training points. So it may not be possible for k-NN to make an accurate forecast if the data in the future exceeds the limits of the training set.

2.4 Evaluation

In this section we describe how we run our experiments and how we evaluate the models. As mentioned before, we do not optimize the parameters for each model, CARET package helps us doing this but instead we use different training and testing methods to observe their effect on the result.

Firstly, we use statistical analysis technique Analysis of Variance(ANOVA) to observe the effect of each experimental parameter that we explained before. ANOVA is a method used in statistical experiments when there is two or more samples to test if these samples belong to the same population. It has two main parameters which are between group variation(1) which is the variation that occurs because of the interaction between the samples and within group variation(2) which is the variation that appears because of the individual differences within samples. The results of the ANOVA test is evaluated by the F test statistic which is calculated by dividing between group variance to the within group variance. If the p value for the F test is less than 0.05 the samples do not belong to the sample population, thus statistically the results are different. In our research, sample is the set of results that are accumulated for a specific parameter.

These parameters include changing the training/testing ratio each time, training in regular order, training in random order, testing on test data, testing on training data, using different forecasting algorithms and running the experiment on different data sets. We use ANOVA test to examine the statistical influence of these parameters and report them in Chapter 3.

Secondly, we draw plots comparing the error metrics for algorithms depending on the above techniques. We point out the differences on each plot and talk about possible reasons for each case such as overfitting, underfitting and high error percentage.

2.5 Comparison Metrics

In the literature for evaluating the accuracy of the methods, there are various performance evaluation metrics. In this study, for the evaluation metrics we opted for Mean Absolute Error(MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), R Squared Error(R^2) and Execution Time.

2.5.1 Mean Absolute Error(MAE)

The mean absolute error(MAE) is the average of all absolute errors which can be calculated by finding the absolute error for each prediction, adding them up and then dividing the result to the number of predictions. It is relatively simpler to calculate than the other error metrics and shows the average difference between the predicted value and the actual value. The equation of MAE is given in the equation below.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid (y_i^{'} - y_i) \mid \tag{2.6}$$

2.5.2 Mean Squared Error(MSE)

Mean Squared Error is calculated by squaring the average of the errors which is, the average squared difference between the estimated values and actual values. It is a measure of how close a fitted line is to data points. Since the errors are squared, large errors are penalized in this error metric, so it is beneficent when detecting large errors is a priority. The equation of MSE is given in the equation below.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i^{'} - y_i)^2 \tag{2.7}$$

2.5.3 Root Mean Square Error(RMSE)

Root Mean Square Error is calculated by finding the standard deviation of the prediction errors(referred as residuals). Residuals are a measure that shows the distance of the data points from the regression line. RMSE is a measure of how spread out these residuals are. It presents how concentrated the data is around the line of best fit. RMSE is generally used in forecasting, climatology and regression analysis to check experimental results. RMSE gives high weight to large errors because the errors are squared before they are averaged, thus it is convenient to use RMSE to detect large errors in forecasting. The equation of RMSE is given in the equation below.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i' - y_i)^2}{n}} \tag{2.8}$$

2.5.4 R-Squared Error(R^2)

R-Squared Error(R^2) is a statistical error metric that shows the proportion of the variance for a dependent variable that is represented by one or more independent variables in a regression model. So, it shows the representation strength of one variable variance to another variable variance. If the value of R^2 is close to 1, it shows that almost all observed variation can be explained by the model's inputs. If the value of R^2 is negative, then it is preferred to use the mean value of the data set as a prediction instead of running the forecasting model. The equation to calculate R^2 is given below.

$$R^2 = 1 - \frac{ExplainedVariation}{TotalVariation} \tag{2.9}$$

2.5.5 Execution Time

Execution time is the last metric for our experiments. It indicates the time passed in seconds to obtain the result. It also stands for the performance of the chosen method and shows the time complexity of the algorithm for the particular data set.

3 EXPERIMENTS AND RESULTS

In this section, we introduce the parameter values that we have used in our experiments and we interpret the results obtained from our tests according to our error metrics.

Our research consists of performing, evaluating and comparing forecast tasks on 5 data sets from different domains which are stock markets, cryptocurrency and weather. We evaluate 17 algorithms' performance coming from 4 categories in the perspective of 5 error metrics. We observe the change in the results as the model building parameters; data ratio, prediction process type and data selection type.

We define a ratio variable that decides how to partition the data into training and testing data. The ratio variable starts from 0.15 which indicates 15% of the data used as training. The values that follow are 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 0.99 for the ratio variable. We investigate the effect of different ratios in training and testing and our results demonstrate the ideal value for the ratio variable for each data set.

After deciding on the ratio variable, we split the data into train and test sets. These are two different values of prediction process type parameter. We then consider the last model building parameter; data selection type. We follow two different approaches in this part. The first method is sampling the data randomly without order to form the training data. In this case, we investigate the importance of data order when forming the training data and observe if this approach has lower error rate than an ordered approach. We expect to have varying results for different data sets according to their type and form. Especially if the value interval of the data set is large, randomization would help the algorithm to expand the forecasting interval, giving more accurate results. Also it may help to avoid overfitting.

The second data selection method we use to form the training data is the regular method that takes the predefined amount of the data as training data in an ordered fashion without randomization. We expect to have lower error rates than randomization in certain data sets but higher error rates on data sets with large value intervals. Especially if the percentage ratio for the training data is low, then very little amount of the data will be used as training data and the forecasting interval will be very narrow, thus ending with worse forecasting results for the data sets, not to mention the high overfitting possibility.

After partitioning the data into training and test data, we run our forecasting models on both training data and test data to compare their performance. First, we perform forecasting on the training data and expect to have lower error rates since the training data is already introduced to the algorithm in the training part and the interval of the training data is the same as the forecasting data in this case.

Second, we perform forecasting on the test data and expect to have higher error rates than testing on the training data since this unseen data by the algorithm may have output values which are outside the interval of the training data so the algorithm cannot predict them which lowers the accuracy percentage.

In this section, we evaluate the results both statistically and empirically. The details of statistical method are introduced before. In this section, we first give the results of ANOVA tests and interpret them. Then, we will evaluate the graphics of error metric results and compare the algorithms' performance.

3.1 Statistical Analysis Evaluation

In this section, we declare the result of the statistical analysis performed to check the effect of the various parameters used in this experimental research to the result. Then, we evaluate these findings. As statistical analysis, we performed analysis of variance(ANOVA) test, multivariate analysis of variance(MANOVA) and Tukey post-hoc test on all the error metrics. Table 3.1 shows the results of the ANOVA test on RMSE error metric. We did not share all the test results because they all point out the same features as the most effective. According to one-way ANOVA with a single factor and a single response variable, we can observe that test type

Table 3.1: ANOVA Results

	Df	Sum Square	Mean Square	F value	Pr(>F)
test_type	3	8.444e+06	2814537	**2.703**	**0.0441**
data_name	3	2.307e+06	768939	0.738	0.5290
ratios	9	1.805e+06	200557	0.193	0.9950
algorithm	14	2.047e+08	14623282	**14.044**	**<2e-16**
test_type:data_name	9	2.057e+06	228531	0.219	0.9918
test_type:ratios	27	5.836e+06	216147	0.208	1
data_name:ratios	21	3.259e+06	155188	0.149	1
test_type:algorithm	34	1.025e+07	301344	0.289	1
data_name:algorithm	13	9.574e+05	73648	0.071	1
test_type:data_name:ratios	26	2.525e+06	97125	0.093	1

and algorithm have the highest F values and lowest p values which leads us to observe that they have more significance over the result with a low risk of error. We also performed two-way and three-way ANOVA to observe the effect of multiple factors to the result. The factor pairs we used for these were test type-data name, test type-ratios, data name-ratios, test type-algorithm, data name-algorithm for two-way and test type-data name-ratios for three-way ANOVA. As a result, we did not observe a high value of F and the p value was very close to 1.

In order to expand our findings, we used MANOVA on the same factors by using the pair groups we had in ANOVA as response variables. The results are displayed in Table 3.2. We can observe that the results are similar to ANOVA with F values high for prediction process type and algorithm. But we can also see that data name has a relatively high F value and a low p value. This shows that data name is also influential like test type and algorithm.

After declaring the results, we look into the hypotheses stated in Chapter 2. We can observe that since the ratio parameter was not influential to the result according to statistical analysis as we expected in our first hypothesis, we can tell that this hypothesis turned out to be false. This might be caused by the high bias forecasting algorithms used in the research which do not tend to overfit but can underfit training data in some cases which makes it harder to catch essential regularities. We find out that the prediction process type has an effect to the result but we do not know if it

Table 3.2: MANOVA Results

	Df	Pillai approx.	F num	Df2	Pr(>F)
test_type	3	0.06762	**9.2121**	18	**<2.2e-16**
data_name	3	0.01573	**2.1059**	18	**0.004056**
ratios	9	0.02778	1.2402	54	0.111148
algorithm	14	0.44461	**13.7198**	84	**<2.2e-16**
test_type:data_name	9	0.02432	1.0855	54	0.310394
test_type:ratios	27	0.04304	0.6423	162	0.999873
data_name:ratios	21	0.03053	0.5845	126	0.999943
test_type:algorithm	34	0.10936	1.3105	204	0.002071
data_name:algorithm	13	0.01204	0.3711	78	1
test_type:data_name:ratios	26	0.03025	0.4677	156	1

has a positive or negative effect. Since our hypothesis states that training randomly increases the forecasting accuracy, we can not say for sure that hypothesis 3 is true or false here. Indeed in the next part, when evaluating numerical results of error metrics we will have more idea about which prediction process type is causing higher error, as a result we will also have more idea about overfitting algorithms.

While these results give us an idea about the influence of our experiments to the result, they do not highlight which test types or algorithms have a higher effect on the result. In order to point out the mostly effective test type, algorithm or data name, we perform post-hoc tests. Post-hoc tests are used to find where the differences occurred between groups. Although there are many post-hoc tests to choose from, it is essential to test with only one. We picked Tukey post-hoc test to compare the differences between groups. We tested prediction process type together with the data selection type. We observed that TrainRegular and TestRandom showed significant difference which points out that changing process type and data selection effects the results. Tukey test results for RMSE are given in Table 3.3. Tukey results for every algorithm pair would take a lot of space so we do not share them but we interpret the algorithms on the error graphs in the following sections.

Table 3.3: Tukey post-hoc Test Results

	diff	lwr	upr	p
TestRegular-TestRandom	30.6265273	-116.023240	177.2763	0.9500362
TrainRandom-TestRandom	30.4294771	-116.220290	177.0792	0.9509279
TrainRegular-TestRandom	149.8217705	3.172004	296.4715	**0.0430776**
TrainRandom-TestRegular	-0.1970502	-146.846817	146.4527	1
TrainRegular-TestRegular	119.1952432	-27.454524	265.8450	0.1567558
TrainRegular-TrainRandom	119.3922934	-27.257473	266.0421	0.1556243

3.2 Metric Evaluation

In this section, we evaluate the error metrics we used, in terms of how helpful they are in comparing the accuracy of different algorithms.

The error metrics used in this research and their properties are explained in detail in Chapter 2. Among them, only R squared error metric has the same value interval with a maximum value of 1 for all data which indicates that the model is a good fit for the data. On the other hand, a negative R squared error indicates that using the mean value as prediction generally would be more efficient than using the forecasting model. Since the value interval is defined, using R squared error makes it easy to compare the influence of different data on the forecasting accuracy. In the figures 3.1, 3.2, 3.3, 3.4 R Squared value of different data with same data selection and prediction process type and training/test ratio are given.

It is easy to compare the performance of models between data even if the predicted feature's value intervals are different for each data. The graphs show that almost all algorithms work well for the ethereum data except LSTM. For bitcoin price data, LSTM has a similar performance, SLP and MLP have around 0.6 ratio which is low compared to the other algorithms. For both the Madrid weather and Hungary weather data, we observe low performance from neural networks and LSTM algorithms.

The other error metrics, however, have different results for each data since they take the absolute error as basis which is the absolute value of the difference of the output and the predicted value. In this case, it is relatively harder to make an

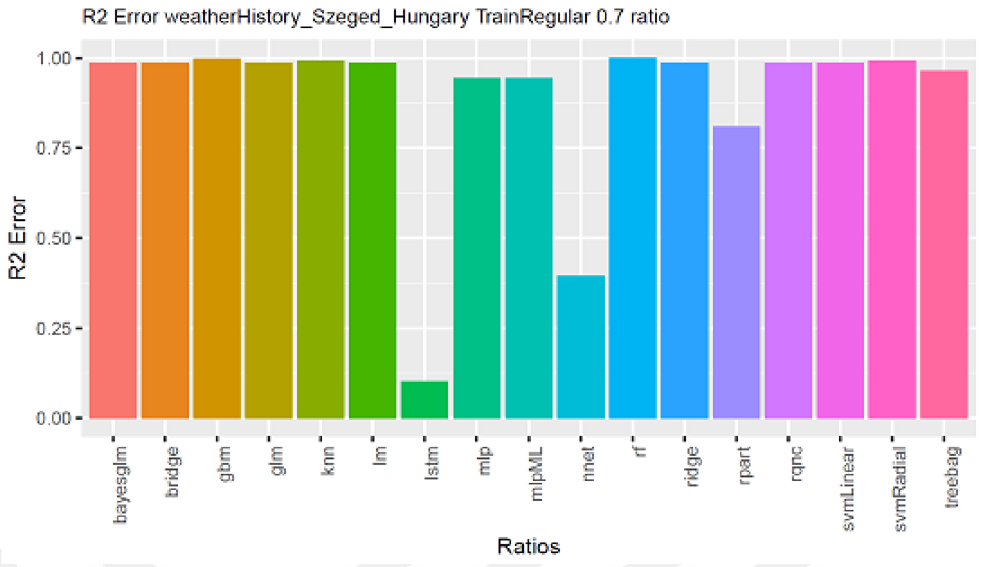

Figure 3.1: R Squared Values of Hungary Weather Data with Train Regular and 0.7 Training Ratio

evaluation since the interval ranges from 0 to infinity. In general, there are some cases to consider when assessing the performance of the algorithms with these error metrics. For example, the predicted feature's value interval. If the value interval is large, then a high RMSE value does not always mean an inaccurate result. In the figures 3.5, 3.6, 3.7, 3.8 RMSE value of different data with same data selection and prediction process type and training/test ratio is given.

We observe that the RMSE value interval changes from 0 to 300 for ethereum data, 0 to 20 for Madrid weather data, 0 to 15 for Hungary weather data and 0 to 6000 for XU100 data. So it is not accurate to make the comparison of the same model between different data just by looking at this graph. Same problem stands for mean squared error and mean absolute error. Though it is convenient to make a comparison between models on the same data. On these graphs, we observe a high error ratio from LSTM in Ethereum, Madrid weather and Hungary weather data. In XU100 data, LSTM has an average performance and CART has the worst performance. We can state that linear models have a low error ratio in general.

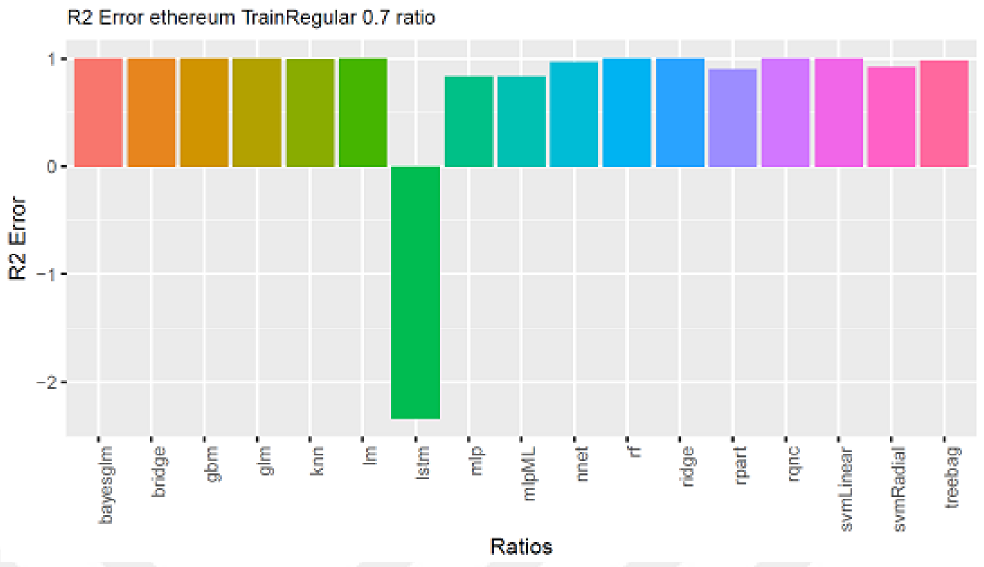

Figure 3.2: R Squared Values of Ethereum Data with Train Regular and 0.7 Training Ratio

3.3 Data Comparison and Evaluation

In this section, we observe using different data's influence on the performance of the models. We have data from three different domains which are stock markets, cryptocurrency and weather.

While not being totally independent of the daily events such as natural disasters and political speeches, stock market and cryptocurrency data tend to follow a trend according to certain technical indicators. However, stock market prices are also open to sudden and unexpected changes. The fluctuations in the prices do not depend on the month of the year or the season. Plus, the stock market prices do not have an upper limit. It is well possible to exceed the maximum price in the training data which would make forecasting difficult for the models that limit their forecasting values to the value interval in the training data.

Weather data relies more on the month of the year than the stock market data. We may observe similar temperature and wind speed values each year in the same period as the year before. This may make forecasting easier for the models that limit their forecasting interval to the interval of the training data. Figures 3.9, 3.10, 3.11, 3.12, 3.13 show the mean absolute error of models for different data with test prediction process type and random data selection.

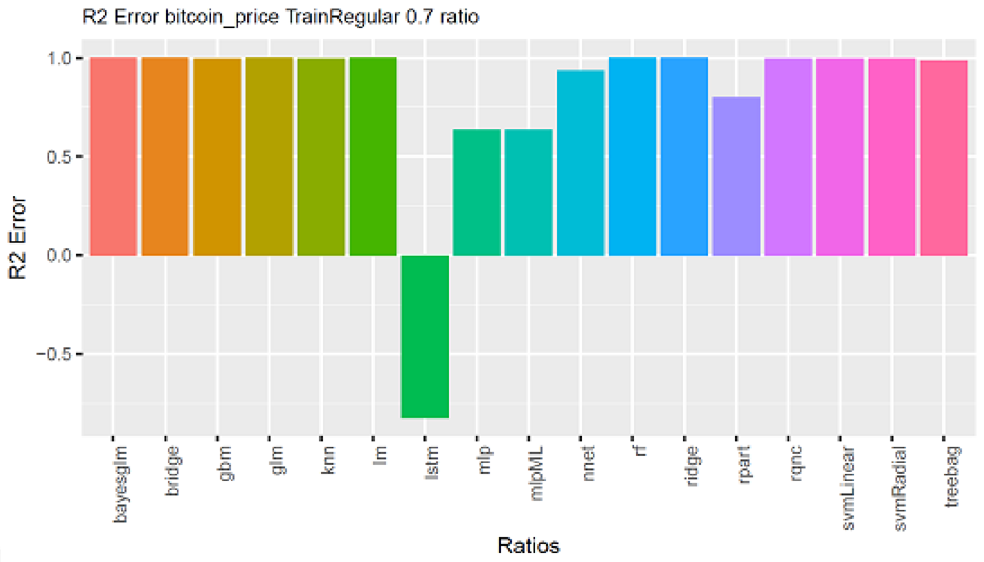

Figure 3.3: R Squared Values of Bitcoin Data with Train Regular and 0.7 Training Ratio

We can observe that linear models have slightly lower error rates with stock market and cryptocurrency data than with weather data. The reason for this might be the price following a trend which is appropriate for linear modelling. Error ratios obtained for SLP and MLP are high for cryptocurrency data, while they are lower for the weather data and even lower for the stock market data. We observe that there is overfitting for SLP and MLP algorithms and training randomly does not decrease the error ratio which disproves the hypothesis 6. On the other hand, using train as prediction process type and regular as data selection type decreases the MAE value significantly. We generally observe in our research that Random Forest algorithm works accurately for all data. Since we use CARET package for building models to make forecasts and leave the parameter tuning to CARET, some models may be underperforming than they should. But Random Forest does not require very fine tuning for its hyper-parameters and they tend to perform well for all data. That may be the reason why it is outperforming many of the forecasting models in this research.

Lastly, we may observe different results for data with different sizes. Our Borsa Istanbul stock market data has 2097, bitcoin price data has 1760, Ethereum data has 928, Hungary weather data has 96453 and finally Madrid weather data has 6812 entries. We can observe that LSTM and Random Forest models perform at their best on the Hungary weather data which is the largest data we have in this

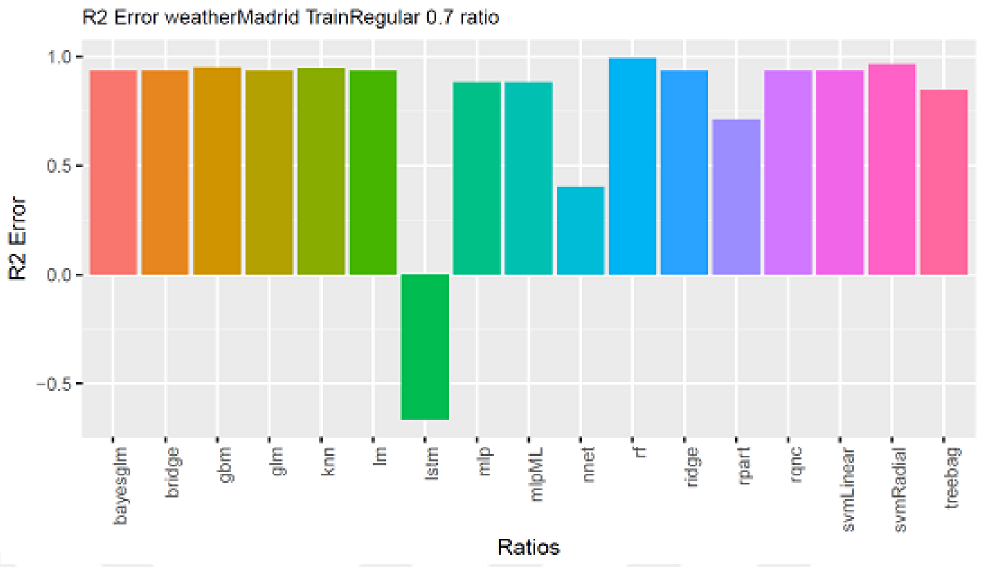

Figure 3.4: R Squared Values of Madrid Weather Data with Train Regular and 0.7 Training Ratio

research. In general, we observe that LSTM has low performance compared to the other models. This appears to be the result of having insufficient size of training data for the model in this research.

3.4 Regular vs Random Training and Testing

In this section, we compare the performance of data selection and prediction process types we used in our research. They include training in regular order and testing on training data(TrainRegular) or testing data(TestRegular), training in random order and testing on training data(TrainRandom) or testing data(TestRandom). These are explained in detail in the methodology chapter of this paper. The figures 3.14, 3.15, 3.16, 3.17 demonstrate the MAE of the algorithms we used on the Madrid weather data. Each graph has the same training/testing ratio but different data selection and prediction process type.

It is possible to observe that TestRandom and TrainRandom have a relatively lower error ratio than TrainRegular and TestRegular which indicates that random data selection decreases the error rate. This disproves the hypothesis 3 since the error rates change after training randomly. While this may be the case for the Madrid weather data, we should also investigate the effect of the data selection and

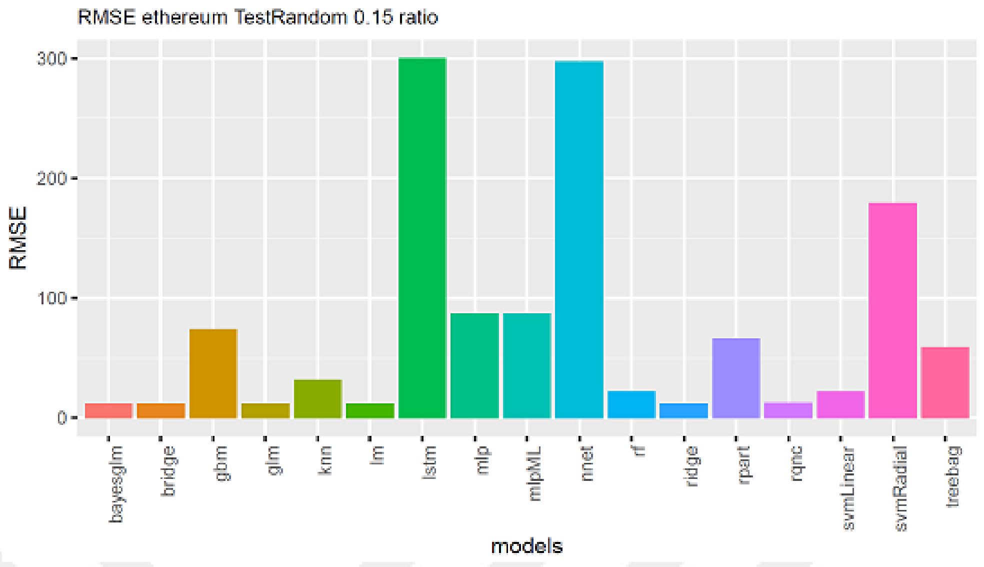

Figure 3.5: RMSE Values of Ethereum Data with Same Data Selection and Prediction Process Type and Training Ratio

prediction process types on the algorithm performance with different data. Madrid weather data has 6812 records while the bitcoin data only has 1760 records. The graphs for bitcoin price data are displayed in figures 3.18, 3.19, 3.20, 3.21.

Firstly, we can observe that linear models perform with a low error ratio for all data selection and prediction process types on the bitcoin price data. The results for LSTM is similar for all data selection and prediction process types. In TestRandom and TrainRandom, SLP and MLP perform similarly but they have higher error rate with TestRegular and lower error rate in TrainRegular. Some models like neural networks, svmRadial, treebag and CART only work well in TrainRegular. We observe that, in general, the best results were obtained with TrainRegular for this data. Results for TrainRandom are similar to TrainRegular for some models but models like SLP, MLP, CART and SVM performed worse in TrainRandom.

The Madrid weather data had very similar results for each data selection and prediction process type. This complies with the hypothesis 3 since the results do not change too much after training randomly. The graphs show that bitcoin price data is more effected from the data selection and prediction process type than the Madrid weather data. In order to understand this better, we investigate this effect in the Hungary weather data in figures 3.22, 3.23, 3.24 and 3.25.

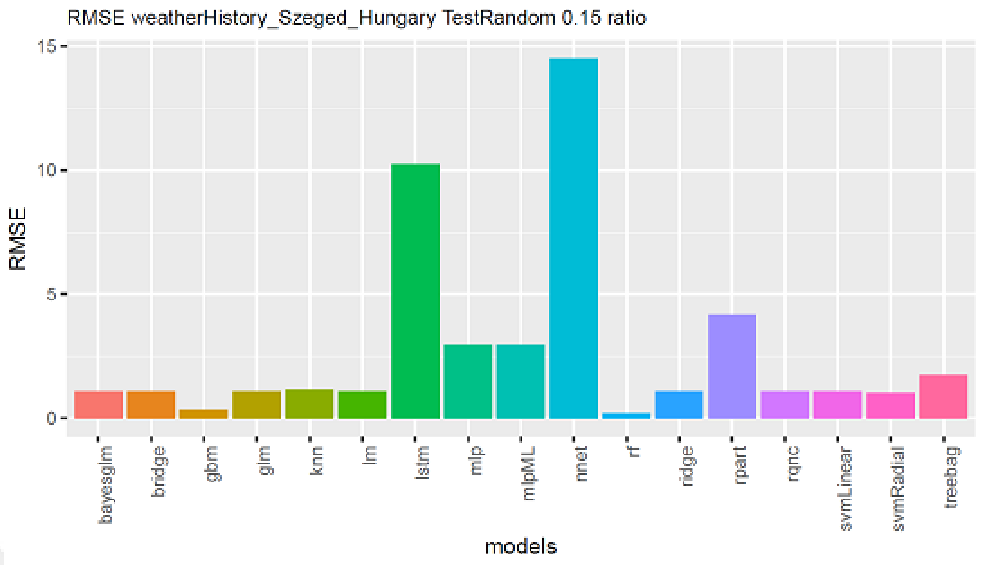

Figure 3.6: RMSE Values of Hungary Weather Data with Same Data Selection and Prediction Process Type and Training Ratio

Hungary weather data has a similar MAE graph to the Madrid weather data for different data selection and prediction process types. We may claim that data selection and prediction process type has a higher influence on the cryptocurrency data than the weather data. In order to make this claim, we draw the same graphs for the ethereum price data. These graphs show similar aspects as the bitcoin price data. This indicates that data influences the performance of the data selection and prediction process type. Also, in our case, data selection and prediction process types have a similar effect on data from same fields like bitcoin price and weather data. Figures 3.26, 3.27, 3.28 and 3.29 show the mean absolute error of models with different data selection and prediction process types on Ethereum price data. We observe that there is overfitting in the cryptocurrency data for models like rpart and SVM since there is a noticeable difference between the error ratios when the forecasting models are tested on the training set and the test set. Testing on the training set gives much lower error ratios. Thus, we can say that hypothesis 2 is correct, if an algorithm overfits, its training and test performances are quite different. We can also see that training randomly for Ethereum data increases the error ratios considerably for models like SLP, MLP and SVM. So, hypothesis 3 and 5 are false for the Ethereum data in this case. Also, hypothesis 4 which expects lower error rates when training randomly for seasonal data is disproved when we compare the TestRandom and TestRegular graphs for the Madrid weather data and Hungary weather data. The results are very close but as opposed to the hypothesis, training

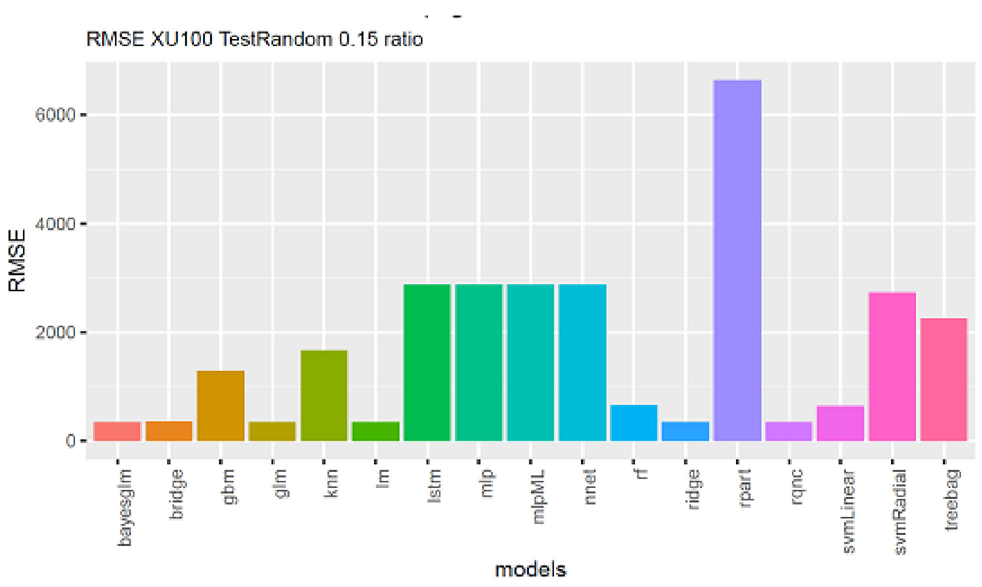

Figure 3.7: RMSE Values of XU100 Data with Same Data Selection and Prediction Process Type and Training Ratio

regularly gives slightly lower error rates. A general observation in the results is that with cryptocurrency data which are very unpredictable, models require more insight of the data to decrease the error. But our cryptocurrency data in this research do not fit well to the models even with random data selection. Thus, random data selection with a very large cryptocurrency data fit to an LSTM model might be a better approach for this type of data to decrease the error rates. Also, since cryptocurrency is a relatively new domain, it may require several more years for it to become more stable.

3.5 Algorithm Comparison and Evaluation

In this section, we compare the performance of the algorithms and look for potential cases of overfitting and underfitting. Overall in our experiments, we observe that linear models are outperforming all the other models in general except for a few cases. Since linear models try to find the best line that minimizes the error rate, they should work with high performance on stable data like weather. Also with stock market data, in our case, we examine a stable uptrend so a high performance from linear models is among expectations. For the cryptocurrency data, there is a sudden change in the price after the year 2017, before 2017, the data looks stable. Since regular data selection cannot include data from 2017 to the training set, it

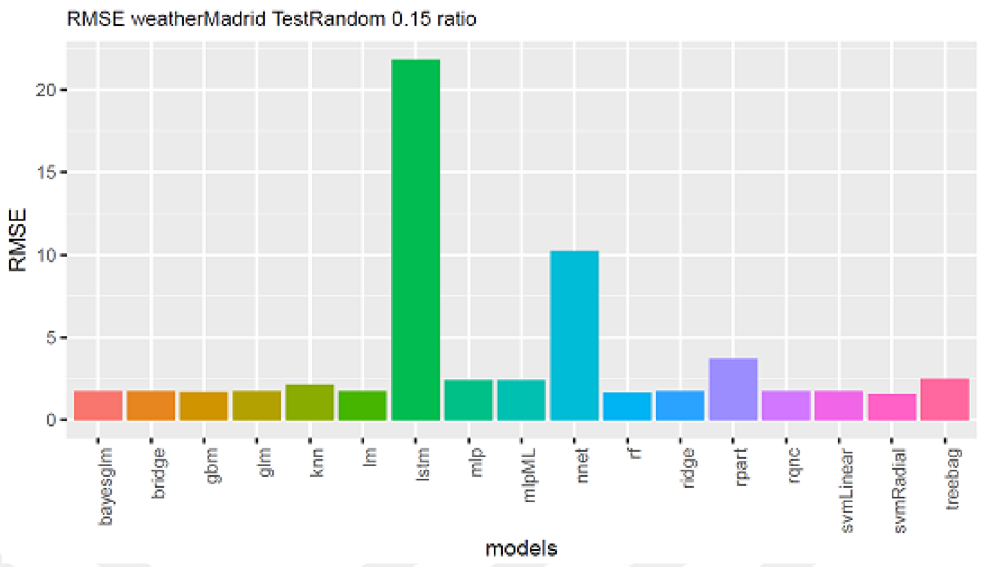

Figure 3.8: RMSE Values of Madrid Weather Data with Same Data Selection and Prediction Process Type and Training Ratio

has low performance with cryptocurrency data compared to random selection. Tree based models have the second best performance. It may be because tree models are generally used for classification tasks. In order to make forecasting, they create branches with value intervals and check to which interval belongs the input and makes the forecasting. But it predicts the same value for multiple inputs if they belong to the same value interval. This causes the error rates to increase. Neural network based models, especially LSTM fail to perform efficiently on our data since the data sizes in our research may not be large enough to train these models enough for them to perform accurately, so they are underfitting. SLP and MLP performed relatively well with TestRandom on Borsa Istanbul stock market data. Knn and SVM models both have acceptable results for almost all cases. Knn is generally used for classification tasks but it can also be used for regression problems. Its predictions are generally based on the mean or the median of the nearest neighbor. Since k value is critical for regression with knn, a better parameter tuning might increase its forecasting performance.

We can observe on figures 3.30 and 3.31 that TestRandom has considerably lower error rates than TestRegular for Borsa Istanbul stock market data for certain models such as knn, gbm, rf and CART. This may be the result of overfitting and shows the benefit of choosing the training set randomly from the whole data which complies with hypothesis 6. A similar case exists for the Ethereum data which is shown on

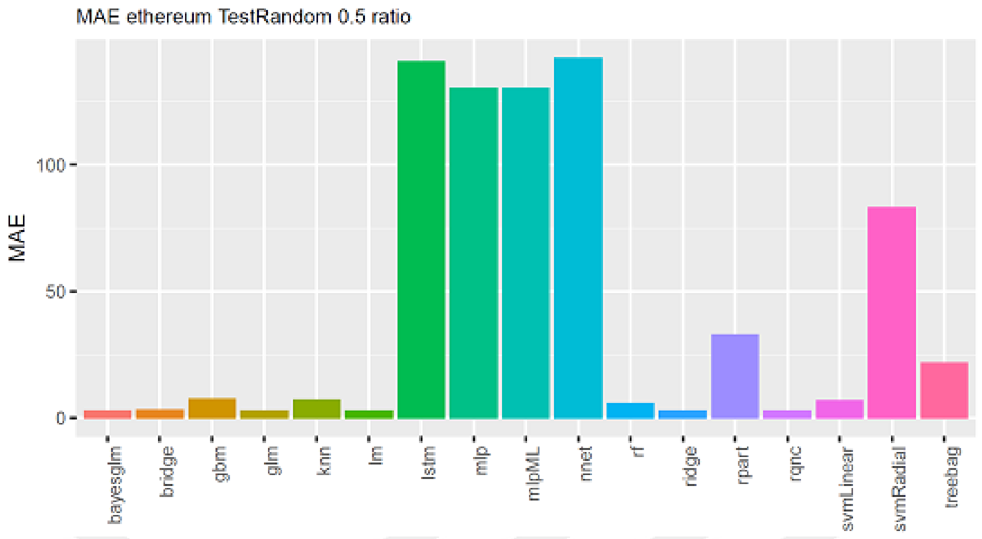

Figure 3.9: MAE of Models for Ethereum Data

figures 3.32 and 3.33 . The affected models are gbm, knn, SLP, MLP, rf, rpart, SVM and CART. This complies with our hypothesis 6 where we expect overfitting to decrease when the model is trained randomly.

We compare the execution time for each model to determine their time complexity. We observe that LSTM takes longer to run than the other models for relatively small data like Ethereum and Madrid weather. But knn, SLP and MLP models' execution time exceeds LSTM's when they are used on a much larger data such as Hungary weather data with 96453 entries. The most time consuming models are LSTM, SLP, MLP, Random Forest, SVM and knn. The least time consuming models are glm, Neural Networks, CART, ridge regression and quantile regression. This shows that although neural networks have low execution times, in general, neural network based models like LSTM, SLP and MLP have higher complexity than the other models which proves the hypothesis 7.

3.6 Summary

This section summarizes the experiments and results part of the research. We highlight the important findings we encountered during the examination of the error rates of the forecasting models.

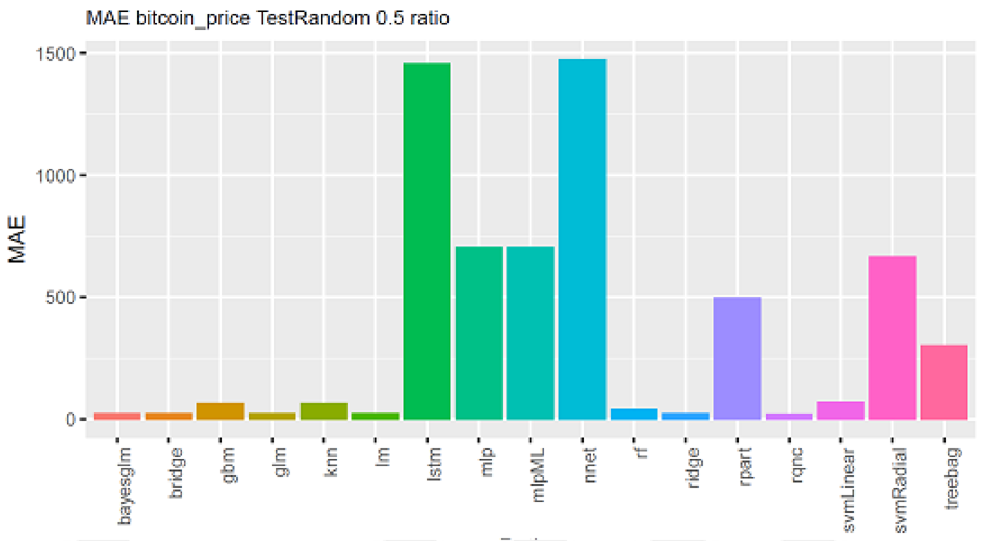

Figure 3.10: MAE of Models for Bitcoin Price Data

1. We saw in statistical analysis that data ratio in our experiments was not highly influential on the result. Statistically, training with different percentage of the data does not change the performance.

2. The results depend mainly on the data, the forecasting model, the prediction process type and the data selection according to statistical analysis.

3. Data selection type(random or regular) and prediction process type(train or test)'s effect depends on the characteristic of the data. For example, using regular order while training gives acceptable results since weather data is generally stable. But, in cryptocurrency data, models require random data selection for training since the target's values change unpredictably.

4. We did not observe a considerable difference between the properties of the error metrics since they all concentrate on the actual and the predicted value to calculate the error rate.

5. We observed that linear models outperformed all the models since they are more accurate in determining an average trend from the data and they try to optimize the error metrics we used in this research. This is the result of a quantitative analysis, this work may be extended with qualitative analysis but it is out of the scope of this research.

6. We saw that neural network based models like LSTM did not perform accurately in our research. This may be caused by the data size we used and testing with larger

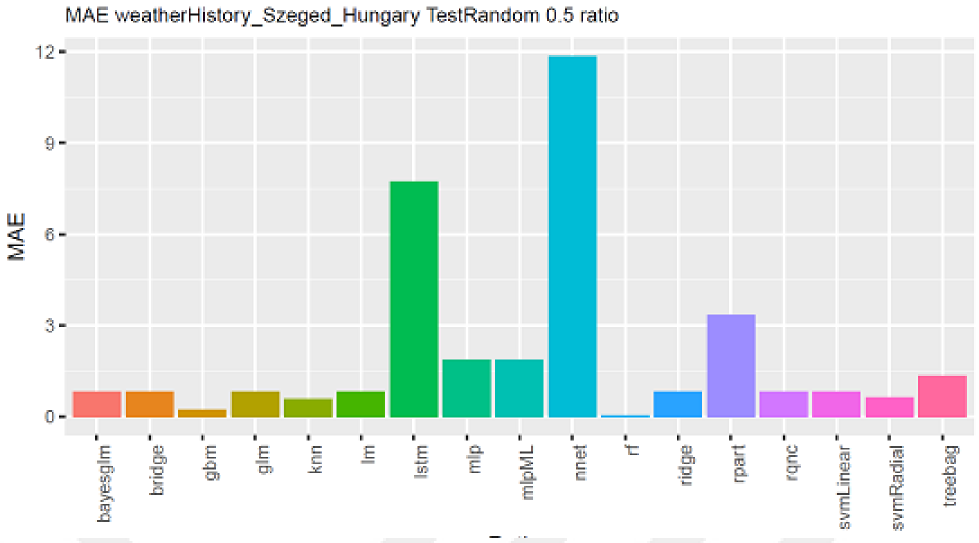

Figure 3.11: MAE of Models for Hungary Weather Data

data may give lower error rates.

7. When we compare the execution times of the algorithms, it is possible to deduce that neural network based models are more complex than the other models. So, they require better infrastructure which increases the costs.

8. As a result, hypotheses 2 and 7 turned out to be correct while the other hypotheses were disproved with examples.

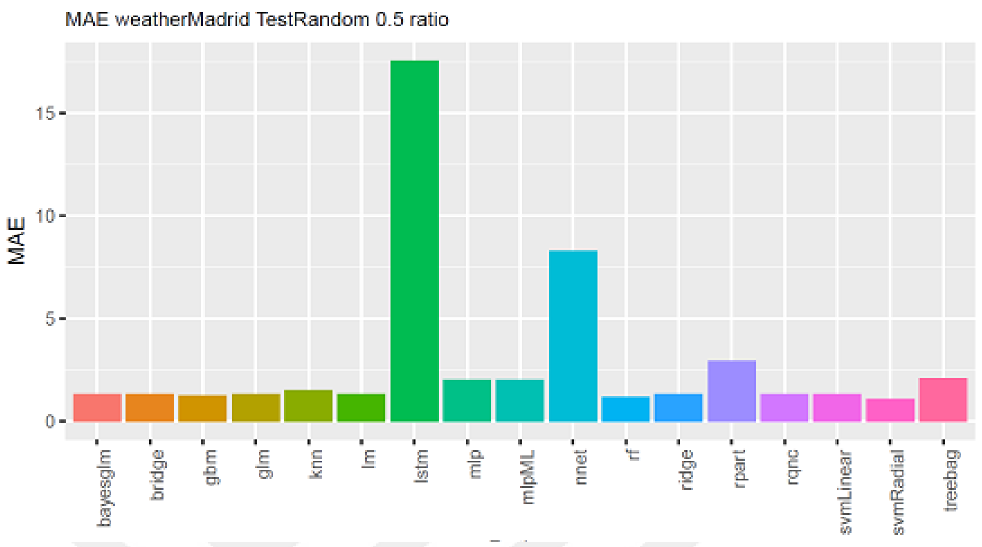

Figure 3.12: MAE of Models for Madrid Weather Data

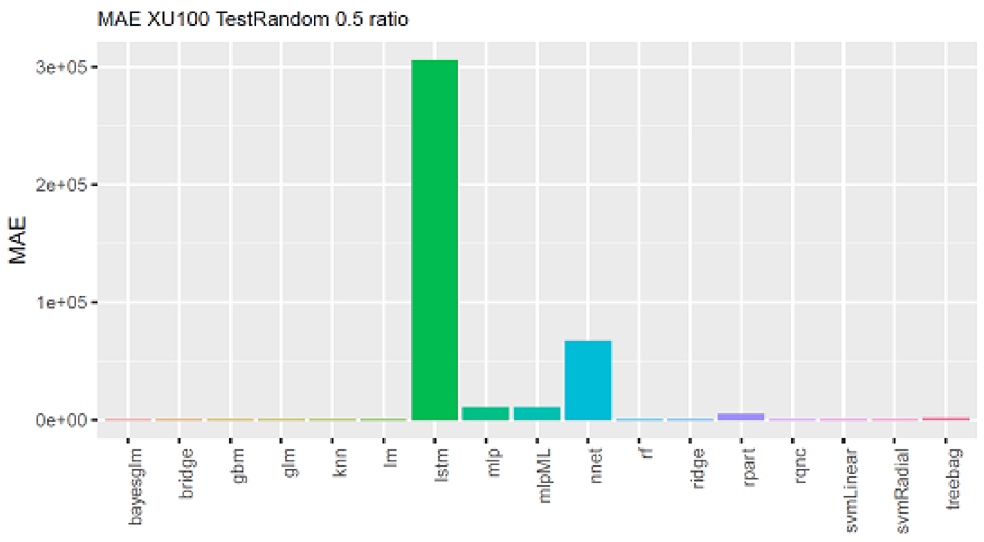

Figure 3.13: MAE of Models for XU100 Data

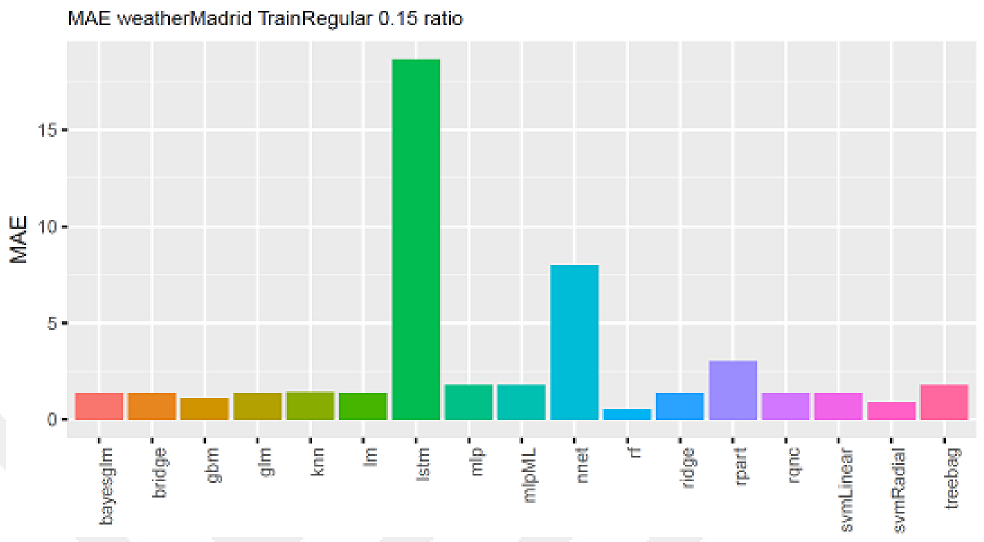

Figure 3.14: MAE of Models for Train Regular on Madrid Weather Data

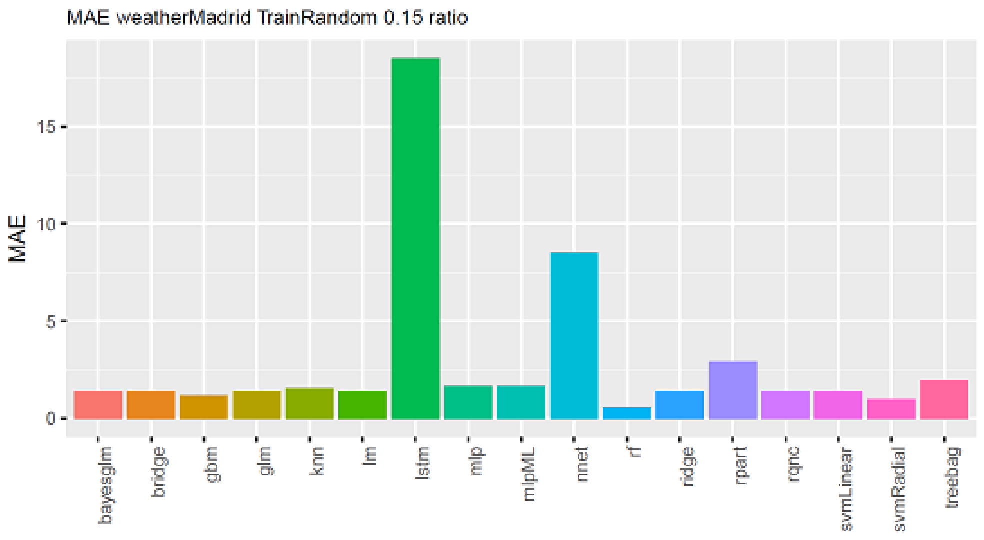

Figure 3.15: MAE of Models for Train Random on Madrid Weather Data

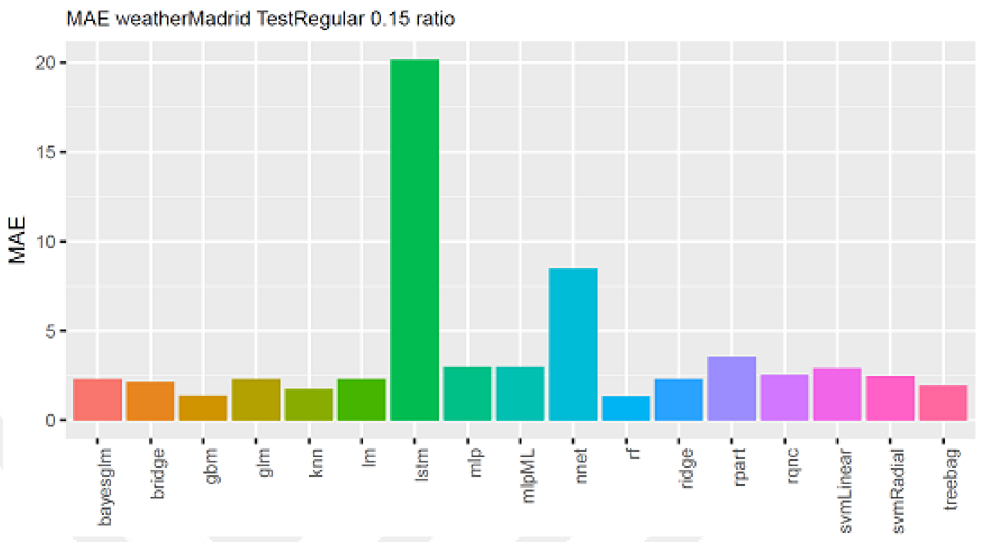

Figure 3.16: MAE of Models for Test Regular on Madrid Weather Data

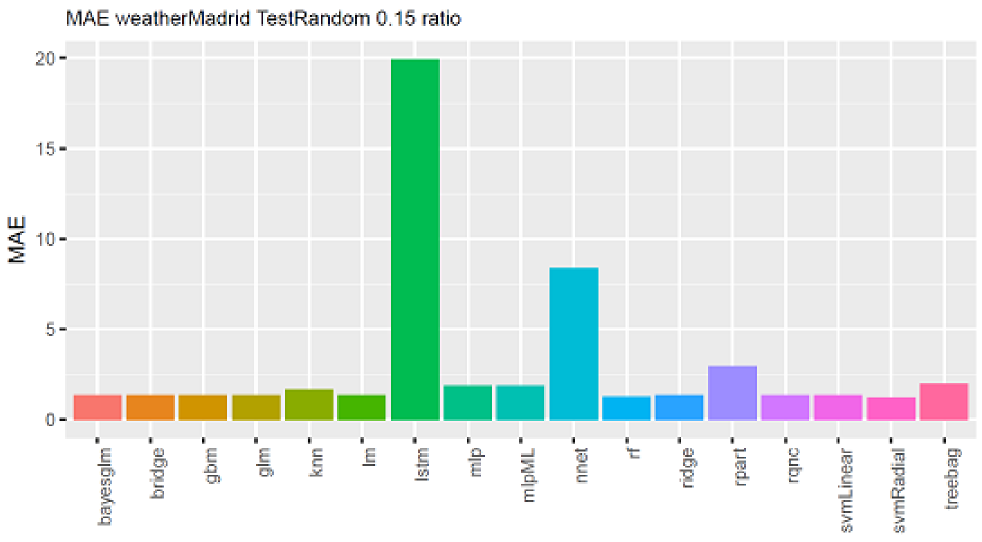

Figure 3.17: MAE of Models for Test Random on Madrid Weather Data

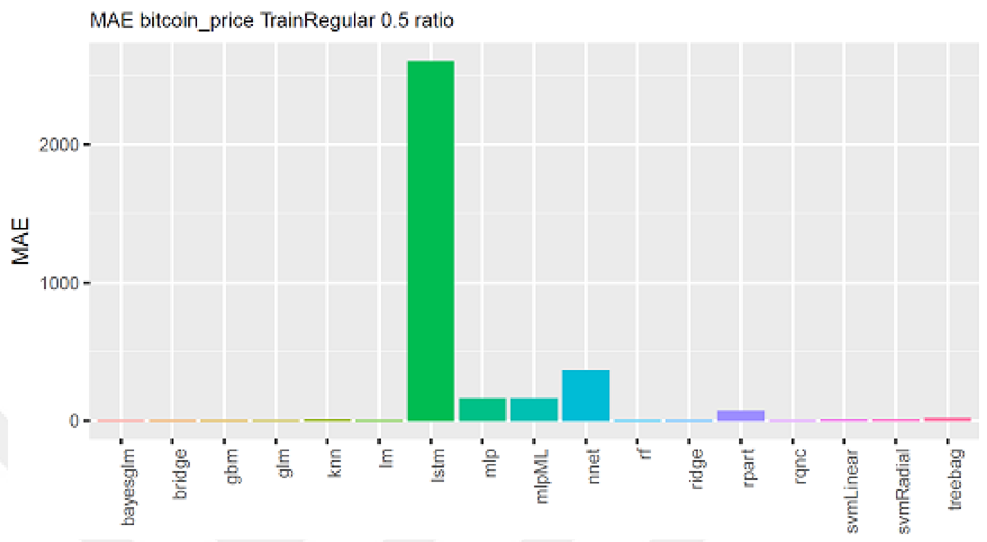

Figure 3.18: MAE of Models for Train Regular on Bitcoin Price Data

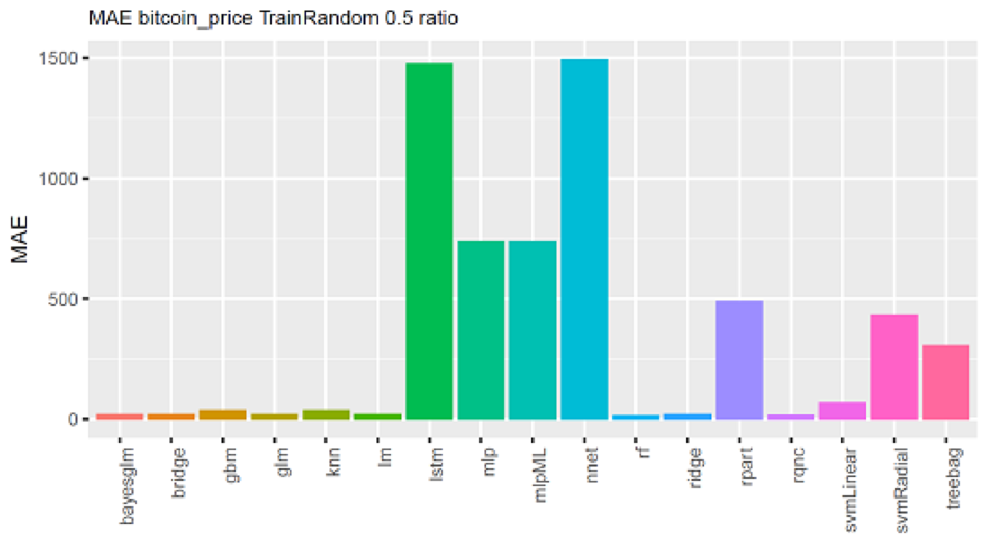

Figure 3.19: MAE of Models for Train Random on Bitcoin Price Data

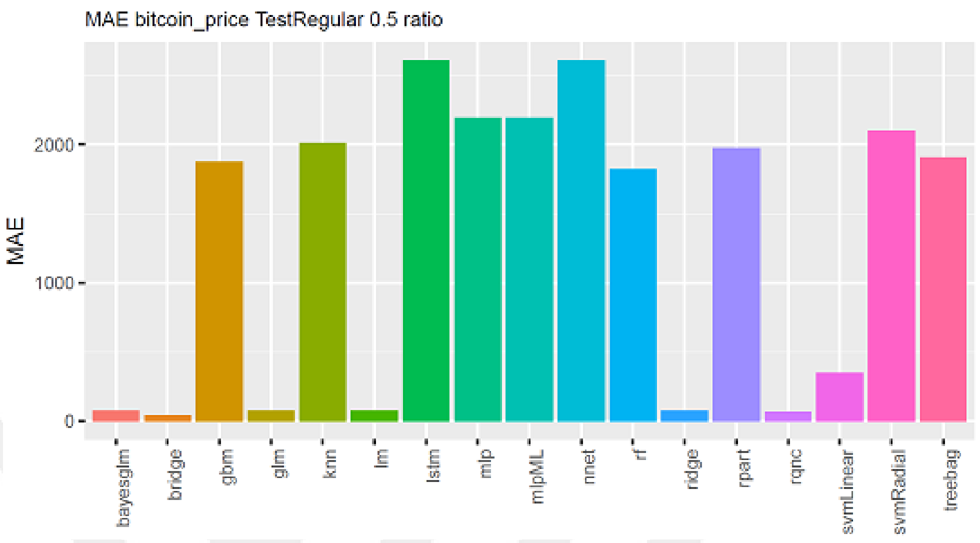

Figure 3.20: MAE of Models for Test Regular on Bitcoin Price Data

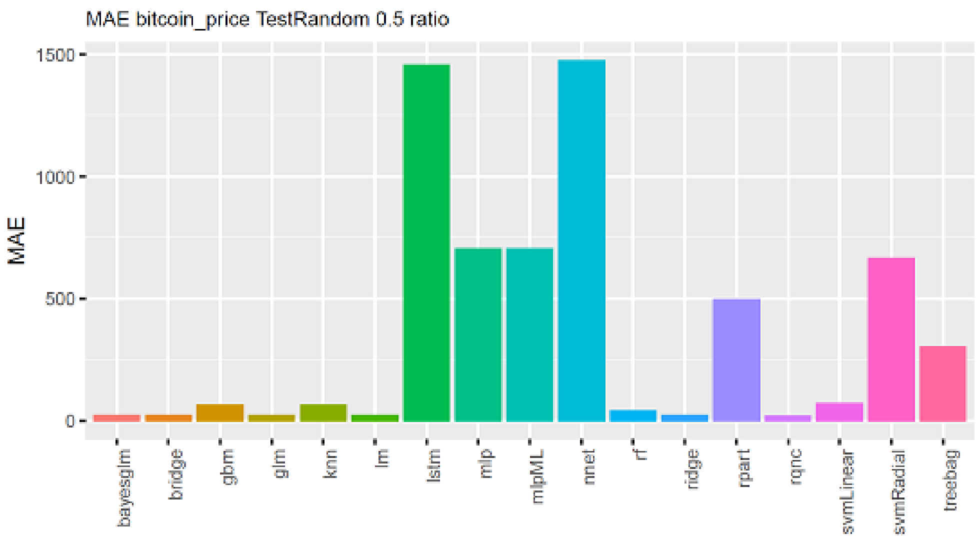

Figure 3.21: MAE of Models for Test Random on Bitcoin Price Data

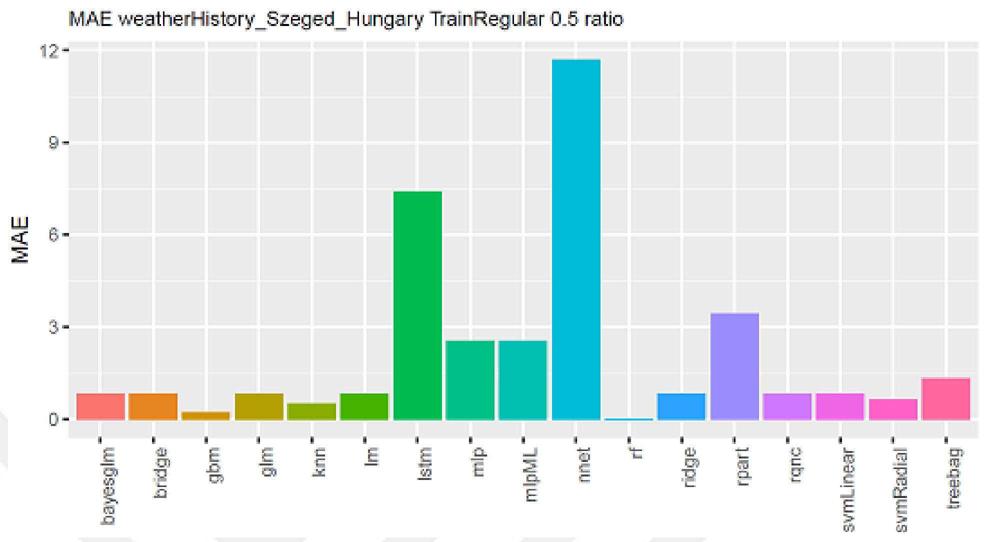

Figure 3.22: MAE of Models for Train Regular on Hungary Weather Data

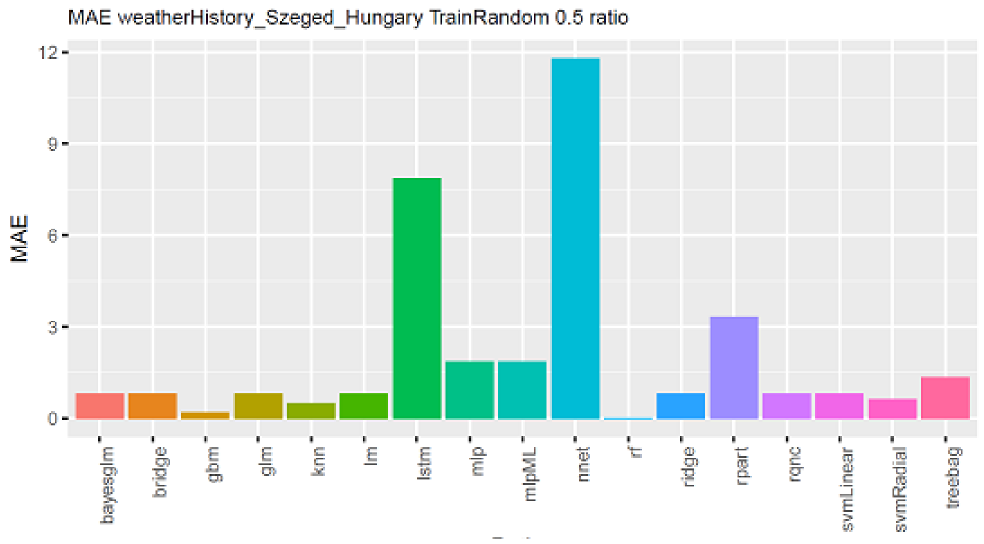

Figure 3.23: MAE of Models for Train Random on Hungary Weather Data

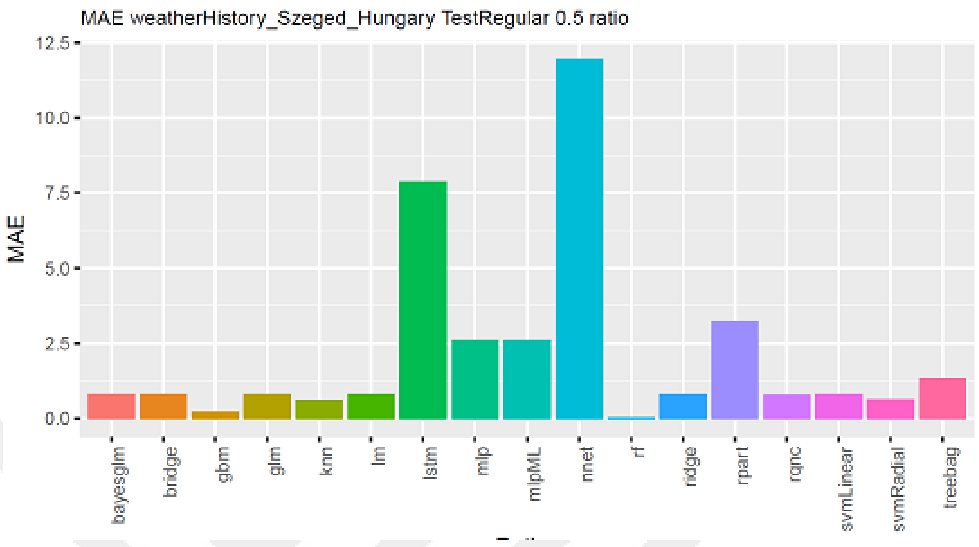

Figure 3.24: MAE of Models for Test Regular on Hungary Weather Data

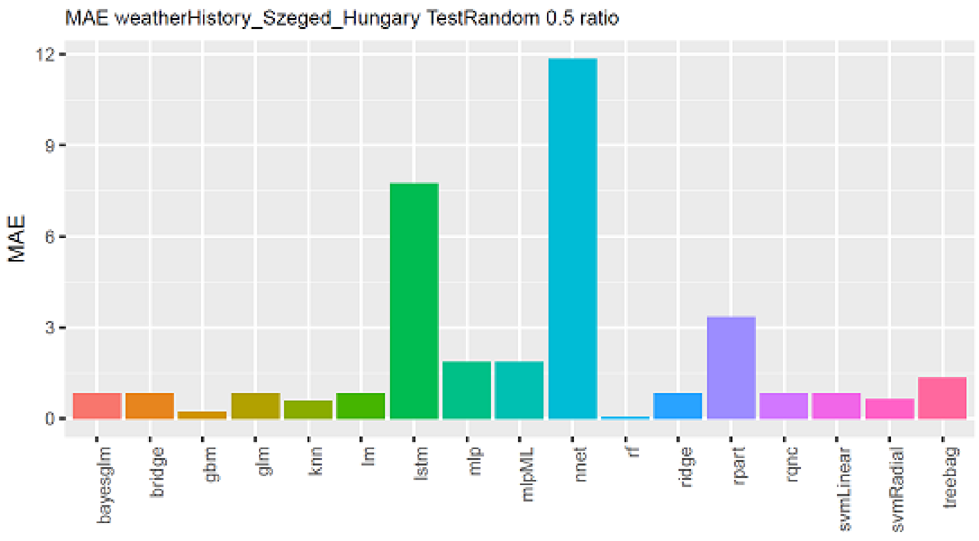

Figure 3.25: MAE of Models for Test Random on Hungary Weather Data

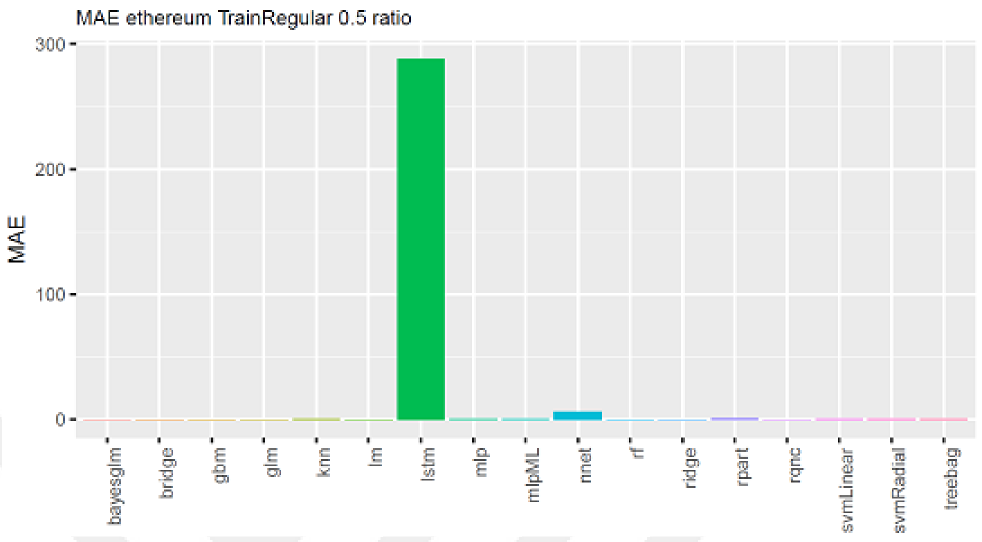

Figure 3.26: MAE of Models for Train Regular on Ethereum Price Data

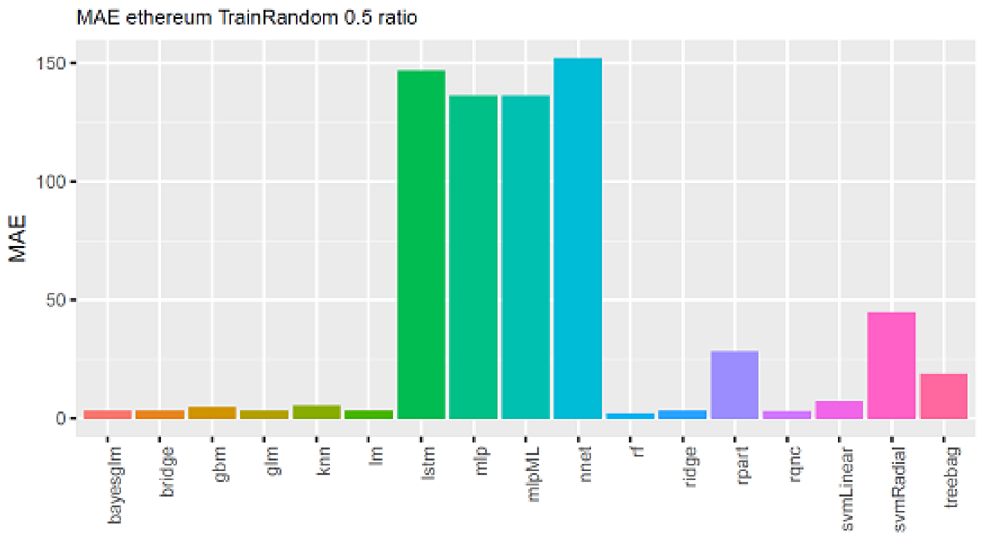

Figure 3.27: MAE of Models for Train Random on Ethereum Price Data

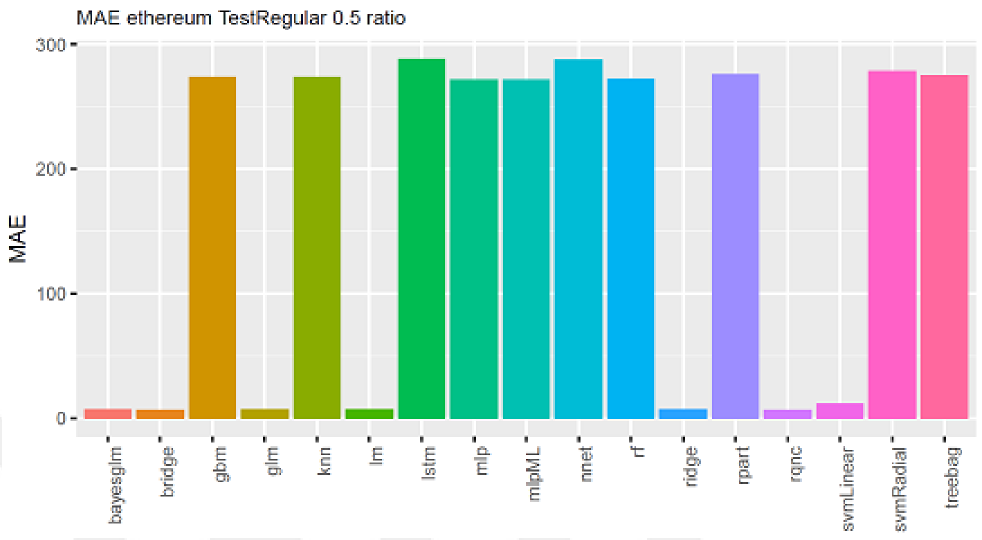

Figure 3.28: MAE of Models for Test Regular on Ethereum Price Data

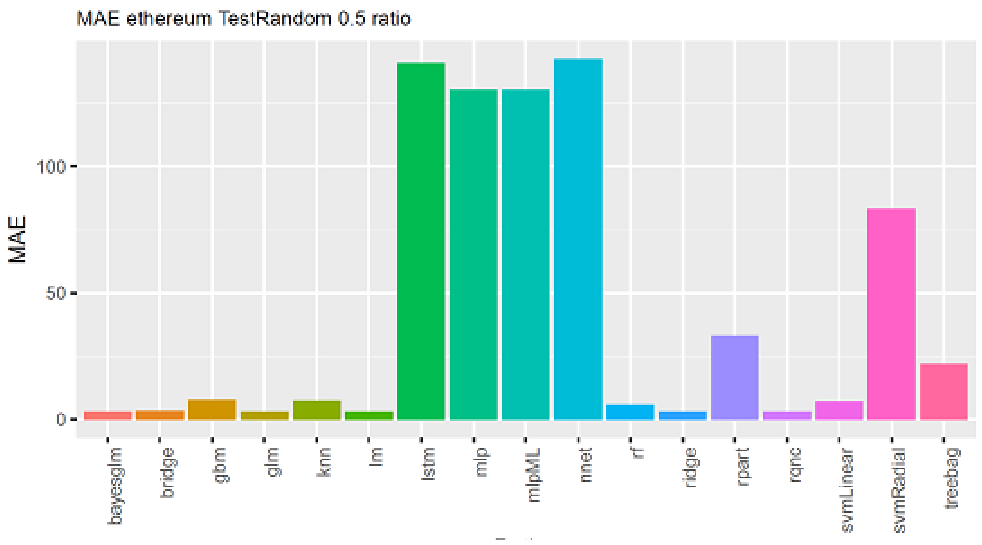

Figure 3.29: MAE of Models for Test Random on Ethereum Price Data

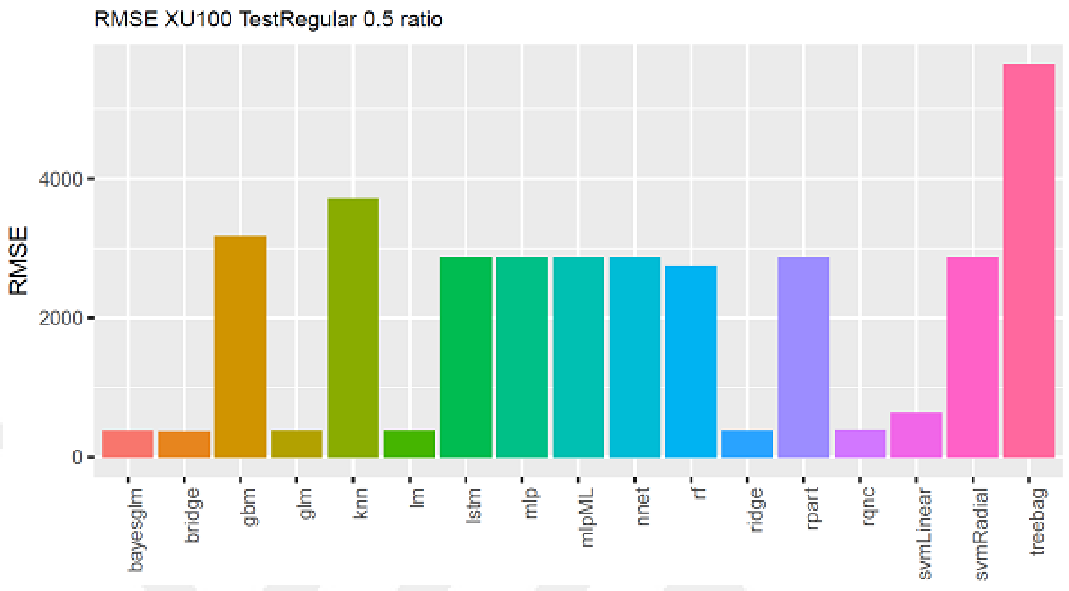

Figure 3.30: TestRegular on Stock Market Data

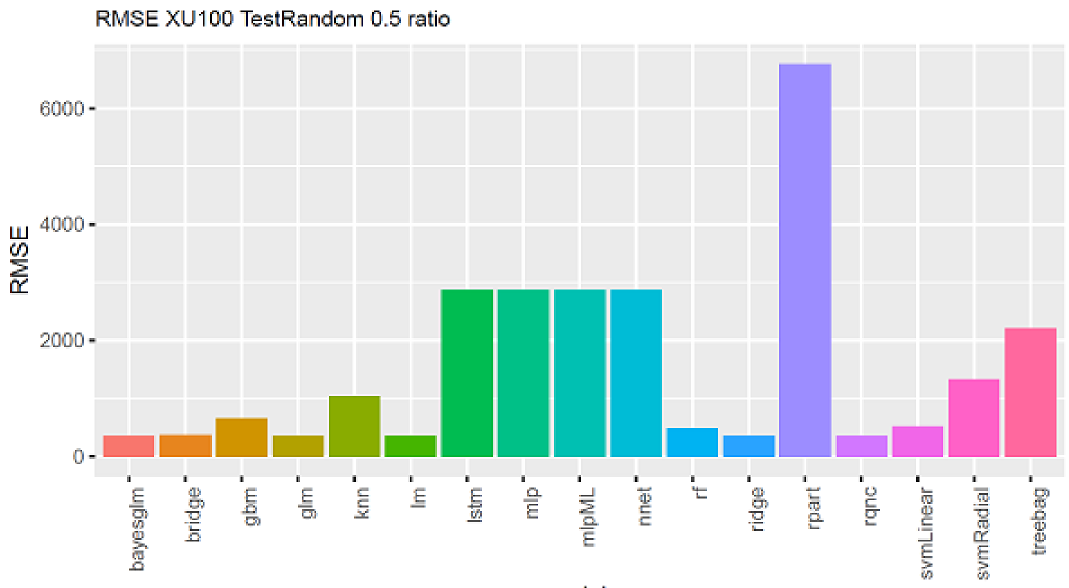

Figure 3.31: TestRandom on Stock Market Data

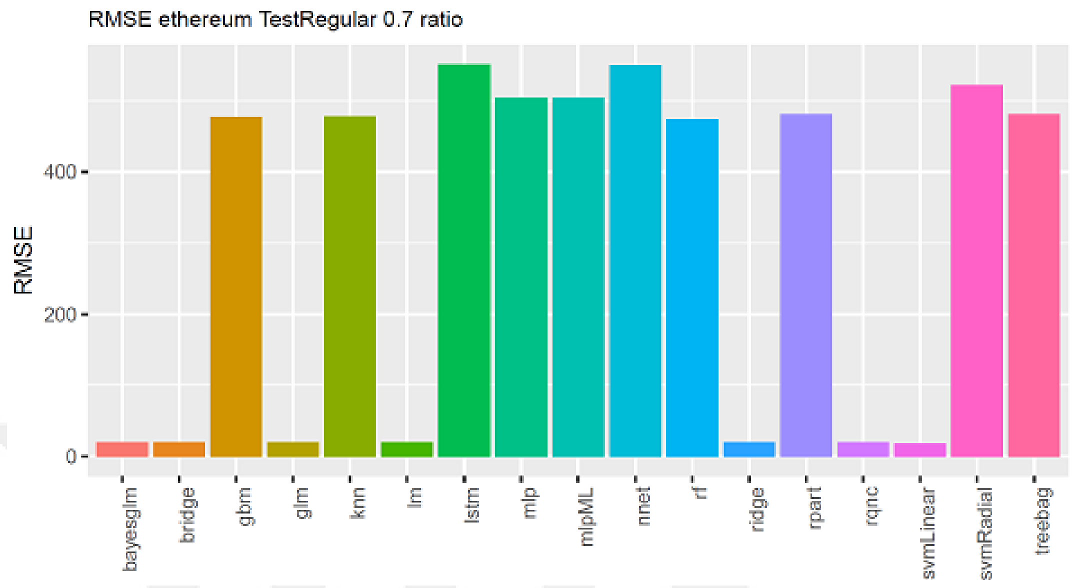

Figure 3.32: TestRegular on Ethereum Price Data

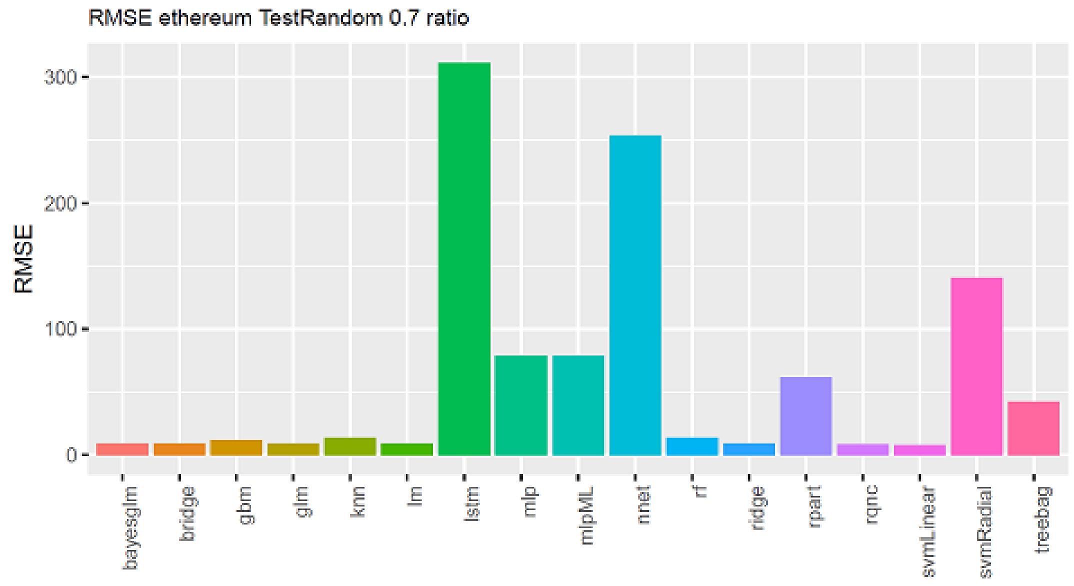

Figure 3.33: TestRandom on Ethereum Price Data

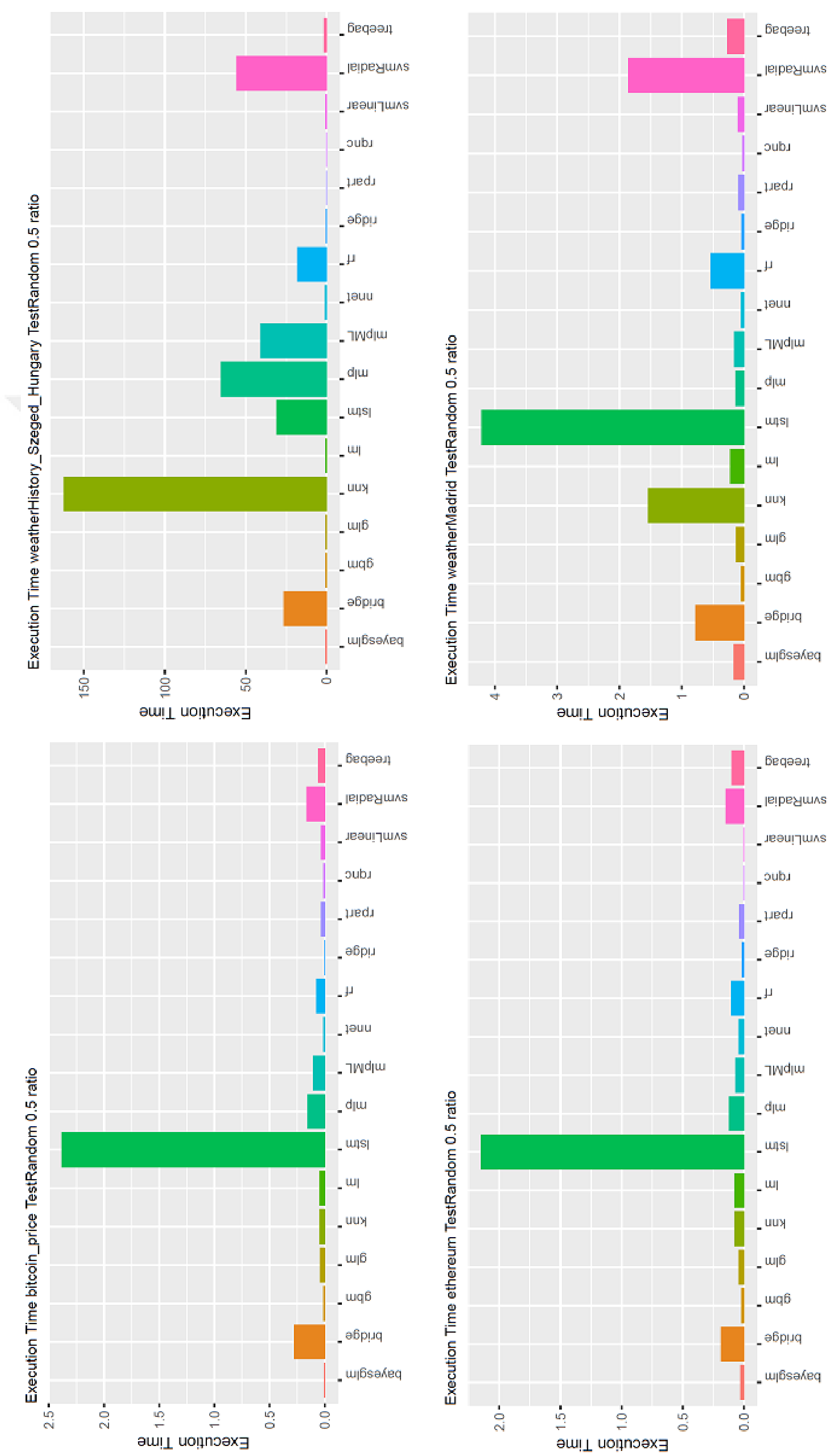

Figure 3.34: Execution Time of Models for Different Data

4 CONCLUSION

This paper aims to compare different forecasting algorithms on data from different domains. During these forecasting operations, we use certain parameters that change the training/testing data ratio, data selection and the prediction process type. These parameters include partitioning the data into training and testing sets according to different ratios at each time(data ratio), using different data selection such as creating the training set by picking the elements from the data in regular order or random order and testing the model performance on both training and testing data(prediction process type).

We performed statistical analysis with ANOVA test to determine the effect of using different parameters(data ratio, data selection, prediction process type) to the error rates. According to ANOVA test, data selection type(random/regular) and prediction process type(train/test) are infleuntial to the result. Using different data and different algorithms also change the result but training with different percentage of the data does not change the performance.

By observing the error rate graphs of the forecasting algorithms, we concluded that the data selection and prediction process type's effect depends on the characteristic of the data. For the cryptocurrency data, models perform with higher performance with random data selection while training because cryptocurrency data is more unstable. But for the weather data, picking the training set in regular order produces acceptable error rates.

We also observed that neural network based models such as LSTM did not perform accurately in our research. The reason for this may be the data size since the largest data we used in this research had 96453 records. The results may differ for larger data.

Also, a comparison between the execution times of the forecasting algorithms indicate that neural network based models take longer time to use. So they have relatively higher computational complexity than the other models.

After carefully considering the results, we concluded that linear models have high forecasting accuracy on our data and it is not necessary to increase the complexity and the execution time of the forecasting process by picking elaborate machine learning methods for the data used in this research. While the results may differ for different data and different tuning parameters for each algorithm, this research shows that it may well be possible to stick to legacy methods like linear regression and random forest while making forecasting.

This work can be extended in many different paths. For example, it is possible to increase the number of algorithms that were used to make forecasting. Making detailed parameter tuning for each algorithm may change the error rates for each algorithm. Since this paper only uses data from 3 different domains, it is well possible to increase the number of data by adding new data from different domains. This may also increase the validity of the research especially if the new data have substantially bigger sizes. Different test types may be used, also different error metrics that are used in multiple regression tasks can be adopted such as mean absolute percentage error. Another approach would be to make qualitative analysis to elaborate this research.For example, it is possible to use features that belong only to the weather domain to increase the regression performance. This may also help with outlier detection during forecasting.

Since this is a vast subject and there are many different researches that explain various algorithms, techniques and parameters, this research is open to progression.

REFERENCES

Adya, M., & Collopy, F. (1998). How effective are neural networks at forecasting and prediction? a review and evaluation. *Journal of Forecasting*, *17*(5-6), 481–495.

Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5-6), 594–621.

Alon, I., Qi, M., & Sadowski, R. J. (2001). Forecasting aggregate retail sales:. *Journal of Retailing and Consumer Services*, *8*(3), 147–156.

Atzori, M. (2015). Blockchain technology and decentralized governance: Is the state still necessary? *SSRN Electronic Journal*.

Basu, S., Das, N., Sarkar, R., Kundu, M., Nasipuri, M., & Basu, D. K. (2005). An mlp based approach for recognition of handwritten 'bangla' numerals. In *IICAI*.

Batista, G. E. A. P. A., & Silva, D. F. (2009). How k-nearest neighbor parameters affect its performance ?

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blanchet, P. (1990). Data compression using multilayer perceptrons. In *Neurocomputing*, (pp. 237–240). Springer Berlin Heidelberg.

Bourlard, H., & Morgan, N. (1990). A continuous speech recognition system embedding mlp into hmm. In D. S. Touretzky (Ed.) *Advances in Neural Information Processing Systems 2*, (pp. 186–193). Morgan-Kaufmann.
URL `http://papers.nips.cc/paper/222-a-continuous-speech-recognition-system-emb`
`pdf`

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). *Machine Learning*, *45*(1), 5–32.

Cade, B. S., & Noon, B. R. (2003). A gentle introduction to quantile regression for ecologists. *Frontiers in Ecology and the Environment*, *1*(8), 412–420.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.
URL `https://doi.org/10.1023/A:1022627411411`

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.

Crone, S. F., Hibon, M., & Nikolopoulos, K. (2011). Advances in forecasting with neural networks? empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, *27*(3), 635–660.

Douglas C. Montgomery, G. G. V., Elizabeth A. Peck (2012). *Introduction to Linear Regression Analysis (5th edition)*. Wiley.

Eide, E., & Showalter, M. H. (1998). The effect of school quality on student performance: A quantile regression approach. *Economics Letters*, *58*(3), 345–350.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, *96*(456), 1348–1360.

Faraway, J. J. (2006). *Extending the Linear Model with R*. Chapman & Hall.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Ghosh, & Reilly (1994). Credit card fraud detection with a neural-network. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences HICSS-94*. IEEE Comput. Soc. Press.

Github (2018). Experiment codes.
URL `https://github.com/msinanergen/regression_analysis`

Graves, A. (2013). Generating sequences with recurrent neural networks.

Graves, A., & Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.) *Advances in Neural Information Processing Systems 21*, (pp. 545–552). Curran Associates, Inc.
URL `http://papers.nips.cc/paper/3449-offline-handwriting-recognition-with-mult`
`pdf`

Gurney, K. (2004). *An Introduction to Neural Networks*. Taylor & Francis.

Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification and regression. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.) *Advances in Neural Information Processing Systems 8*, (pp. 409–415). MIT Press.
URL `http://papers.nips.cc/paper/1131-discriminant-adaptive-nearest-neighbor-cl`
`pdf`

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Isobe, T., Feigelson, E. D., Akritas, M. G., & Babu, G. J. (1990). Linear regression in astronomy. *The Astrophysical Journal*, *364*, 104.

Justesen, N., Bontrager, P., Togelius, J., & Risi, S. (2017). Deep learning for video game playing.

Kimoto, T., Asakawa, K., Yoda, M., & Takeoka, M. (1990). Stock market prediction system with modular neural networks. In *1990 IJCNN International Joint Conference on Neural Networks*. IEEE.

Koenker, R. (2005). *Quantile Regression*. Cambridge University Press.

Koenker, R., & Bassett, G. (1978). Regression quantiles. *Econometrica*, *46*(1), 33.

Kuhn, M. (2008). Building predictive models inRUsing thecaretPackage. *Journal of Statistical Software*, *28*(5).

Laurent, S., Rombouts, J. V. K., & Violante, F. (2011). On the forecasting accuracy of multivariate GARCH models. *Journal of Applied Econometrics*, *27*(6), 934–955.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 14–23.

Makridakis, S., & Hibon, M. (2000). The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, *16*(4), 451–476.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018a). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, *34*(4), 802–808.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018b). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, *13*(3), e0194889.

Melly, B. (2005). Public-private sector wage differentials in germany: Evidence from quantile regression. *Empirical Economics*, *30*(2), 505–520.
URL https://doi.org/10.1007/s00181-005-0251-y

P. McCullagh, J. N. (1991). *Generalized Linear Models*. Chapman & Hall.

Pilkington, M. (2016). Blockchain technology: Principles and applications. Post-print, HAL.
URL https://EconPapers.repec.org/RePEc:hal:journl:halshs-01231205

ROSENBLATT, F. (1957). *The Perceptron: A Perceiving and Recognizing Automaton (Project PARA). Report No. 85-460-1*. Cornell Aeronautical Laboratory.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Neurocomputing: Foundations of research. chap. Learning Representations by Back-propagating Errors, (pp. 696–699). Cambridge, MA, USA: MIT Press.
URL http://dl.acm.org/citation.cfm?id=65669.104451

Šajn, L., & Kukar, M. (2011). Image processing and machine learning for fully automated probabilistic evaluation of medical images. *Computer Methods and Programs in Biomedicine*, *104*(3), e75–e86.

Soon, W. M., Ng, H. T., & Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, *27*(4), 521–544.

Stucky, B., & van de Geer, S. (2015). Sharp oracle inequalities for square root regularization.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, *58*, 267–288.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). Show and tell: A neural image caption generator.

Wang, D., & Nyberg, E. (2015). A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D. thesis, Harvard University.

BIOGRAPHICAL SKETCH

First Name: Mithat Sinan

Surname: Ergen

Place of Birth and Date of Birth: Şişli/İSTANBUL 1991

Bachelor Degree: Kadir Has University, Information Technology (2014)

Double Major Kadir Has University, Computer Engineering (2014)

PUBLICATIONS

DTSS 2018 Conference - A Comparative Study for Accurate Forecasting