THE REPUBLIC OF TURKEY
BAHÇEŞEHIR UNIVERSITY

FORECASTING ELECTRICITY CONSUMPTION USING DEEP LEARNING METHODS WITH HYPERPARAMETER TUNING

Master's Thesis

ONUR ARSLAN

İSTANBUL, 2020

THE REPUBLIC OF TURKEY
BAHÇEŞEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE
BIG DATA ANALYTICS AND MANAGEMENT

FORECASTING ELECTRICITY CONSUMPTION USING DEEP LEARNING METHODS WITH HYPERPARAMETER TUNING

Master's Thesis

ONUR ARSLAN

Thesis Supervisor: ASSIST. PROF. DR. SERKAN AYVAZ

İSTANBUL, 2020

THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
BIG DATA ANALYTICS AND MANAGEMENT

Name of the thesis: Forecasting Electricity Consumption Using Deep Learning Methods With Hyperparameter Tuning
Name/Last Name of the Student: Onur ARSLAN
Date of the Defense of Thesis: 08/01/2020

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Assist. Prof. Dr. Yücel Batu SALMAN
Graduate School Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Serkan AYVAZ
Program Coordinator

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Comittee Members	Signature
Thesis Supervisor Assist. Prof. Dr. Serkan AYVAZ	----------
Member Assist. Prof. Dr. Atınç YILMAZ	----------
Member Assist. Prof. Dr. Tarkan AYDIN	----------

ACKNOWLEDGMENT

I would like to thank my advisor Assist. Prof. Serkan AYVAZ who supported me during my master's program and the thesis.

And thanks to Mr. Mehmet Ali VAROL to all of his support.

Finally, I would like to thank my family and friends who have always been with me throughout my life.

İstanbul, 2020 Onur ARSLAN

ABSTRACT

FORECASTING ELECTRICITY CONSUMPTION USING DEEP LEARNING METHODS WITH HYPERPARAMETER TUNING

Onur ARSLAN

Big Data Analytics and Management

Thesis Supervisor: Assist. Prof. Dr. Serkan AYVAZ

January 2020, 50 pages

In this study, it is tried to estimate one-day electricity consumption by using deep learning methods with a dataset that includes the change in time-dependent electricity consumption. After explaining the time series components and machine learning concepts, general information about previous studies on electricity consumption estimation is given. Since the dataset used is a time series, all the features are emphasized in detail and necessary operations like resample and reshape are performed before proceeding to the modeling. Tuning was applied to hyperparameters which significantly affect the performance of the algorithms used in the modeling stage and the most suitable parameters were searched for each method. Then the best results were compared with each other and the method with the lowest error rate was determined.

Keywords: Energy Consumption, Time Series, Deep Learning, Hyperparameter Tuning

ÖZET

HİPERPARAMETRE AYARLI DERİN ÖĞRENME YÖNTEMLERİ İLE ELEKTRİK TÜKETİMİNİN TAHMİNİ

Onur ARSLAN

Büyük Veri Analitiği ve Yönetimi

Tez Danışmanı: Dr. Öğr. Üyesi Serkan AYVAZ

Ocak 2020, 50 sayfa

Bu çalışmada, zamana bağlı elektrik tüketimindeki değişimi içeren bir veri seti ile derin öğrenme yöntemleri kullanılarak bir günlük elektrik tüketimi tahmin edilmeye çalışılmıştır. Zaman serisi bileşenleri ve makine öğrenimi kavramları açıklandıktan sonra, elektrik tüketimi tahmini ile ilgili daha önceki çalışmalar hakkında genel bilgiler verilmiştir. Kullanılan veri kümesi bir zaman serisi olduğundan, zaman serisi özellikleri ayrıntılı olarak vurgulanmış ve modellemeye geçmeden önce yeniden örnekleme ve yeniden şekillendirme gibi gerekli işlemler gerçekleştirilmiştir. Modelleme aşamasında kullanılan algoritmaların performansını önemli ölçüde etkileyen hiperparametreler üzerinde çeşitli ayarlamalar yapılarak her yöntem için en uygun parametreler araştırılmıştır. Daha sonra en iyi sonuçları veren modeller birbirleriyle karşılaştırılmış ve en düşük hata oranına sahip yöntem belirlenmiştir.

Anahtar Kelimeler: Enerji Tüketimi, Zaman Serileri, Derin Öğrenme, Hiperparametre Ayarı

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

ADAM	:	Adaptive Moment Estimation
ADF	:	Augmented Dickey-Fuller
AIC	:	Akaike Information Criteria
ANN	:	Artificial Neural Network
AR	:	Auto-Regressive
ARIMA	:	Autoregressive Integrated Moving Average
DNN	:	Deep Neural Network
GRU	:	Gated Recurrent Units
KPSS	:	Kwiatkowski–Phillips–Schmidt–Shin
LSTM	:	Long Short Term Memory
MA	:	Moving Average
MAE	:	Mean Absolute Error
RMSE	:	Root Mean Square Error
RMSPROP	:	Root Mean Square Propagation
RNN	:	Recurrent Neural Network
S2S	:	Sequence to Sequence
SGD	:	Stochastic Gradient Descent

SYMBOLS

Cell state of LSTM cell	:	c(t)
Forget gate vector of LSTM cell	:	f(t)
Input gate vector of LSTM cell	:	i(t)
Input modulation vector of LSTM cell	:	g(t)
Input vector of LSTM cell	:	x(t)
Number of autoregressive terms in ARIMA	:	p
Number of lagged forecast errors in ARIMA	:	q
Number of non-seasonal differences in ARIMA	:	d
Output vector of LSTM cell	:	y(t)
Output vector of LSTM cell	:	h(t)
Remainder component	:	R_t
Seasonal component	:	S_t
Time	:	t
Timeseries Data	:	y_t
Trend-Cycle component	:	T_t

1. INTRODUCTION

Electrical energy is a type of energy that needs to be transmitted rapidly, with high quality, efficient and rapid consumption throughout the world. With the increasing population, urbanization, industrialization and especially rapidly developing technology as an indispensable part of human life, the need for electrical energy is increasing day by day [1]. Energy production, distribution, and transmission facilities need to be planned for the future in order to meet the increasing energy demand. On the other hand, environmental impacts such as global warming and political problems such as energy dependency; it brings the necessity of saving in electricity consumption. The most important way is to achieve a balance between electricity production and consumption of electrical energy.

The continuous increase in the need for electrical energy, the limited resources and the inability to store electricity force the sector participants to make various plans. The necessity to consume electricity at the moment of production creates an obligation to maintain production/consumption balance. In recent years, this issue has been focused on since the inability to respond to the increasing consumption will cause great problems especially for countries in economic terms. On the other hand, environmental reasons such as global warming or national factors such as external dependence require saving in electrical energy. Under these conditions, regular plans are made in production, transmission and distribution systems in order to prevent bottlenecks in the electricity market; various consumption estimation methods are used to give predictable results [2]. The electricity demand estimates that are being made and to be made have an important place in the operation, control and planning of electric power systems, tariff arrangements and export areas for energy suppliers and consumers.

For Turkey, which is largely dependent on foreign countries for energy resources, it is quite important to predict electric power consumption accurately. Differences in production/consumption resulting from estimates that do not reflect the reality will have a negative impact on the national economy. Excess of production compared to consumption will cause waste of available energy. If consumption is higher than production, energy deficits will occur; interruptions, system bottlenecks, etc. problems.

Therefore, studies on energy demand, especially electricity demand forecasting, have gained importance in recent years [3].

The development of technology has enabled deeper learning algorithms to work faster and more efficiently. Thanks to the machines with high processing power, realistic estimation results have been obtained from large amounts of data sets. By taking advantage of these results, serious measures have been taken in many areas such as electricity consumption and significant financial gains have been achieved. In this way, it has become easier to develop and implement methods that have not been tried before in the machine environment, the detection and improvement processes of the errors have been accelerated and the door has been opened to the development of new methods depending on the data characteristics. In short, in the field of energy consumption, as in other areas, the development of technology is of paramount importance in the application of prediction algorithms, especially based on old data.

Even though the improvement in forecasting methods is increasing day by day, it is not possible to achieve realistic results in all areas. It is quite difficult to foresee the changes created by situations such as natural disasters or holidays in the estimation processes performed in areas affected by many such variables. After a natural disaster such as an earthquake in the area where consumption is estimated, long-term power outages will be experienced, which will result in a significant decrease in consumption compared to previous times. In addition, considering the renewal of residential areas and changing usage patterns and user profiles, it is difficult to make consumption estimates covering long periods and covering large areas.

The aim of this study is to make realistic estimates of electricity consumption by using deep learning methods by making use of innovations brought by technology. In order to obtain more accurate results in the estimations, adjustments will be made on each method according to the dataset used and the most suitable model will be searched to solve this problem.

The other parts of the study are listed as follows: in the second part, the estimation methods that are planned to be used for the study are explained and the previous energy consumption estimation studies are given. In the third chapter, the dataset used in the study is introduced and the time-series features of this dataset are discussed in detail. In the fourth chapter, the steps in the methods used are explained and in the fifth chapter, the findings obtained from these analyses are presented. In the sixth chapter, these findings were compared and the study was briefly summarized and finalized in the last chapter.

2. LITERATURE REVIEW

2.1. BACKGROUND

2.1.1. Time Series

Time series are special data models that are widely used in science, engineering, and business. Data sets associated with discrete-time values are generally defined as time series. The prediction of future data based on past observations is also called time series analysis. However, for a successful time series analysis, the components of the time series must be known. In this context, time series can have four important characteristics: seasonality, trend, cyclic and irregular variations.

2.1.1.1. Seasonal Variations

Repeated patterns over less than one year are called seasonal variations, e.g. increase in swimsuit sales in summer or a decrease in natural gas consumption in summer meanwhile increase in the winter season. Although the term seasonal effect first comes to mind within a year, it actually includes all periods limited to a certain period of time. Increasing or decreasing traffic density at certain times of the day, the number of hourly users of a website or a phone application during the day, the average number of customers per week in a store are other examples of seasonal effects. Climate, social traditions, religious holidays cause seasonal effects [4].

2.1.1.2. Trend

In time series the long-term tendency is expressed by term trend. The change in the averages of observations can be used as a practical test for the presence of the trend to us. To understand the term trend, it is necessary to examine the data as long as possible period of time. For example, the trend on infant mortality rates may not seem to change when looking at ten-year data, but the existence of the trend can be noticed when one hundred years of data is examined. Some examples of the trend are: increase in inflation, decrease in the number of living species, increase in the need of food, increase in education level, increased life expectancy with increased income [4].

2.1.1.3. Cyclic Variations

When the observations of a time series show fluctuations without a specific time, such fluctuations are defined as cyclic fluctuations or variations. Usually, the duration of these fluctuations is more than two years. Business-related cycles have an unpredictable duration of more than two years. Cyclic behavior, often confused with seasonal behavior, refers to fluctuations that do not depend on a fixed period of time, unlike seasonal behavior. In such patterns, the cyclic length is usually longer than seasonal cycle lengths. The population of some species may show cyclic fluctuations in decades or some economic indicators also have some patterns like this [5].

2.1.1.4. Irregular Variations

As the name suggests, it means unpredictable variations that occur in nature without being bound to a particular order. In a time series, these variations, which are excluded from trend, seasonal and cyclical components, are called residual variations. Irregular fluctuations are seen as a result of events that are not predictable, such as natural disasters, famine, wars, fires, etc [6].

Irregular fluctuations can be divided into two parts: episodic and residual fluctuations. Fluctuations that occur as a result of unpredictable but identified events such as strikes; fire and earthquakes are called episodic fluctuations. Residual variation is the name of random and unidentified variations that remain after the episodic variation is removed in a time series [6].

2.1.2. Time Series Decomposition

As we explained above, time series can come up with many different characteristics. And the task of separating a time series into these components makes it easier to predict. When dividing into the components of the time series, the trend and cycle components are often combined into one component under the name of trend. Thus, a time series can be expressed by trend, seasonal and residual components [7].

y_t data, S_t seasonal component, T_t trend-cycle, and R_t indicate that the remainder component and t is the time parameter thus the time series can be expressed by one of the additive (Equation 2.1) and multiplicative (Equation 2.2) decomposition methods.

$$y_t = S_t + T_t + R_t \quad (2.1)$$

$$y_t = S_t \times T_t \times R_t \quad (2.2)$$

2.1.3. ARIMA (Autoregressive Integrated Moving Average)

ARIMA is a time series model with the integration of AR (Autoregressive) and MA (Moving Average) models [8]. Such models are also referred to as Box-Jenkins models. When time-series data are relatively stationary, ARIMA models give high accuracy in forecasting. However, such models have a strong assumption that linear relationship between past and future data [8].

2.1.4. ANN (Artificial Neural Networks)

The term artificial neural network was first used in 1943 by Warren McCulloch and Walter Pitts in an article entitled "A Logical Calculus of Ideas in Nervous Activity" [9]. The ANNs continued their development until the 1960s. The period between 1960 and 1980 is called the dark period for ANNs. With the new network architectures discovered during the 1980s, interest in ANNs awoke. In the 1990s, Machine Learning algorithms for example Support Vector Machines had become preferable than ANNs. Since the early 2000s, several new major developments have led to a tremendous development in the area of ANNs [10].

Since ANNs are systems that require large amounts of data due to their structure, increasing data amount positively affected their development. However, the huge increase in computer power also allowed the training of ANN architectures. In addition, improvements in training algorithms have contributed positively to the development of ANNs. Finally, increasing popularity after each success has led to the transfer of larger funds to ANN development [10].

2.1.5. DNN (Deep Neural Networks)

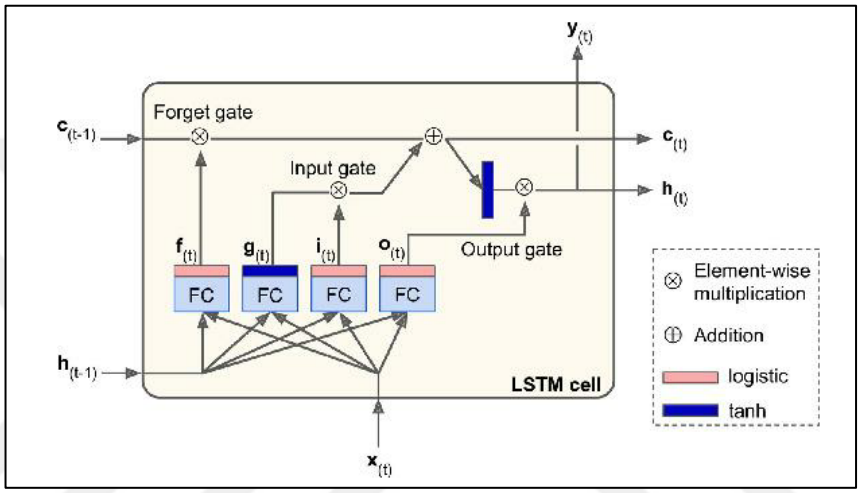
Artificial neural network structures with multiple layers established to extract applicable properties from raw data are called deep neural networks [11]. Deep neural networks are successful in areas such as image recognition, computer vision, natural language processing, voice recognition, speech recognition, etc [12]. The deeper you are in the

neural network, the more complex features you can recognize by your neurons, because they combine and reassemble the features from the previous layer [13].

2.1.6. LSTM (Long Short-Term Memory)

LSTM networks are kind of special RNN network architectures that can learn long-term dependencies [14]. They were introduced by Hochreiter & Schmidhuber in 1997 [15].

Figure 2.1: Long Short-Term Memory



Source: [10]

In the long term, the ability to learn what to store, what to discard and what to use from that stored information forms the character of LSTM architectures. When we examine the architecture, we can see from the upper left corner that the long-term c(t-1) has traveled through the network and forgotten some memories at the forget gate and obtained new information from the input gate. As you can see, the result c(t) is sent without any further conversion. It is seen that some memories were dropped and some memories were added. The short-term result h(t) is generated after the addition at the input gate by the hyperbolic tangent activation function and through the filter of the output gate [10].

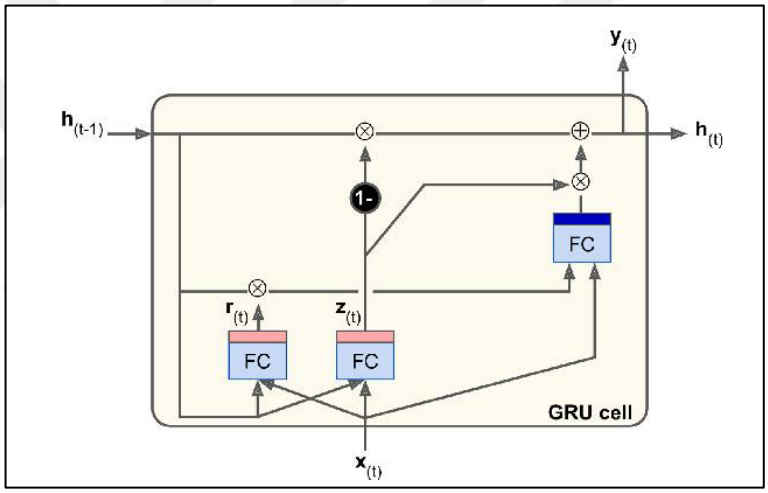
The four connected layers perform the task of creating new memories fed by the input vector x(t) and the previous state h(t). The main layer g(t) produces outputs. It produces direct outputs (y(t), h(t)) in a normal ANN neuron. However, it does not produce direct output in LSTM, instead of that is stored in a relatively long term [10].

The other three layers carry out the control of the gates. The logistic activation function is used in all three layers. For this reason, the outputs of these gates are between 0 and 1. The gates are opened or closed according to the value received. The gate f(t) determines which of the long-term stored information should be deleted. Which information created by g(t) is added in the long-term storage is decided by gate i(t). Finally, it is decided by the output gate that the generated information is read by the long-term situation and convert to output [10].

2.1.7. GRU (Gated Recurrent Unit)

GRU can be thought of as a simplified version of LSTM.

Figure 2.2: Gated Recurrent Unit



Source: [10]

Both state vectors in the LSTM are combined in the GRU. Unlike LSTM, the input and output gates are controlled by a single gate controller. If the input gate is open and the forget gate is closed, it is parameterized with a value of 1 in the gate controller. If the opposite occurs, the gate controller takes the value 0. Actually, this is common in LSTM. The GRU architecture does not have an output gate. Output generation is performed by the full state vector in any case [10].

2.2. RELATED WORKS

Time data has been tried in many applications in electricity consumption and price estimation models. In the study aimed to find the monthly energy need, the trend and fluctuation characteristics of the time series were separated and a single model with high estimation accuracy was produced by combining the results of two separate models [16]. In another application, time series were used to estimate the electricity price with error rates below 5% on a weekly basis by dynamic regression and transfer function methods [17].

The results of previous studies with time-series datasets were effective in the selection of the ARIMA method. The ARIMA algorithm has many applications in the fields of social sciences, engineering, and finance [8]. In a study used to estimate the price of electricity, it was observed that some countries produced results with an error rate of about 5% in the hourly price estimate for the electricity market [18]. In another electrical charge estimation study, ARIMA showed that it was more successful than the ANN model of 20 neurons. According to the same findings, it has been observed that as the prediction range increases, performance decreases even if it is ahead of ANN [19].

ANN, which is the traditional machine learning method, has been used in short and long term trials in energy consumption estimation [20] [21]. The ANN network, trained by the Levenberg-Marquardt algorithm, produced a model with an error rate of 5% -14% in short-term electricity price estimation. The ANN model, which was created using only three layers, as compared with ARIMA results and it was found that the ANN method was successful compared to ARIMA for each trial. According to ARIMA, the ability to calculate nonlinear functions, establish an input-output relationship and operate data-driven was effective in the emergence of this performance [22].

In order to solve the long term dependency problem, the traditional ANN methods were replaced by the LSTM method and used in many analyzes in various fields. There are studies that have achieved success between 60% - 65% by trying various optimization approaches on stock price [23]. In a study in which LSTM was used together with S2S (Sequence to Sequence) architecture, it was observed that more successful results were

obtained compared to the classical LSTM method [24]. In the LSTM S2S method, variable-length input series are converted by the encoder into a fixed-length vector and used as input for the vector decoder. The results generated by the decoder include estimation up to the desired time. In this way, arbitrary-length inputs are allowed, and the random measurement values in the previous time steps become available as input for estimation in a random time step. As will be emphasized in this study, the success of the results obtained by optimizing the LSTM method varies considerably. There are applications that are more successful than other optimization methods such as SGD (Stochastic Gradient Descent) and RMSProp (Root Mean Square Propagation) in the estimation of electricity price made by ADAM (Adaptive Moment Estimation) optimization method [25].

Despite the significant differences in the architecture of the gate, such as the spread of memory information to all units without checking, these two methods have been used in comparison with each other in many studies. Given its structural similarity and performance in previous studies, it is difficult to say which method is better [26]. The GRU model, which has been applied by optimization in some of the analyses, where close results are generally obtained, has outperformed other machine learning algorithms [27]. When the Sequence to Sequence (S2S) technique is used together, it appears to be quite successful compared to the more comprehensive models such as RNN S2S and LSTM S2S as well as traditional RNN (Recurrent Neural Networks) and LSTM [28].

The fact that the dataset used in the study has the time series characteristics enabled us to work with supervised learning techniques in the machine learning model. Based on a real dataset, 1-day energy consumption is estimated with LSTM and GRU modeling. Firstly, detailed tests and visual evidence were obtained to eliminate the effects of time series attributes such as stationarity, and then the dataset was made ready for analysis. According to the aforementioned studies, the contribution of hyperparameter tuning to LSTM and GRU methods has been examined in detail and the best results have been created according to the data obtained with 1-year energy consumption. During the tuning process, the focus was on insufficient data. In addition to RNN methods, tuning was

performed on the ARIMA method which is traditionally tried in time data analysis and all results were compared with each other.

3. DATA EXPLORATION AND PREPARATION

In this section, the data set used in the study is introduced. In order to better understand the data, various visual outputs were used before proceeding to the analysis stage, and detailed information was given on the preliminary preparations for the analyses to be performed.

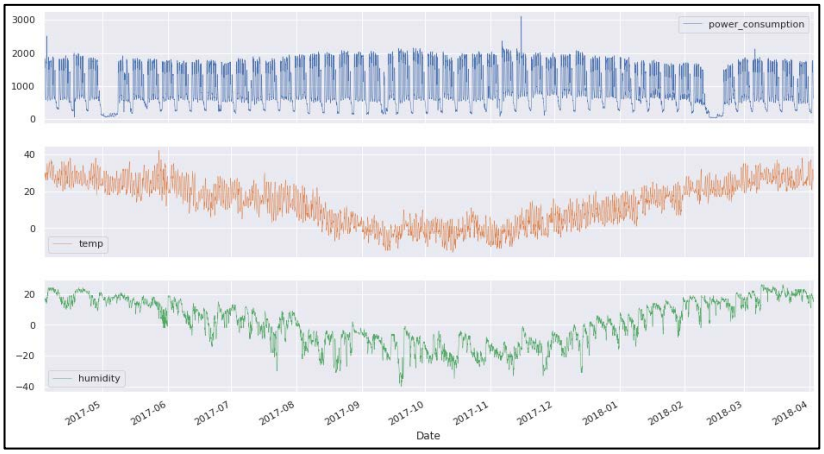
3.1. GENERAL INFORMATION OF DATA

The dataset was collected from an energy distribution company in Turkey and it indicates the total hourly electricity consumption between 2017-04-03 and 2018-04-02. It was obtained directly from the organization and reflects the actual values. Since the value, which is kept as an index, describes a time, it is necessary to define this data as a time series and the investigations have been made in this direction.

The data used consists of a total of 8760 rows and 4 columns. The dataset consists of the features listed below.

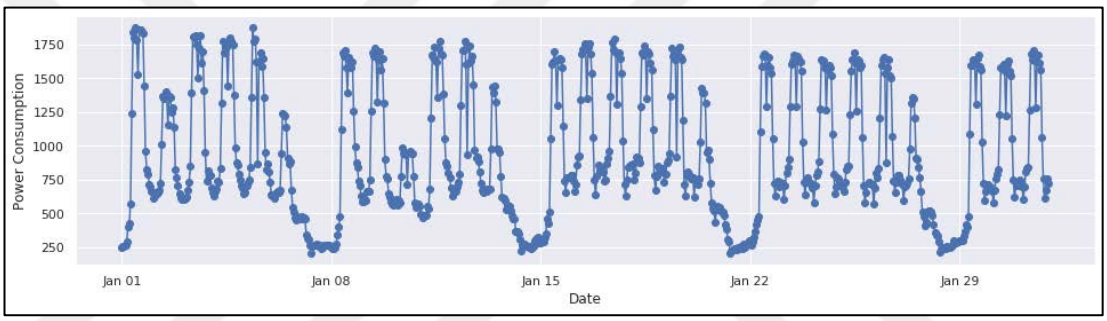
i. **date:** hourly distributed time information. The index of the dataset.
ii. **power_consumption:** energy consumption in an hour time period
iii. **temp:** average temperature in that hour interval
iv. **humidity:** average humidity in the same time interval.

Figure 3.1: Dataset distribution



When the Figure 3.1 is examined, it is seen that the energy consumption, temperature, and humidity values change over time. According to these graphs drawn from hourly data, energy consumption shows almost no trend characteristics but includes a certain seasonality feature on a weekly basis. In addition, it is seen that these values are quite low at times and very high in rare cases. These disorders will be examined more clearly in the future. For the temperature and humidity values, it can be predicted that there is an annual seasonality, but these columns also do not contain a trend feature either. The values of these two columns appear to be more like fluctuation or appear to be very much affected by noise.

Figure 3.2: Power consumption distribution in January 2018



When the monthly power consumption change is examined data taken from the real dataset for January 2018 in Figure 3.2, the weekly seasonality can be seen clearly. Another point that needs to be emphasized here is that consumption is quite low compared to weekdays at weekends. Considering the intensity of working weekdays, it is quite predictable that the consumption is more active than the weekend. The above graph also proves this situation.

3.2. RESAMPLING

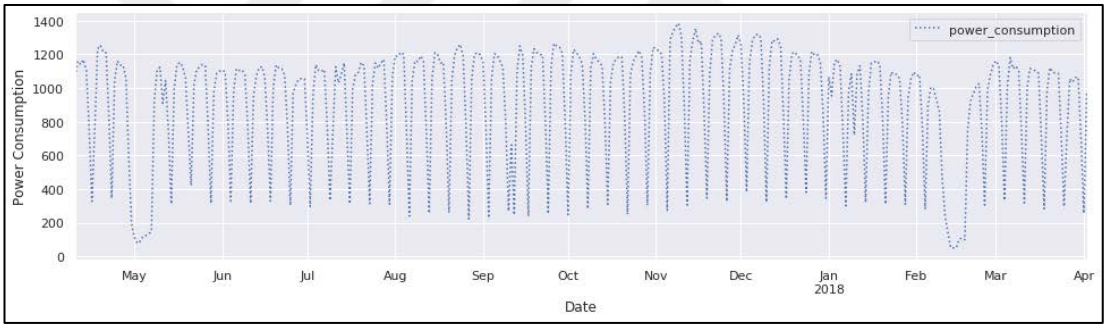
Resampling is often used in time series datasets to better understand the data and achieve more accurate results during the analysis phase. This method can be defined as reconstructing the dataset at a lower or higher frequency. Downsampling is a process of aggregating data using operations such as sampling to a lower frequency, usually averaging or summing. For example, as we will do here, it is a downsampling process

that averages every 24-hour data group to create a daily dataset. The opposite is the process of creating the hourly dataset from a daily dataset using specific methods.

The result required to be obtained is daily based and the completion of the studies with hourly data in a much longer time than expected shows that the resampling in the dataset should be performed. Therefore, the first preprocessing step with the dataset is to resample the data to an appropriate value.

The power consumption distribution for the dataset that is regenerated based on the daily average is as follows in Figure 3.5. Looking at this graph, there was no change in trend, but it was observed that the weekly seasonality could be easily monitored.

Figure 3.3: Daily resampled power consumption distribution



The following graphs Figure 3.6 and Figure 3.7 illustrate other down-sampling techniques, weekly and monthly average respectively. By looking at these two graphs, it can be clearly seen that it would be of no use to treat the data in this way. It is not possible to come across trend and seasonality components that define time series data in both types of resampling with this data set. In addition, 4-hour and quarter (3-month) resampling was also tried, but the same results were obtained.

Figure 3.4: Weekly resampled power consumption distribution

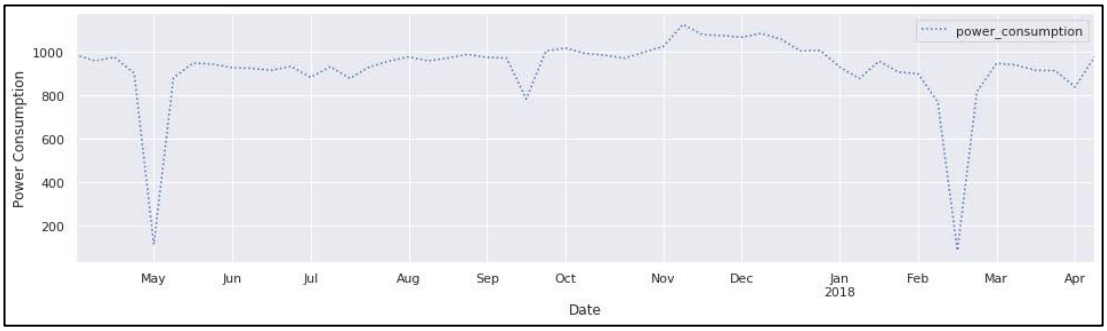
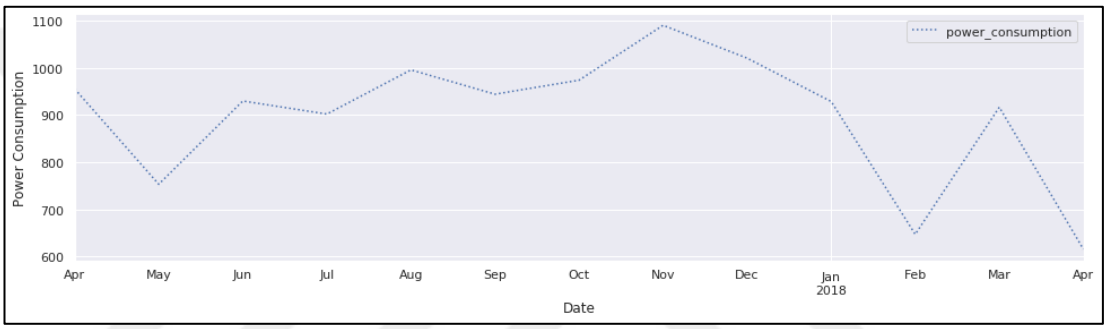


Figure 3.5: Monthly resampled power consumption distribution



3.3. FLUCTUATIONS

Figure 3.3 and Figure 3.4 show hourly power consumption between 2017.04-2017.05 and 2018.02-2018.03 respectively. The two graphs draw attention to the fact that there is hardly any energy consumption for a week. The fact that the data at these points contains a very small value reveals the possibility that this may be caused by any consumption problems. Therefore, in order to use this situation in the analyses, the values in the specified ranges are not changed by automatic filling operations, a new feature (isActive) is added to the dataset that describes these irregularities. This new column is filled in as 0 for days where energy consumption is less than 200 and as 1 for days when it is high.

Figure 3.6: Hourly power consumption between 2017.04-2017.05

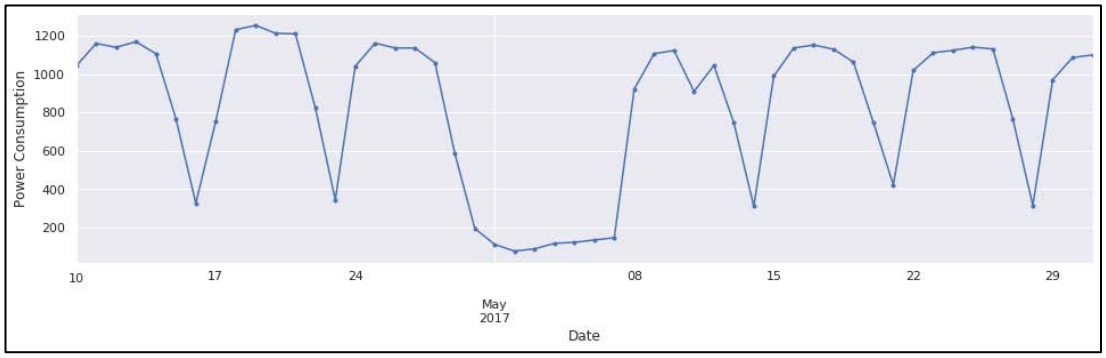
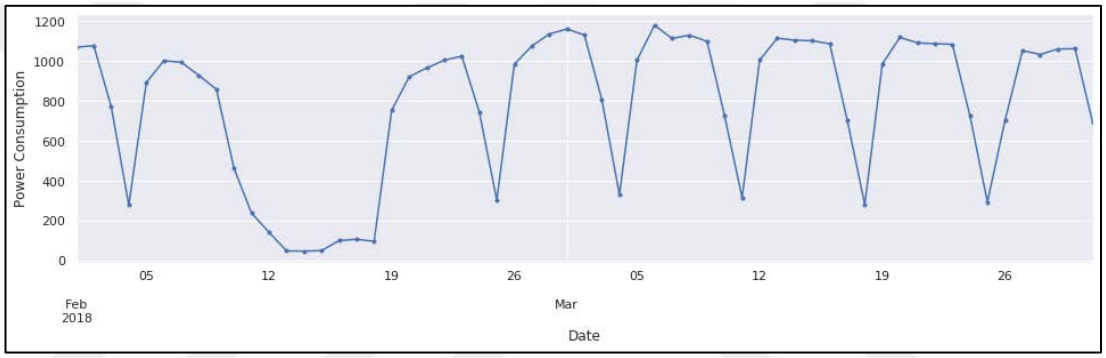


Figure 3.7: Hourly power consumption between 2018.02-2018.03



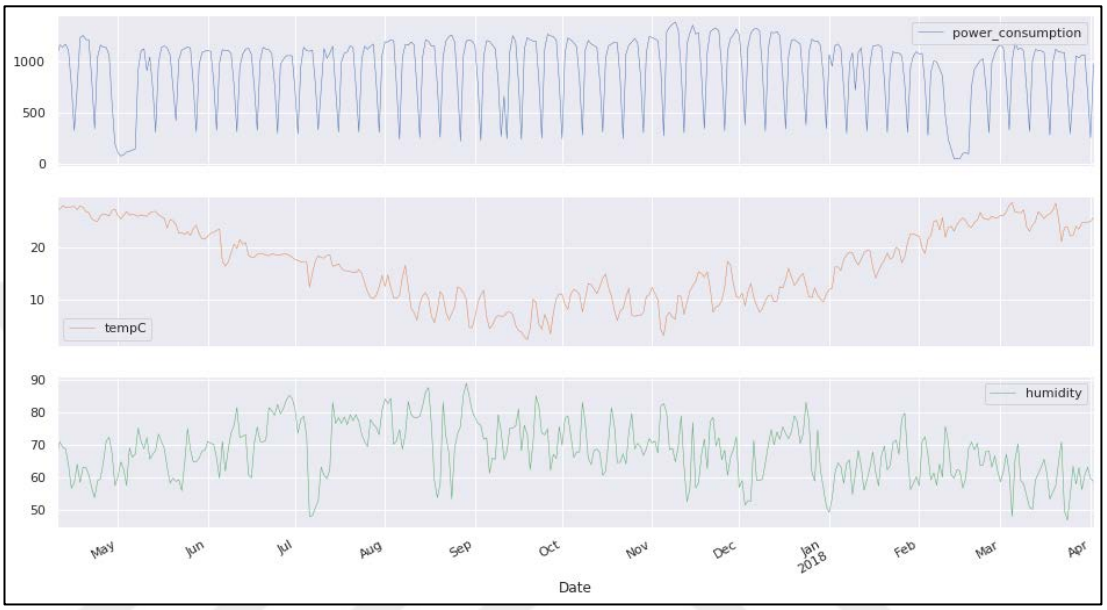
3.4. DATA CLEANING

Since no column contains the missing value, no automatic filling technique was required. The fact that all columns except the time information are numeric type provides great convenience in the analysis methods to be used. Therefore, preprocessing operations such as One Hot Encoding to use categorical columns for analysis such as deep learning methods or ARIMA are not needed.

Irregular progress was observed in the temperature and humidity features to be used for the analysis. For both of these, the probability that these qualifications are determined randomly creates a trust problem. Therefore, an external source [29] was used for weather information related to the days in the data set. Since the current dataset is distributed over the daily average, the added weather information was collected in the same format and the old temperature and humidity values were removed during the addition process. The distributions of the resulting dataset are as in Figure 3.8. Looking at these graphs, sudden

humidity drops can be seen in some months. The temperature data exhibits an annual seasonality as expected and exhibits a highly variable distribution during the winter months.

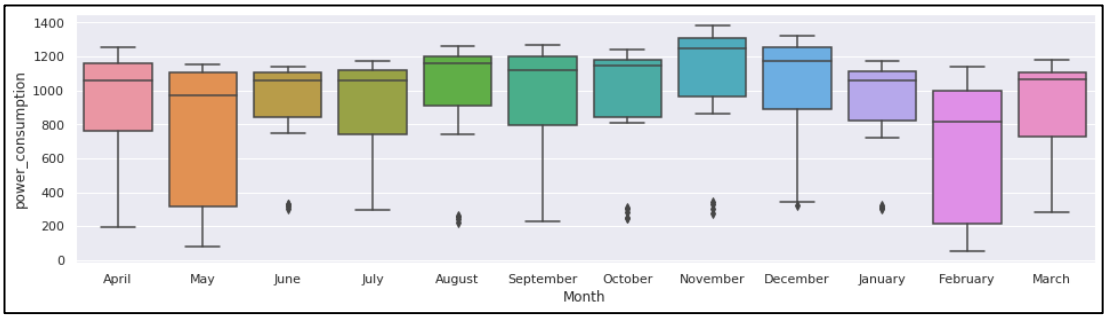
Figure 3.8: Daily power consumption after difference process



3.5. SEASONALITY & TREND

Seasonal component, one of the most important elements explaining the temporal series, was re-examined by means of various boxplots in addition to the previous items. Firstly, the dataset containing the daily averages created by resampling was copied and two new columns, month and week names, were added to the copied new dataset. Below it is possible to see the distribution of power consumption according to these two new columns on the boxplot monthly (Figure 3.9) and days of the week (Figure 3.10). By looking at these graphs, it is re-proved that it is impossible to determine a monthly seasonality. Apart from the increase in March, the change in power consumption has continued on a fairly stable level. The low two quartiles in May 2017 and February 2018 are due to the phases in which the aforementioned energy consumption is almost nonexistent. While power consumption continues normally in both months, a one-week interruption or consumption issue has caused data corruption.

Figure 3.9: Monthly power consumption boxplot



In spite of all these, the weekly seasonality was confirmed when the data was examined weekly in Figure 3.10. Just as envisaged, the energy consumed at weekends is at very low levels almost every week. Outliers on weekdays are thought to be due to holidays. During the national days or public holidays on weekdays, energy consumption was observed just like the weekends and therefore they could not enter the 95 percent range in normal distributions. In summary, the use of weekly periods in seasonal studies will yield the most accurate results.

Figure 3.10: Days of week power consumption boxplot

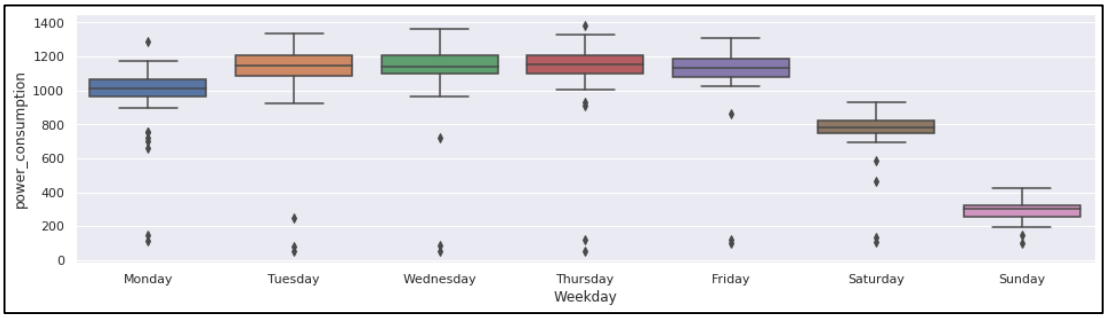
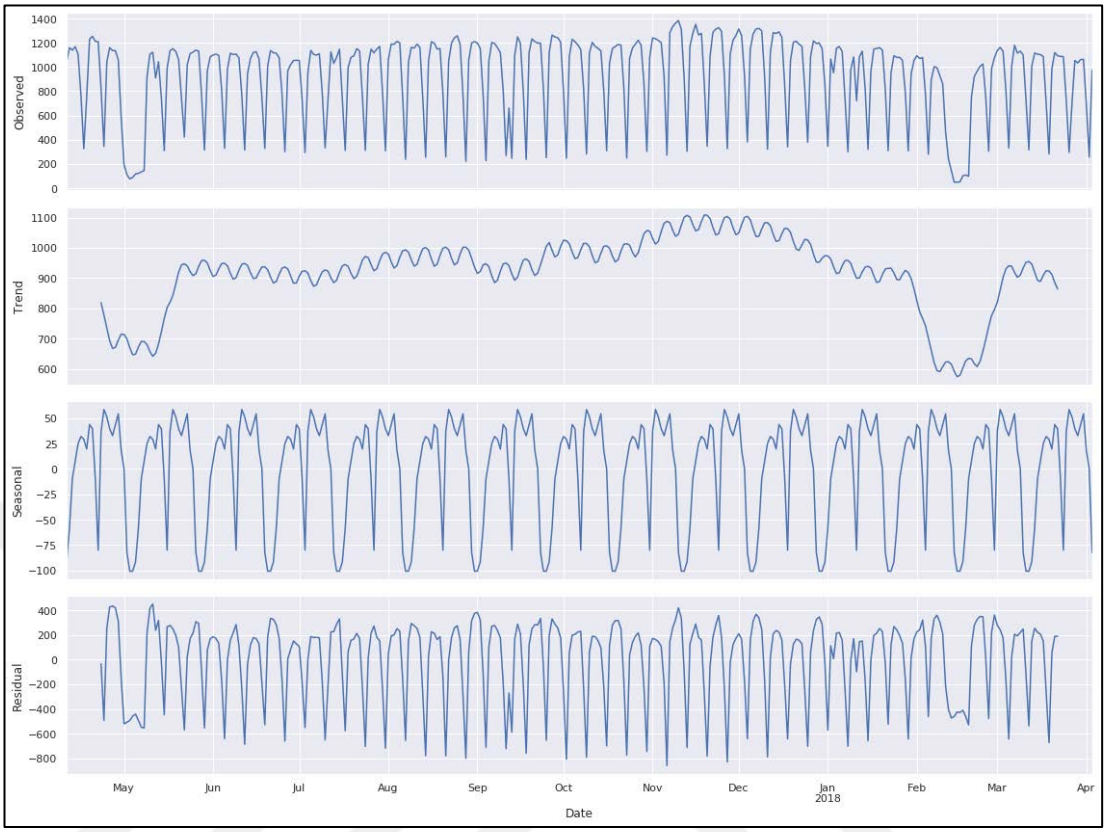


Figure 3.11 shows the decomposition of seasonal, trend and residual values for the power consumption attribute. Since the seasonal variation is generally constant at previous findings, the decomposition process is set to additive. By looking at the obtained graph, it has been proved that our data does not really contain a trend. It is not possible to determine the seasonality with the help of these results. The absence of a certain trend, or even its irregularity, has made this decomposition very uncertain and prevented the determination of the seasonality.

Figure 3.11: Decomposition of power consumption



In light of all these findings, it was decided to continue on the basis of daily resampling of the dataset, and there was no clear trend for this new dataset but there was a weekly seasonality. Further studies were carried out on this daily dataset.

3.6. STATIONARITY

For a series, stationarity means that statistical properties such as mean, variance, autocorrelation are constant over time. If one of these statistic values is not stable in time, the series does not show stationarity. Tests such as ADF (Augmented Dickey-Fuller) or KPSS (Kwiatkowski–Phillips–Schmidt–Shin) are used in addition to using visual tables to determine whether a series is stationary. These tests are based on the rejection or validity of the null hypothesis proposed. Table 3.1 shows the ADF and KPSS test results related to the power consumption in the daily data set.

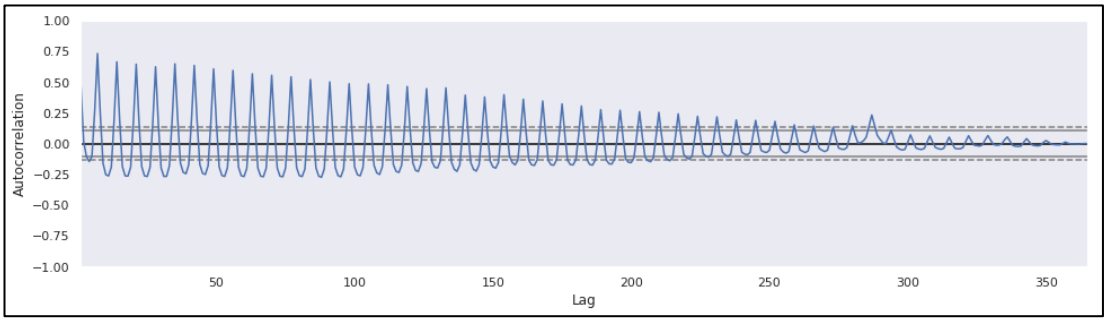
Table 3.1: ADF and KPSS test results

	ADF	KPSS
Test Statistic	-4.567	0.19
p-value	0.000148	0.10
Critical Value (1%)	-3.449	0.739
Critical Value (5%)	-2.870	0.463
Critical Value (10%)	-2.571	0.347

In order to interpret these results, it is necessary to mention in which situations the series can be specified as stationary. For ADF, if the test statistic is lower than critical values, the null hypothesis is rejected and in this case, the series is considered stationary. p-value of less than 0.05 is also one of the conditions for rejecting this hypothesis. For KPSS, the null hypothesis is rejected if the critical values are smaller than the test statistic, but this means that the series is not stationary. In the light of this information, when the results in Table 3.2 are considered, it is appropriate to say that the data set is stationary with both tests.

Whether a series is stationary can be interpreted by looking at the autocorrelation graph in addition to the tests mentioned above. Figure 3.12 shows the autocorrelation graph for the daily power consumption changes. Autocorrelation is an important way to measure and explain the intrinsic relationship between observations within the time series. The strength of an internal correlation within a given time period can be checked by this method. The values shown on the graph are +1 if the correlation is very strong and positive, -1 if the correlation is very strong and negative, and 0 if no correlation exists. The area formed by the dashed lines indicates the confidence interval, and in general, it can be said that dataset has a certain correlation. If looking carefully, a high and positive correlation can be seen in lag values that are multiples of 7.

Figure 3.12: Autocorrelation plot of power consumption



Unlike the results of ADF and KPSS tests, the autocorrelation graph clearly shows that this series is not stationary. Therefore, in order to stationaries the series, the difference taking method was used. This method can be summarized as subtracting the values in the series from their values after shifted. As in the datasets used in this study, it is necessary to shift the series by the number of days creating seasonality. The new series (Figure 3.13) obtained after the difference process shows the difference between the value of each day and their value in the next period (7 days). When the graph is examined visually, it can be clearly seen that there is no seasonality as before.

Figure 3.13: Daily power consumption after difference process

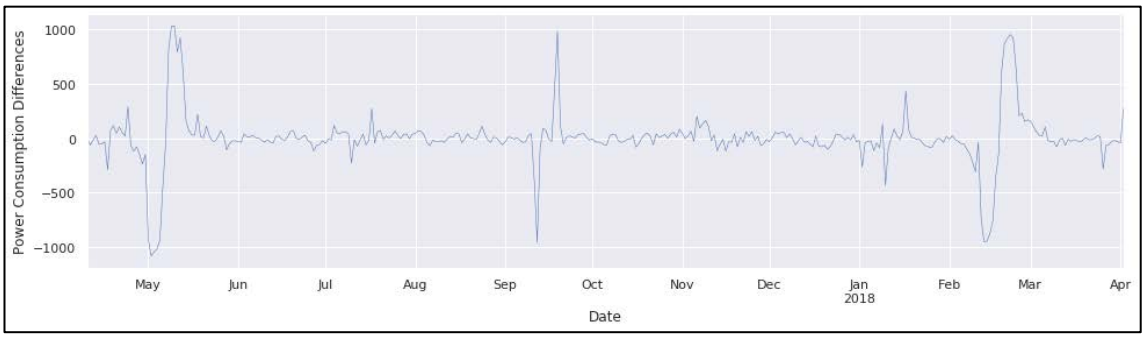
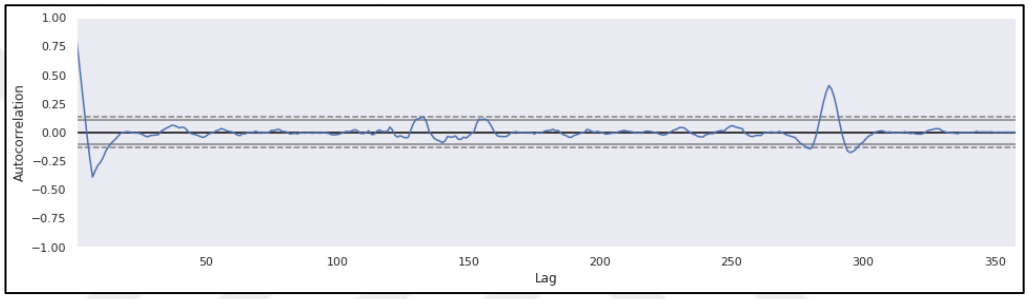


Table 3.2 shows the results of the ADF and KPSS tests with the new series and Figure 3.14 shows the autocorrelation graph. Both outputs characterize that the new series is stationary in line with the aforementioned explanations. As a result, the stationarity in the series has been removed and subsequent studies have been continued on this new series.

Table 3.2: ADF and KPSS test results after difference process

	ADF	KPSS
Test Statistic	-5.412	0.028
p-value	0.000003	0.10
Critical Value (1%)	-3.449	0.739
Critical Value (5%)	-2.870	0.463
Critical Value (10%)	-2.571	0.347

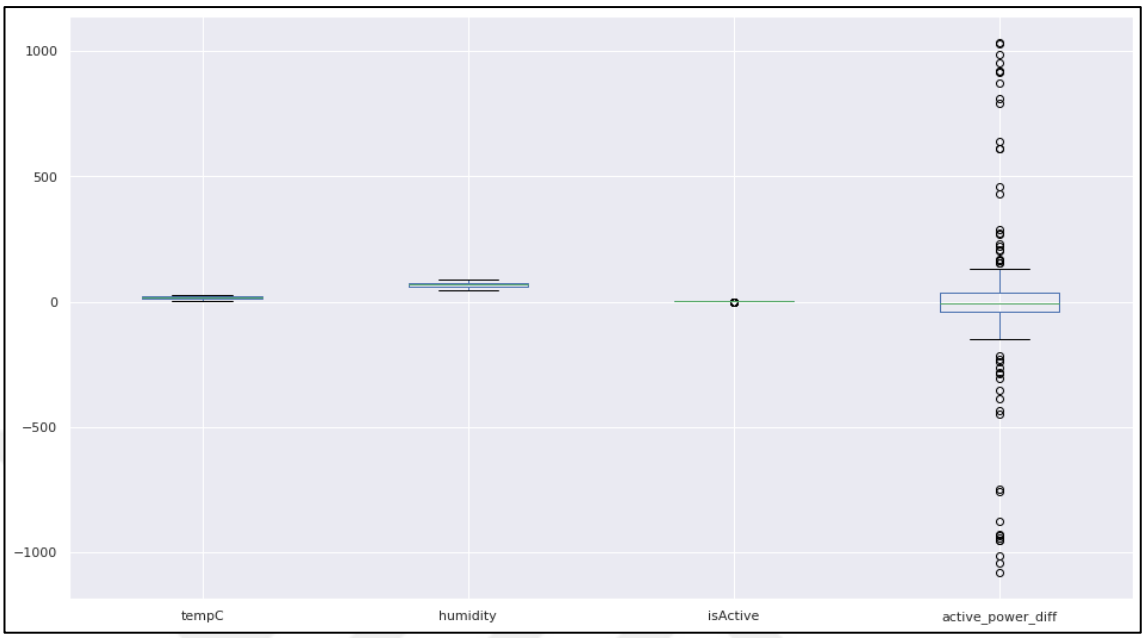
Figure 3.14: Daily power consumption after difference process



3.7. NORMALIZATION

Figure 3.15 shows the boxplot for each attribute in the data set to be used for analysis. In general, even if there is a propagation around 0, there are values between -1000 and 1000. The normalization of such series that are not of the same scale plays an important role in eliminating optimization problems during modeling. The normalization process makes learning less sensitive to the scale of the features and improves the quality of the analysis. For this reason, power consumption normalization was performed and the values in the whole series were changed with their equivalents between 0 and 1. In addition, unlike power consumption, temperature, humidity, and isActive qualities are not required a normalization process. As seen in Figure 3.15, all three attributes have the same scales and values close to each other.

Figure 3.15: Autocorrelation plot of power consumption



3.8. SPLIT TRAIN & TEST SET

Before proceeding to the modeling stage, one of the most important procedures to prevent problems such as overfitting and underfitting is to divide the dataset into training and test set. After the modeling process is done with the training set, it is observed by using the test set how accurate a result is found. This prevents data from being memorized and achieves a comparable model success.

The resample dataset, which indicates 365 days, was 358 rows as a result of the difference. 70 percent of these 358 samples are split for use as a training set and the rest as a test set. Thus, the training set consists of 250 samples and the test set consists of 108 samples.

3.9. RESHAPE

According to the modeling methods to be used when analyzing time series, various transformations should be made on the data. Since the data set used is a time data, the problem is handled as supervised learning (using the observation in the previous time steps as an input to predict the observation in the current time step). This can be

considered as trying to predict the next day with 7-day data. In this way, each element in the training set that will form the model will have a small dataset containing 7 days of data. Univariate or multivariate time series are reshaped by this method and made ready for use in various RNN models.

In special RNN methods such as LSTM and GRU, the dataset given to the input layer must have a 3-dimensional shape. These dimensions are given below with their description.

i. **Samples:** The number of rows of the dataset given to the input layer, or the size of the data to be used for training in the dataset.
ii. **Time steps:** Number of points containing backward information in each instance of the training data. For the example given above, this value is 7.
iii. **Features:** Number of features per observation. For univariate datasets, this value is 1, whereas for multivariate datasets it is the number of features.

For special RNN methods such as LSTM and GRU, it is very important to determine how many days of data will be used in reshape, to determine the number of features or the number of outputs. Since the expected time specified in the problem definition is 1 day, each sample in the output (Y) series to be generated must contain a single value. During the creation of the input (X) series, 3-day backward information was used. As a result of the reshaping process, the inputs and outputs to be used in the neural network models are divided into training and test sets as shown in Table 3.3.

Table 3.3: Input and output sets after reshape

X_train	Y_train	X_test	Y_test
(247, 3, 4)	(247, 1)	(105, 3, 4)	(105,1)

This reshaping process is not required in the ARIMA method to be applied. Also, in this method, the model will only be fed with the difference of power consumption attribute, so a univariate analysis will be performed. Therefore, the sizes of the training and test datasets to be used for ARIMA are (250, 1) and (108, 1) respectively.

4. METHODS & ANALYSIS

4.1. LSTM

The first method used for analysis is the LSTM, a deep learning method, unlike ordinary machine learning approaches. In the training of this model, the aforementioned 247 sample training set was used. Each sample in this training set contains 4 numerical attributes, including 3-day temperature, humidity, isActive and power_consumption_diff_scaled.

LSTM application is made in 3-dimensional tensor using Python language Tensorflow background in Keras deep learning library. 2 layers were used in each experiment and Dropout layers with a threshold value of 0.2 were added after these layers. On the output, there is a 1-unit Dense layer that indicates the power consumption value of a single day. In the optimization methods, learning rate and momentum values are left by default.

The structuring phase in deep learning methods such as LSTM is a very difficult task. In order to fully understand a particular predictive modeling problem, different configurations need to be explored from both a dynamic and objective perspective. These configurations are called hyperparameters in machine learning. There are many hyperparameters that affect the accuracy of the model for the LSTM method. Although this analysis method is very popular for time series, it differs in each dataset for hyperparameter selection, and the use of correct parameters yields very high successes. Therefore, the choice of hyperparameters was emphasized in the analysis performed with LSTM and the success of this study was tried to be increased.

In order to summarize the success of the model, RMSE (Root Mean Square Error) was used as the error score scale between the test set and predictions, and MAE (Mean Absolute Error) was used during the fit procedure. The RMSE scale is very successful in punishing big errors and also produces a score in the same units as the predicted data. The numerical results shown in all studies will be evaluated on this scale.

10 experiments were performed with the same hyperparameters in order to avoid the randomness of the model performance. This is because initial weights within the LSTM network can produce very different results in each trial. These weights can also be considered as a hyperparameter for LSTM and initial values are of great importance in that the analysis results are not subject to the problem of vanishing/exploding gradients. To address these problems, the LSTM architecture includes the activation functions mentioned earlier. In the study, such as changing the initial values or differentiation of activation functions were not performed. Randomly selected initial values and tanh/sigmoid activation functions from the LSTM architecture were used.

In order to monitor the model performance more controlled, loss graph of both training and test data was plotted for epoch steps during each fit process, and error (RMSE) values were calculated after the fit and test dataset. In the graphs drawn, the training set was tried to be separated as blue and the test set as orange. The following are the hyperparameters that are evaluated respectively.

4.1.1. Epochs

In the LSTM modeling process, all of the training data may not participate in the training at the same time in the backward parameter update process. They can take part in training in a number of parts. The first part is trained, the performance of the model is tested, and the weights are updated according to the success with backpropagation. Then the model is re-trained with the new training set and the weights are updated again. This process is repeated in each training step and the most suitable weight values are calculated for the model. Each of these training steps is called epoch. The first LSTM parameter used for tuning is the number of training periods.

Since the most appropriate weight values are calculated step by step to solve the problem in deep learning, the performance in the first epochs will be low and the performance will increase as the number of epochs increases. However, after a certain step, the learning momentum of the model will decrease considerably. The size of the epoch number also varies according to the type of problem. Epoch number should be kept larger than other models in methods such as pattern learned LSTM. As the number of epochs increases, the performance of the model increases significantly. But, since the performance will

increase in very small units after a certain epoch, training can be terminated at these points.

Other fixed hyperparameters determined for the epoch comparison are as in Table 4.1. Using these fixed parameters, model training was repeated 10 times in 50, 100, 200, 500 and 1000 epochs.

Table 4.1: Fixed hyperparameters on LSTM epoch tuning

Batch Size	Neuron Sizes	Optimizer
8	50x75	ADAM

4.1.2. Batch Size

In deep learning applications, learning by processing all the data in the data set at the same time is a costly task in terms of time and memory. Because backpropagation is used to calculate the gradient descent and the weight values are updated in each iteration of learning. The higher the number of data in this calculation, the more the calculation takes. To solve this problem; before the data is applied to the learning process, it is divided into small groups and applied in this way. In this way, the processing of multiple inputs into pieces is called batch. The value specified as batch parameter when designing the model; means how much data the model can process at the same time.

Batch value can be specified as a value between 1 and the number of all data in the training set. When this value is set to 1, which is the smallest value it can take, “stochastic gradient descent” is made. This means that only one data is processed for each iteration. If the batch value is equal to the number of all elements in the training set, the process will be the same as the “batch gradient descent”, since all data in the training set will enter the training at the same time. Fewer oscillations will appear in the error graphs and the process will take shorter but often no real learning will take place.

Other fixed hyperparameters specified for batch size comparison are as in Table 4.2. Using these parameters, analysis was repeated 10 times with 4, 8, 16, 32 and 64 batch sizes.

Table 4.2: Fixed hyperparameters on LSTM batch size tuning

Epoch	Neuron Sizes	Optimizer
1000	50x75	ADAM

4.1.3. Neuron Sizes

The most important feature that distinguishes the deep learning method from other artificial neural networks, especially in complex problems, is the number of layers and neurons. The concept of depth comes from here. Since the increase in the number of layers reduces the backpropagation effect to reach the first layers, it does not affect much after a certain point. Therefore, the process of determining the correct number of layers emerges as a hyperparameter problem and it is not clear how many layers' structure should be established with the available data.

The number of neurons indicates the number of information stored in each layer. The high number of neurons leads to a significant increase in memory requirement and calculation time. However, the low number of neurons causes underfitting. Just like the number of layers, the number of neurons is a hyperparameter that has serious performance effects.

Other fixed hyperparameters identified are listed in Table 4.3. Along with these parameters, the analysis was repeated 10 times using the number of [50x50], [50x100], [100x100], [100x200] and [200x200] neurons, first and second layers respectively.

Table 4.3: Fixed hyperparameters on LSTM neuron sizes tuning

Batch Size	Epoch	Optimizer
8	1000	ADAM

4.1.4. Optimizer

The learning process in deep learning applications is basically an optimization problem. Various optimization methods are used to find the optimum value in the solution of this problem and each algorithm shows significant differences in performance and speed between each other. Even though stochastic gradient descent is often used in deep learning models, it has been found to be insufficient in many studies. Other methods, such as ADAM or RMSPROP, set the learning rate itself unlike the SGD, so it can change dynamically during training. However, it is impossible to say that these algorithms give the best results in any case.

The learning speed and momentum values of the algorithms used in the optimization test were not changed and only the algorithm differences were tested. Other fixed hyperparameters identified are listed in Table 4.4. With these parameters, ADAM, RMSPROP and SGD optimization methods were repeated 10 times.

Table 4.4: Fixed hyperparameters on LSTM optimizer tuning

Batch Size	Neuron Sizes	Epoch
8	50x75	1000

4.2. GRU

Although the GRU method is a simpler model that saves time compared to LSTM, it gives serious results in performance. Therefore, the second method to be used in the analysis was GRU. The input layer configurations for LSTM also apply to this modeling and are identical to LSTM. Since there are very similar methods, there is no extra process to be done for input data or input layer. It is therefore quite common for most LSTM analyzes to compare with GRU.

Python Keras library is used in GRU modeling. The applications mentioned in the LSTM method were also tested for GRU and the parameters yielding the best results were searched.

4.3. ARIMA

Since the dataset used is a time series, ARIMA which is a traditional statistical algorithm has been tried. ARIMA modeling is widely used with such series even though it does not generally give as good results as deep learning methods.

In order to determine the specified parameters (p, d, q) while defining this method, the model was tried again for all combinations in the range of 0-2 and the final model was obtained with the parameters giving the smallest AIC (Akaike Information Criteria) value.

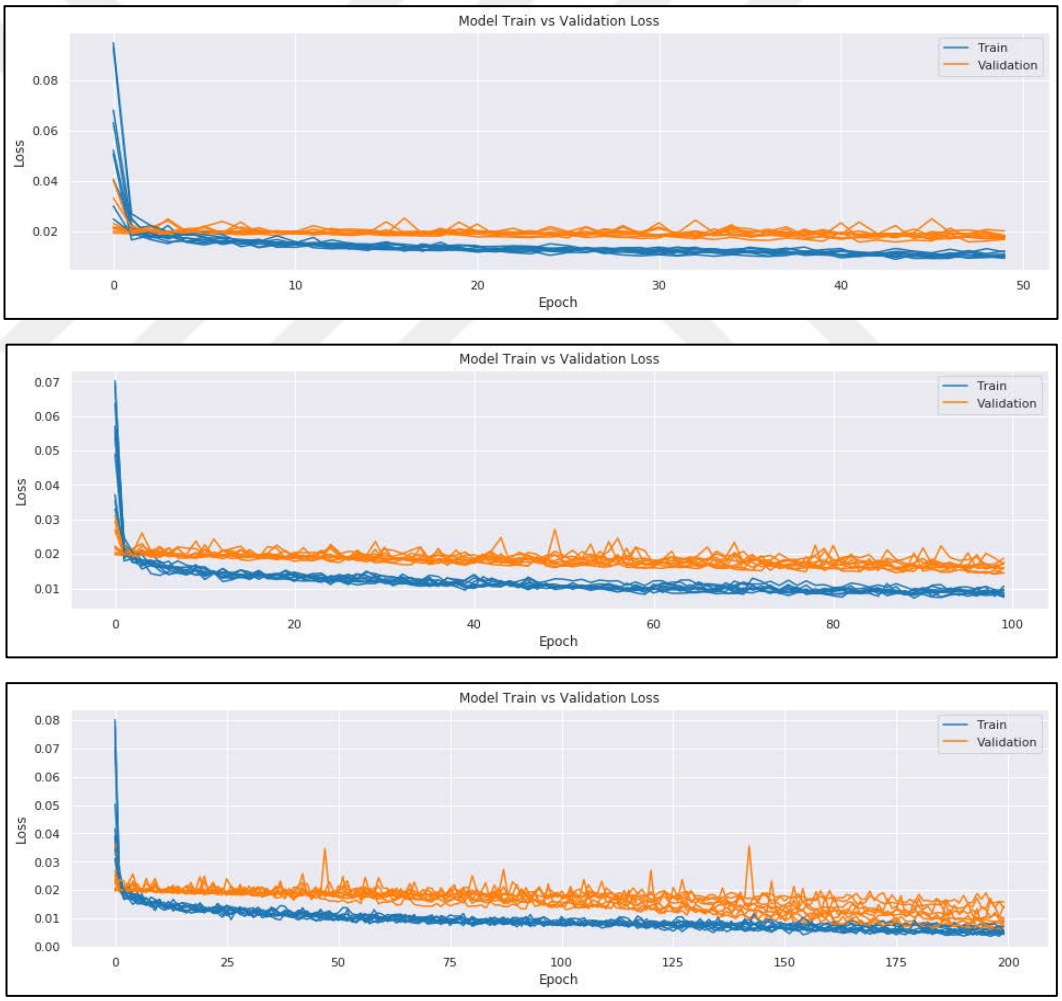
5. RESULTS

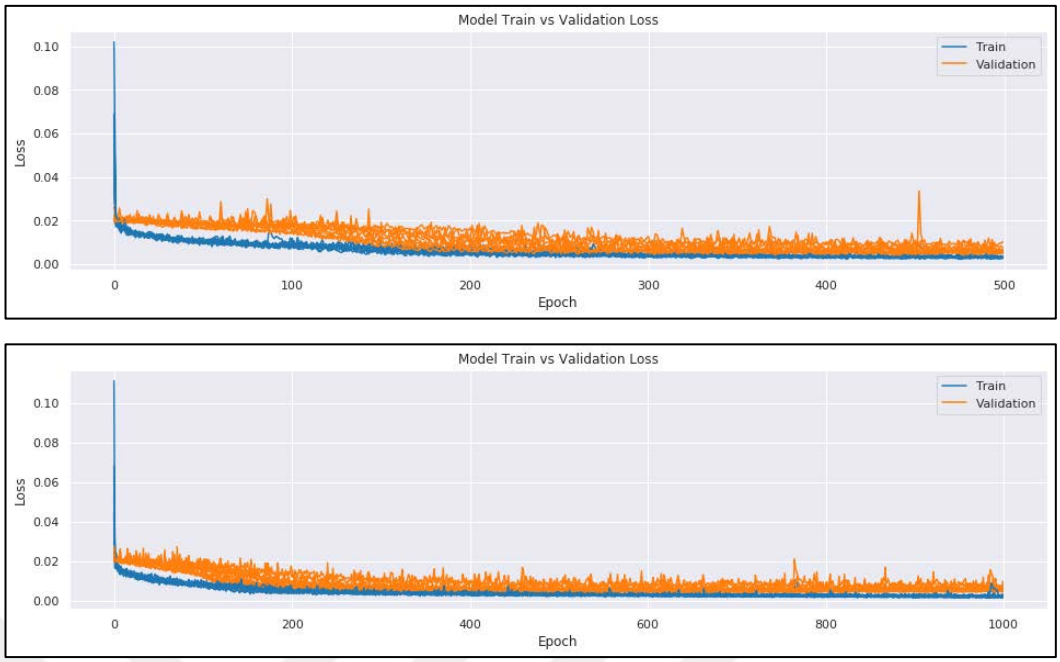
5.1. LSTM

5.1.1. Epochs

In order to find the optimal epoch value, the loss graphs during the model training are shown in Figure 5.1. Looking at the loss graphs of the models obtained with 50, 100, 200, 500 and 1000 epoch, respectively, the decrease continued up to 200 epochs but little change was observed after this point.

Figure 5.1: Train and validation loss graphs on LSTM epoch tuning





In addition, the box graph of the RMSE error values obtained after the evaluation of the models obtained from the experiments with each epoch value using the test set is given in Figure 5.2 and the numerical values are given in Table 5.1. A closer look at these values shows that there is a significant loss in the 50 and 100 epoch values compared to the 500 and 1000 epoch values. Even if the 500 and 1000 epoch results are very close to each other, it can be said that the best result according to the minimum value and average is obtained at 1000 epoch.

Figure 5.2: Boxplot of RMSE scores on test set for LSTM epoch tuning

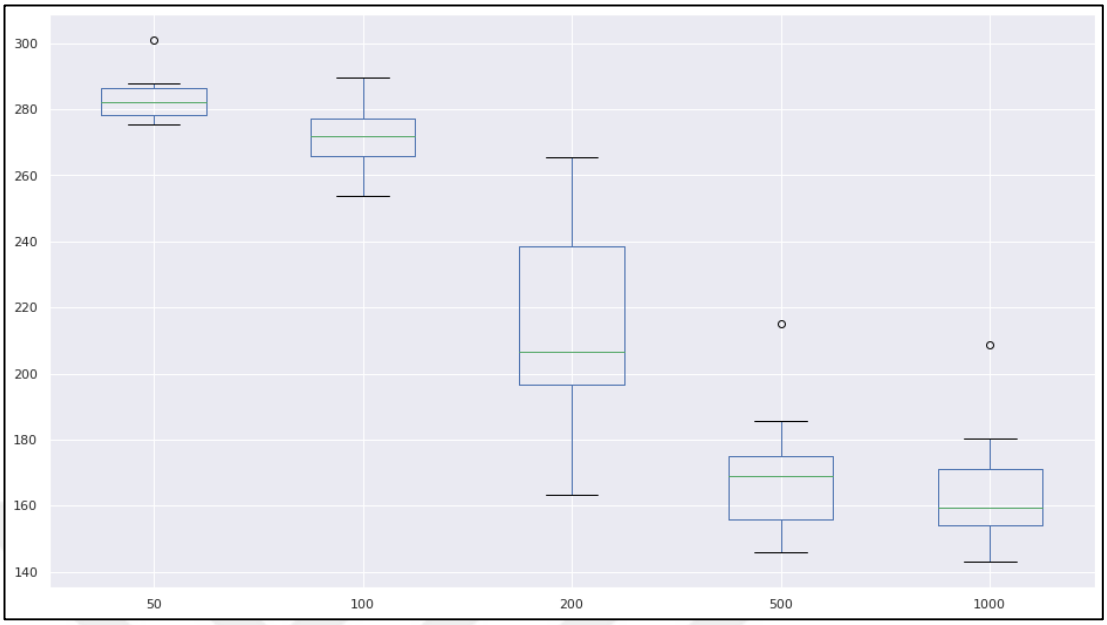


Table 5.1: RMSE scores on test set for LSTM epoch tuning

	50	100	200	500	1000
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	283.520539	270.812690	213.908783	169.990968	164.434183
std	7.460235	11.325412	32.203664	19.743634	19.416149
min	275.266416	253.839944	163.128961	145.917819	143.045504
25%	278.207906	265.722232	196.809879	155.985935	154.031371
50%	282.136338	272.012532	206.663074	168.909331	159.238482
75%	286.438350	277.067459	238.559849	175.136357	171.089739
max	300.873477	289.533909	265.424189	215.194983	208.873126
t (sec)	20.11	35.22	66.32	148.38	285.53

5.1.2. Batch Size

As a result of changes in batch size parameter, error statistics in predictions made with test data set are shown in Figure 5.3 and Table 5.2. It is easy to see that error averages and standard deviations increase as the batch grows. When batch size is 4, it can be said that the median value is the lowest and the lowest mean error occurs at this value. Even

if the smallest RMSE scores of the 50 trials in total were obtained at 16 and 32 batch size values, moving with average and standard deviation results would produce a more accurate result in generalizing the model results. When the model working time and training set RMSE scores were taken into account, the optimum value was determined as 8.

Figure 5.3: Boxplot of RMSE scores on test set for LSTM batch size tuning

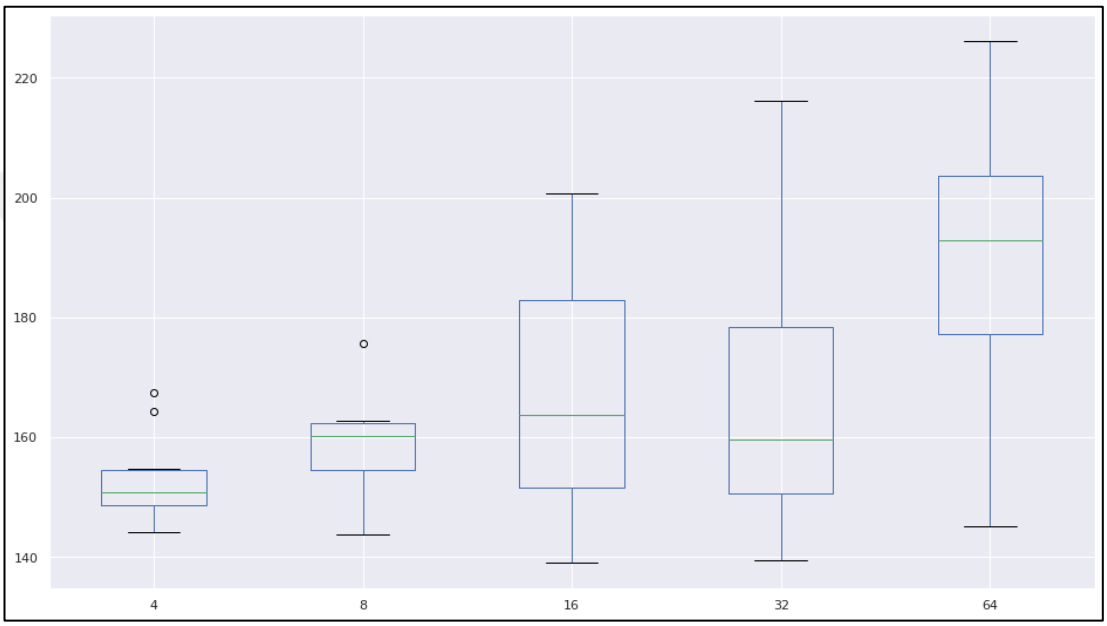


Table 5.2: RMSE scores on test set for LSTM batch size tuning

	4	8	16	32	64
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	153.182355	158.514539	166.519215	165.343546	189.047181
std	7.393169	9.048812	20.866356	24.021260	26.244330
min	144.122504	143.869367	139.080442	139.427379	145.078555
25%	148.681620	154.607908	151.571187	150.624652	177.179728
50%	150.823043	160.184897	163.825079	159.690964	192.928754
75%	154.507492	162.304250	182.990252	178.420388	203.582843
max	167.469270	175.685157	200.713642	216.089866	226.076749
t (sec)	500.53	261.18	141.64	89.77	52.98

5.1.3. Neuron Sizes

The error statistics of the predictions made with the test data set as a result of changes in the number of neurons are shown in Figure 5.4 and Table 5.3. When the results are examined, it is seen that the error scores obtained at 50x50 are in a very high range. Even if the most stable results were obtained at 100x100, when the average and minimum values were taken into consideration, it was observed that the best results were obtained in neuron numbers of 100x200. It would not be wrong to say that there is no improvement in the number of 200x200 neurons compared to 100x200, and even worse results are obtained.

Figure 5.4: Boxplot of RMSE scores on test set for LSTM neuron sizes tuning

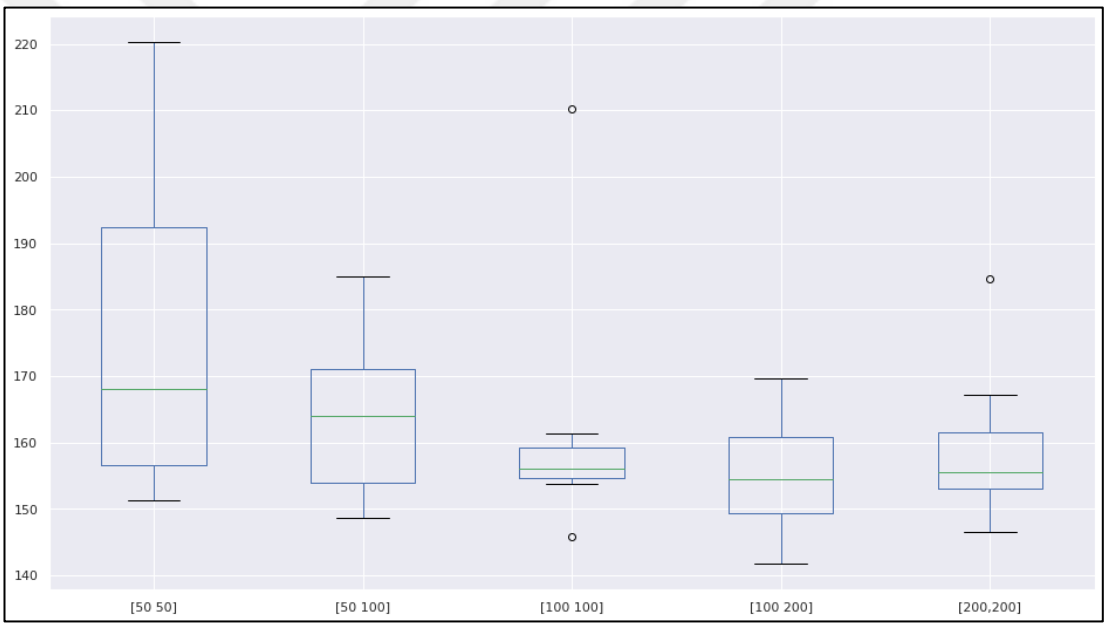


Table 5.3: RMSE scores on test set for LSTM neuron sizes tuning

	[50 50]	[50 100]	[100 100]	[100 200]	[200 200]
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	175.685195	164.172904	161.036149	155.575822	158.840529
std	23.957854	11.629970	17.756128	8.845982	10.792678
min	151.389766	148.660474	145.811042	141.785663	146.503004
25%	156.611914	153.910903	154.603056	149.452513	153.019893
50%	168.102890	164.007033	156.160852	154.483164	155.620396
75%	192.355569	171.094593	159.214115	160.769929	161.466729
max	220.218070	185.058480	210.154045	169.639731	184.699626
t (sec)	244.25	281.66	332.63	584.61	877.69

5.1.4. Optimizer

Figure 5.5 and Table 5.4 show the results of the optimization algorithms. The SGD method has the highest error values among these three methods. ADAM and RMSPROP produced almost the same results. The average values for these two methods do not show much difference, but the minimum and 25 percent threshold values are lower in the ADAM algorithm. Although the standard deviation amount is less in RMSPROP algorithm, it can be said that the best result is obtained from the ADAM algorithm because of the success of ADAM at minimum values.

Figure 5.5: Boxplot of RMSE scores on test set for LSTM optimizer tuning

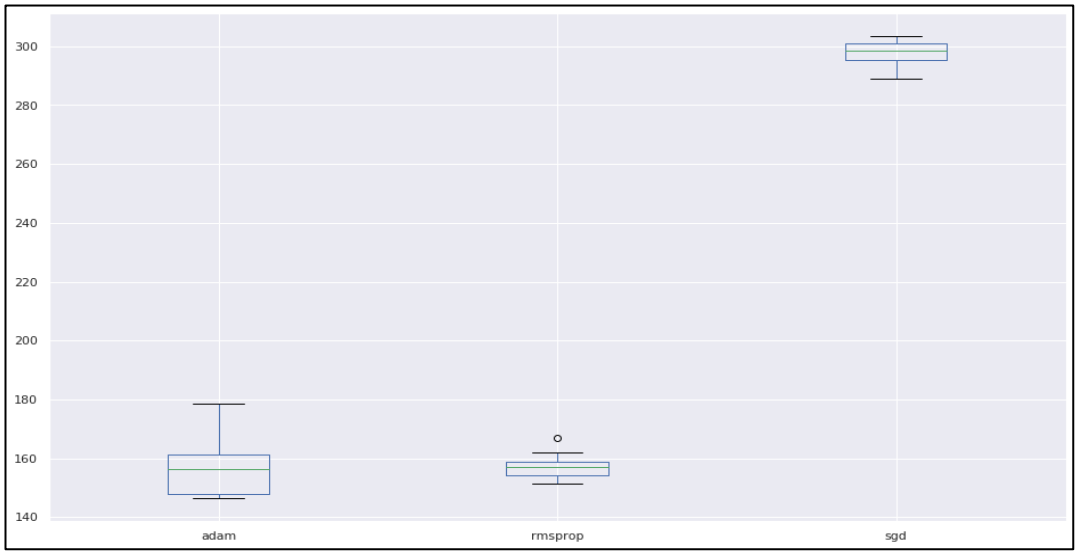


Table 5.4: RMSE scores on test set for LSTM optimizer tuning

	ADAM	RMSPROP	SGD
count	10.000000	10.000000	10.000000
mean	157.538988	157.235942	297.919606
std	11.168737	4.686880	4.462940
min	146.434476	151.333688	288.994125
25%	147.886829	154.082046	295.308935
50%	156.320678	156.919133	298.625578
75%	161.439789	158.896919	301.110326
max	178.692520	166.839376	303.347586
t (sec)	275.37	279.30	260.56

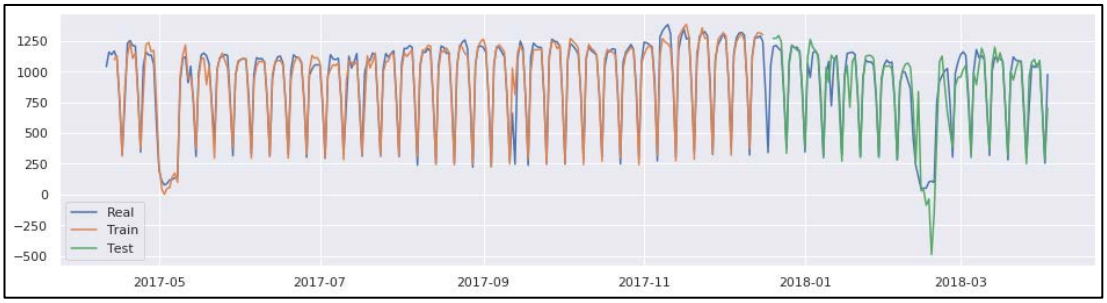
5.1.5. Tuned Model

The hyperparameters that yield the best results from the tuning process are shown in Table 5.5. LSTM modeling using these parameters was done one more time. RMSE scores obtained with training and test datasets were 63.37 and 142.11, respectively. As mentioned before, these scores may vary slightly according to the random initial value, but generally close to this value will be obtained.

Table 5.5: Tuned LSTM hyperparameters

Epochs	Batch Size	Neuron Sizes	Optimizer
1000	8	100x200	ADAM

Figure 5.6: Tuned LSTM predictions



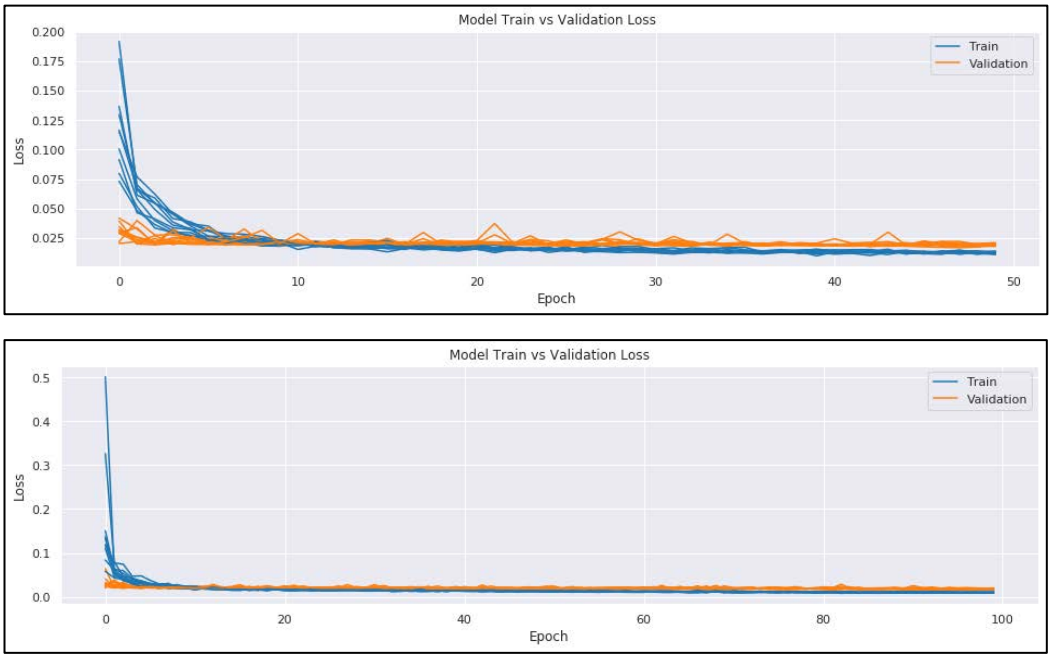
In addition, the estimates made using the latest model generated on the daily power consumption distribution graph are shown in Figure 5.6. The blue color in this graph shows the actual power consumption values in the data set, the orange color is the estimates made by the training set, and the green color is the estimates made by the test set. As can be seen from this graph, the model produced very close results to the actual values in the estimations made with the training set, but it was difficult to capture some points in the test set. It has successfully predicted the fluctuation points in the training set but failed in the test dataset.

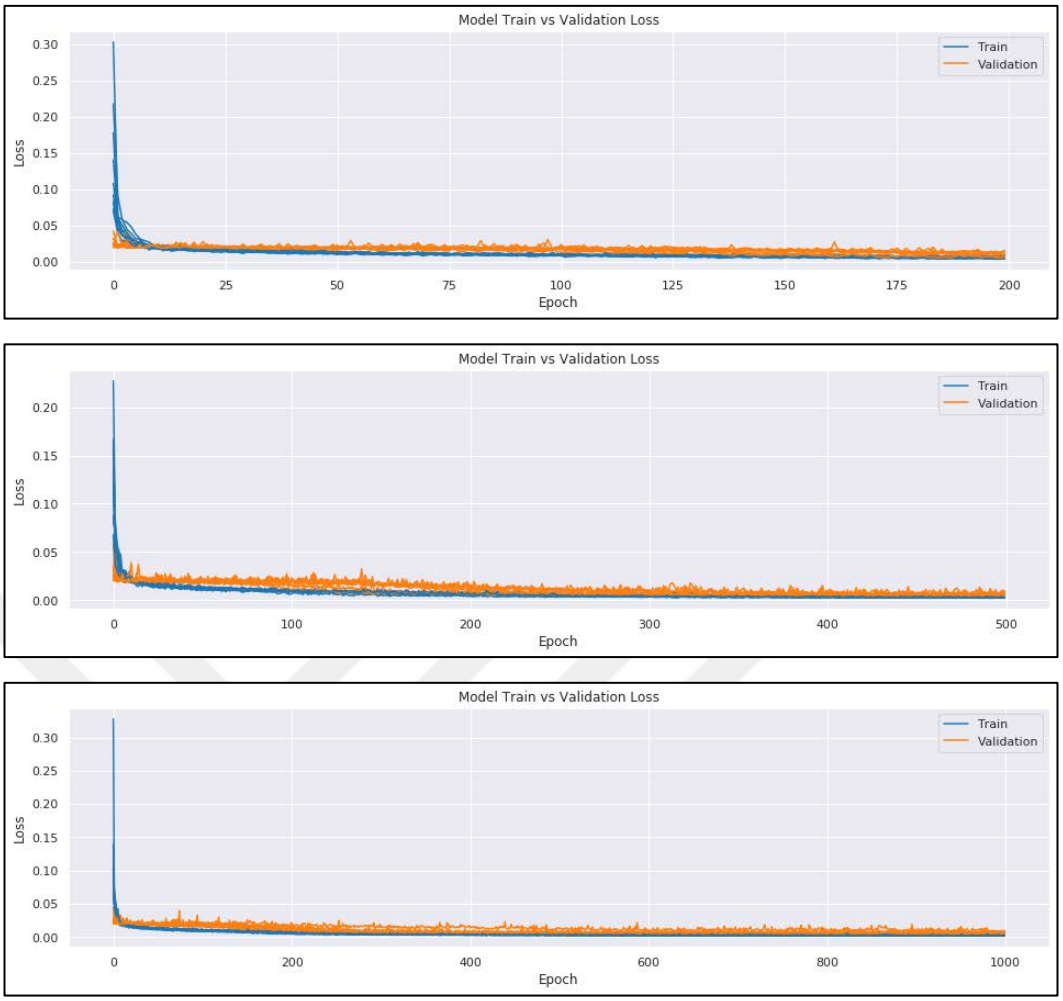
5.2. GRU

5.2.1. Epochs

The loss graphs from the experiments with the determined epoch numbers are as in Figure 5.7. It is seen that the models in all graphs are rapidly reaching saturation. Almost all of the training set losses were found to be lower than the validation set, and after 500 epoch values, there was almost no decrease for both datasets.

Figure 5.7: Train and validation loss graphs on GRU epoch tuning





The RMSE scores in the estimations made with the obtained models are shown in Table 5.6 and the box graph is shown in Figure 5.8. When these values were examined, 50 and 100 epoch values showed very high error amounts, but it was observed that this ratio decreased as the epoch grew. Even if the results of the experiments with 500 and 1000 epochs were very close to each other, it was found that the error rate on average and 25 percent threshold was better for 500 epochs. Even if the standard deviation amount is high for 500 epochs, it can be said that the best results for this dataset can be obtained in the examination made according to minimum values.

Figure 5.8: Boxplot of RMSE scores on test set for GRU epoch tuning

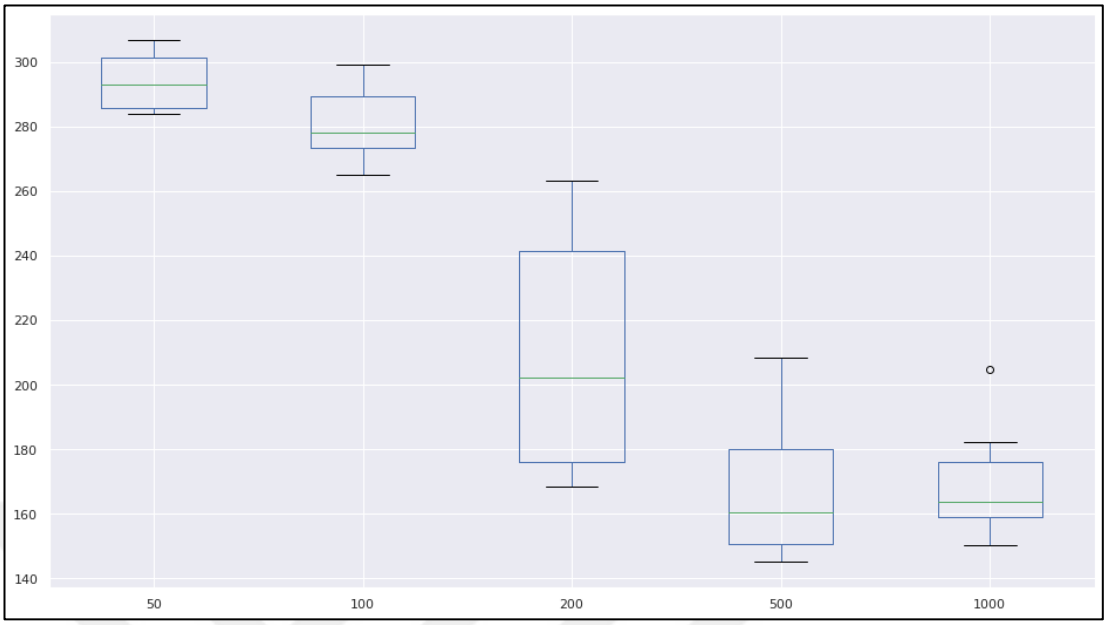


Table 5.6: RMSE scores on test set for GRU epoch tuning

	50	100	200	500	1000
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	293.853688	281.503900	209.513137	166.285670	168.151003
std	8.863664	12.026953	36.662356	21.265344	16.470604
min	283.923186	265.083179	168.695438	145.245438	150.289281
25%	285.756096	273.692475	176.200011	150.557154	159.217694
50%	293.302473	278.147492	202.489678	160.685719	163.789620
75%	301.386614	289.703740	241.575708	179.983464	175.988485
max	306.876463	299.408947	263.340273	208.585559	204.881170
t (sec)	14.05	23.74	44.79	113.67	215.05

5.2.2. Batch Size

Table 5.7 and Figure 5.9 were obtained as a result of the predictions made with the test set of the models obtained from batch size trials. RMSE scores for all batch values can be seen in a very large range. There is a big difference between the maximum and minimum error values for the 8 and 32 batch size parameters. Results for 4 and 16 were very similar.

Since minimum and 25 percent threshold results are quite successful, it is not wrong to say that the best results for this modeling method are obtained at this 16 batch value.

Figure 5.9: Boxplot of RMSE scores on test set for GRU batch size tuning

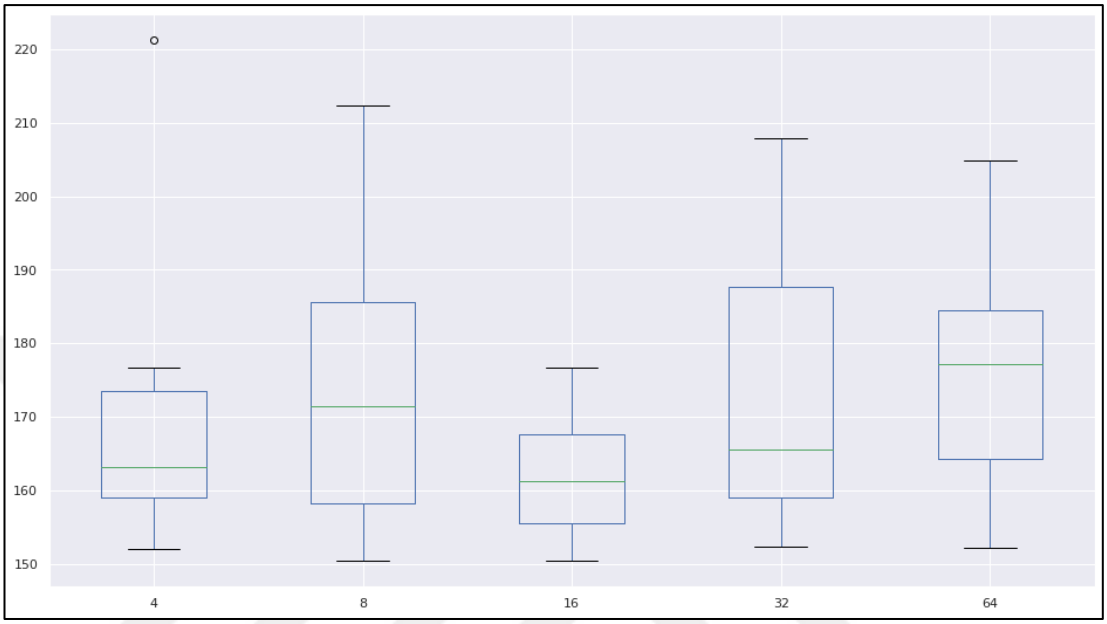


Table 5.7: RMSE scores on test set for GRU batch size tuning

	4	8	16	32	64
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	169.421403	175.167263	162.012748	173.158660	175.909591
std	19.810899	21.415485	8.681756	19.002828	15.441555
min	152.102224	150.421032	150.490196	152.305822	152.156453
25%	159.084510	158.175633	155.578127	159.016863	164.329316
50%	163.127222	171.447831	161.308094	165.577610	177.134362
75%	173.435542	185.573891	167.616149	187.619287	184.535884
max	221.226383	212.388241	176.726579	207.832395	204.875120
t (sec)	400.87	208.14	114.83	74.07	43.93

5.2.3. Neuron Sizes

Figure 5.10 and Table 5.8 show the RMSE scores obtained by evaluating the models obtained from 10 experiments with various neuron numbers with the test data set. As a

result of this evaluation, close values were obtained for all neuron numbers. The minimum and maximum range of 50x50, 50x100 and 200x200 were found to be higher than others. According to the standard deviation and average values, it was determined that 100x200 neuron numbers would give the best results for this modeling method.

Figure 5.10: Boxplot of RMSE scores on test set for GRU neuron sizes tuning

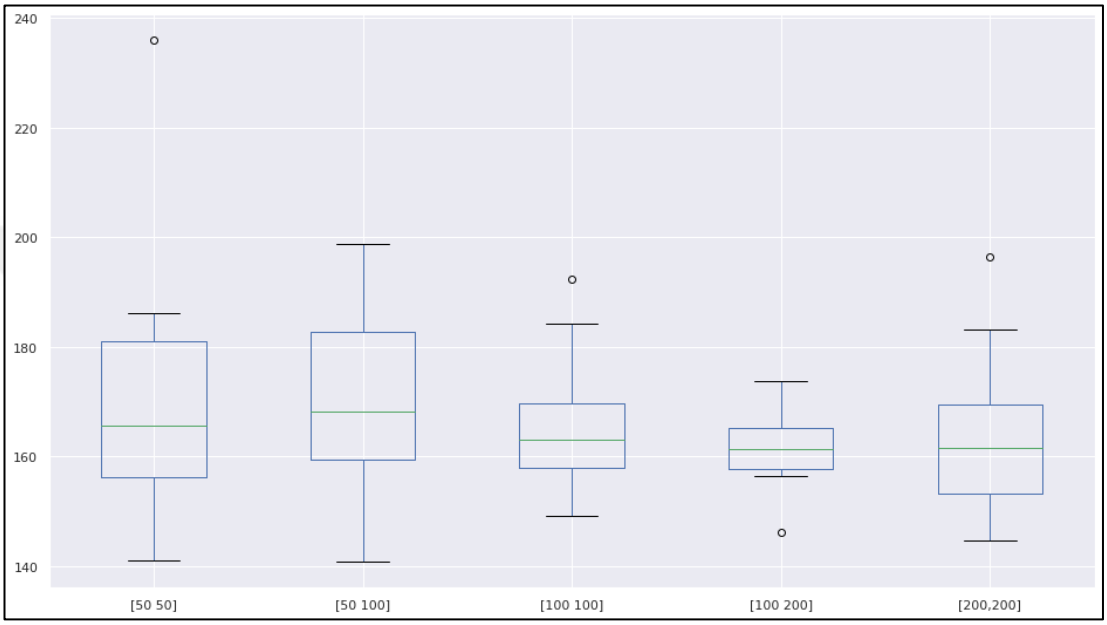


Table 5.8: RMSE scores on test set for GRU neuron sizes tuning

	[50 50]	[50 100]	[100 100]	[100 200]	[200 200]
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	171.421920	170.217221	166.259818	161.709010	164.092121
std	26.792710	17.703515	13.301766	8.067852	16.058728
min	141.085789	140.805947	149.239883	146.077288	144.660984
25%	156.207693	159.388332	157.860563	157.748615	153.129637
50%	165.609998	168.121712	163.090456	161.285290	161.479899
75%	181.042755	182.658302	169.645607	165.232409	169.566665
max	235.960561	198.777702	192.403549	173.659698	196.534449
t (sec)	199.41	227.60	263.09	427.03	629.80

5.2.4. Optimizer

The RMSE scores of the test data evaluations obtained from the optimization methods experiment, the last hyperparameter trial, are as in Figure 5.11 and Table 5.9. When these results are analyzed, it is seen that the SGD method is quite unsuccessful compared to the other two optimization methods. The ADAM algorithm produced model results of 172.42 averages and performed quite successfully compared to the others. Therefore, using ADAM optimization method in GRU modeling with this dataset will give the best results.

Figure 5.11: Boxplot of RMSE scores on test set for GRU optimizer tuning

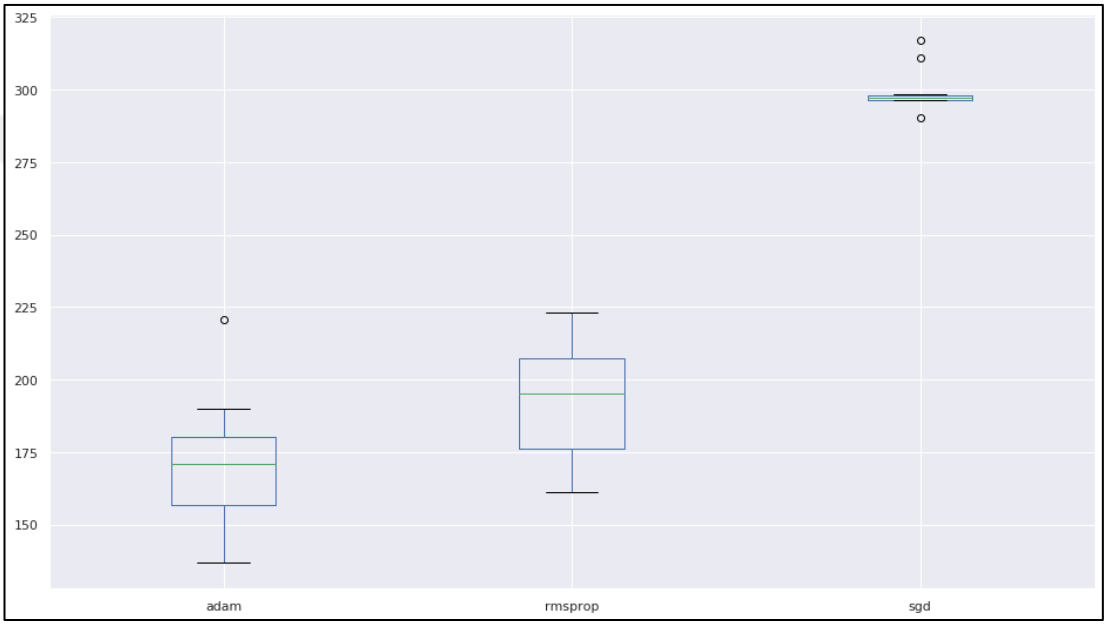


Table 5.9: RMSE scores on test set for GRU optimizer tuning

	ADAM	RMSPROP	SGD
count	10.000000	10.000000	10.000000
mean	171.423307	191.610631	299.792814
std	23.181214	20.832299	7.880646
min	136.819984	161.292229	290.349117
25%	156.851501	176.169125	296.479238
50%	170.730910	195.232235	297.162455
75%	180.331058	207.345136	298.188706
max	220.524742	223.233208	316.896592
t (sec)	197.00	192.56	190.29

5.2.5. Tuned Model

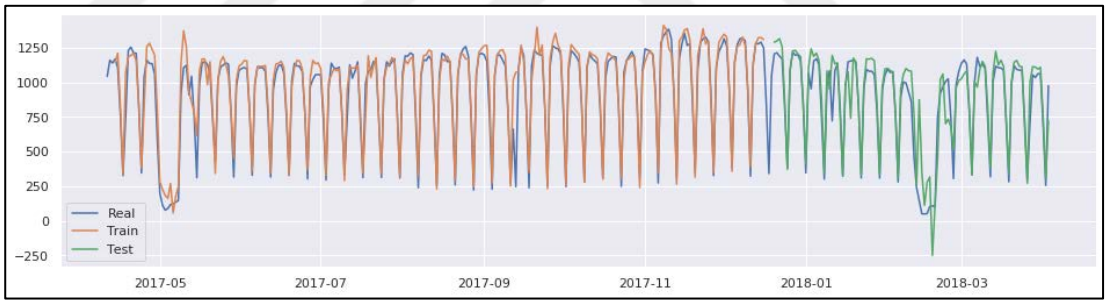
The hyperparameters that yield the best results from tuning are shown in Table 5.10. By using these parameters, GRU modeling was done one more time. RMSE scores obtained with training and test datasets were 98.14 and 158.51, respectively.

Table 5.10: Tuned GRU hyperparameters

Epochs	Batch Size	Neuron Sizes	Optimizer
500	16	100x200	ADAM

Figure 5.12 shows the comparison of the predicted GRU model with the actual data. Like the LSTM, it produced a very successful estimate for the training set at GRU but failed on the test set. It is observed that it performs quite poorly in fluctuation points in both train and test data.

Figure 5.12: Tuned GRU predictions



5.3. ARIMA

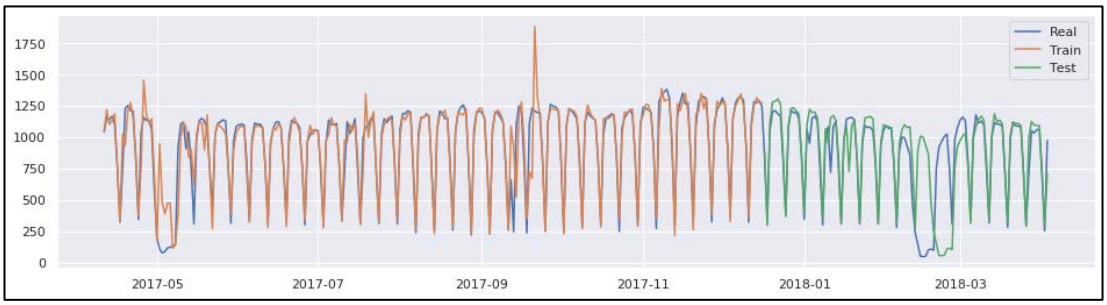
The most appropriate (p, d, q) parameters were also searched for the ARIMA model, which is different from the deep learning methods. The model was reproduced with all combinations from 0 to 2 and the parameters giving the lowest AIC value were searched. The obtained AIC value is as in Table 5.11. While creating this table, parameters with higher AIC values compared to the previous value were not included and only lower combinations were tried to be shown. When looking at this table, it is seen that the best results can be obtained by parameters (2, 0, 1).

Table 5.11: AIC values on ARIMA tuning

p	d	q	AIC
0	0	0	-397.45
0	0	1	-547.89
0	1	0	-584.22
1	0	0	-615.75
1	0	1	-620.03
2	0	0	-620.37
2	0	1	-644.98

As a result of ARIMA modeling using the most appropriate parameters, RMSE scores of 138.33 and 291.22 were obtained in the evaluation made with the training and test set, respectively. In addition, the comparison of the actual values with the estimates made with this model is given in Figure 5.13. Looking at the estimates produced by the test set, it can be said that there was almost no performance at the fluctuation points. This situation has a significant effect on RMSE scores.

Figure 5.13: Tuned ARIMA predictions



5.4. FINAL

Based on this problem, the best results obtained according to the results of time-series analyses (Table 5.12) with 1-year hourly data were obtained by LSTM. While determining the error score, tuning was emphasized and the most important hyperparameters were changed. With the RMSE score of 142.11, 13 percent error was obtained according to the data set with a mean of 1076.

Table 5.12: RMSE comparisons of tuned methods

Method	Train Set RMSE	Test Set RMSE
Tuned LSTM	63.37	142.11
Tuned GRU	98.14	158.51
Tuned ARIMA	138.33	291.22

6. DISCUSSION

It is important to make various estimates using the available data in order to be able to make various plans or arrangements in advance against the situations that may occur in the consumption of energy. As long as the accuracy of these estimations increases, the operations can be carried out more effectively and in a timely manner, thus providing an efficient process in energy consumption.

Although the score obtained from GRU modeling is close to LSTM results, it can be clearly stated that LSTM is successful in comparing deep learning methods. When comparing the two methods, batch size and epoch hyperparameters are observed. The best results were obtained in 8 batch sizes & 1000 epoch for LSTM and 16 batch sizes & 500 epochs for GRU. GRU's unique gate approach and the time-saving hyperparameters that are sufficient for learning make this algorithm different from LSTM.

The ARIMA method, which is widely used in time series analysis, has been quite unsuccessful compared to other methods. The reasons for this are; univariate analysis, lack of trend in the original dataset, inadequacy of the ARIMA method for learning fluctuation points. At the same time, it can be easily emphasized that the ARIMA model cannot make a clear learning process when the results of the estimations made with the test dataset are examined.

It can be said that there is an overfitting situation by looking at the difference between the estimation results with training and test sets for all types of analysis. However, this is due to irregularity in the test set rather than the problem of overfitting. Looking at the graphs showing the predictions, it can be seen that the results that affect the amount of error the most are at the points where these irregularities occur. When all methods tried to apply the order that they learned with the training set with a small and irregular test set, a higher error rate than normal was observed.

Since the data set of the time series type collected was not large enough, an error rate of 13 percent could be concluded. The number of layers used in deep learning methods was

also kept low due to the insufficient number of samples in the data set. The first thing to do to achieve better results should be to collect more data. In this way, it is possible to make weekly or monthly estimations with more downsampling operations on the data instead of daily analysis because the available data is insufficient. At the same time, a significant increase in performance in daily results can easily be predicted.

In order to improve the results, technical data affecting the number of people living in the region may be added to the dataset, except temperature and humidity information. In this way, it is easier for deep learning methods to learn the situations indicated as fluctuations.

7. CONCLUSION

Electricity consumption is of great importance as an increasing need as time goes on. Efficient regulation of the balance between the generation and consumption of this energy source yields substantial financial gains. The fact that it is an energy type that cannot be stored and the amount of consumption is increasing day by day with the advancement in technology requires investment and capacity determination issues in this field. More effective planning in the future is directly related to how much the predictions reflect the reality.

With the advancement of technology, advances in machine learning accelerated and prediction algorithms with low error rates were developed. As mentioned in the literature review, many estimation methods have been tried before and these modeling methods have been compared with each other. Although many studies have achieved very successful results, ways of minimizing error rates have been sought in order to make the necessary adjustments early.

In this study, it is aimed to produce realistic predictions by using deep learning methods to solve the above-mentioned problems. A daily consumption estimate was made with the time series dataset, which includes hourly electricity consumption within a year. After explaining the features such as seasonality and trend of the time series in detail, the preparatory processes such as resampling, normalization and reshape required for the analysis phase were completed. As a method of analysis, LSTM and GRU algorithms which are one of the most popular methods of deep learning and the ARIMA which is frequently used in time data analysis has been tested. During these trials, hyperparameter improvements affecting prediction were emphasized. For each type of modeling, the parameters that produced the best results were searched and the most suitable parameters for each model were compared with the results.

As a result of the tests and comparisons, the best results were obtained with LSTM and a daily electricity consumption estimate was obtained with a 13 percent error rate. Although the GRU method has a close error rate, it is lagging behind LSTM. ARIMA, on the other

hand, showed a rather unsuccessful performance compared to the other two deep learning methods and proved to be inadequate for this prediction process.

Although the result of the study can produce very good estimates of electricity consumption for this dataset, more samples and independent variables are needed to achieve more reliable success rates. Looking at the results obtained, the first process that should be performed for other researchers who will work in this field can be described as data collection. It is inevitable that when the number of samples is increased and other factors affecting consumption are added, significant-good results can be achieved.

REFERENCES

[1] Y. Akan and S. Tak, ‘TÜRKİYE ELEKTRİK ENERJİSİ EKONOMETRİK TALEP ANALİZİ’, *Atatürk Üniversitesi İktisadi Ve İdari Bilim. Derg.*, vol. 17, no. 1–2, p. , Nov. 2010.

[2] D. Aydın and H. Toros, ‘Prediction of Short-Term Electricity Consumption by Artificial Neural Networks Using Temperature Variables’, p. 6.

[3] J. W. Taylor, L. M. de Menezes, and P. E. McSharry, ‘A comparison of univariate methods for forecasting electricity demand up to a day ahead’, *Int. J. Forecast.*, vol. 22, no. 1, pp. 1–16, Jan. 2006.

[4] M. I. Ullah, ‘Component of Time Series Data’, *Basic Statistics and Data Analysis*, 15-Jun-2014. .

[5] ‘Cyclic and seasonal time series | Rob J Hyndman’, 24-Nov-2019. [Online]. Available: https://robjhyndman.com/hyndsight/cyclicts/. [Accessed: 24-Nov-2019].

[6] MDUtheintactone, ‘Components of time series Viz. Secular trend, Cyclical, Seasonal and irregular variations’, *Home | Management*, 12-May-2019. .

[7] *Chapter 6 Time series decomposition | Forecasting: Principles and Practice*. 2019.

[8] Ü. Ç. Büyükşahin and Ş. Ertekin, ‘Improving forecasting accuracy of time series data using a new ARIMA-ANN hybrid method and empirical mode decomposition’, *Neurocomputing*, vol. 361, pp. 151–163, Oct. 2019.

[9] W. S. Mcculloch and W. Pitts, ‘A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY’, p. 17.

[10] A. Géron, ‘Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems’, p. 718.

[11] L. Deng and D. Yu, *Deep Learning: Methods and Applications*. now, 2014.

[12] D. Ciresan, U. Meier, and J. Schmidhuber, ‘Multi-column deep neural networks for image classification’, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[13] ‘Programming a Deep Neural Network from Scratch using MQL Language’, 26-Nov-2019. [Online]. Available: https://www.mql5.com/en/blogs/post/724245. [Accessed: 26-Nov-2019].

[14] ‘Understanding LSTM Networks -- colah’s blog’, 26-Nov-2019. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 26-Nov-2019].

[15] S. Hochreiter and J. Schmidhuber, ‘Long Short-Term Memory’, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[16] E. Gonzalez-Romera, M. A. Jaramillo-Moran, and D. Carmona-Fernandez, ‘Monthly Electric Energy Demand Forecasting Based on Trend Extraction’, *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1946–1953, Nov. 2006.

[17] F. J. Nogales, J. Contreras, A. J. Conejo, and R. Espinola, ‘Forecasting next-day electricity prices by time series models’, *IEEE Trans. Power Syst.*, vol. 17, no. 2, pp. 342–348, May 2002.

[18] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, ‘ARIMA Models to Predict Next-Day Electricity Prices’, *IEEE Power Eng. Rev.*, vol. 22, no. 9, pp. 57–57, Sep. 2002.

[19] G. Gao, K. Lo, and F. Fan, ‘Comparison of ARIMA and ANN Models Used in Electricity Price Forecasting for Power Market’, *Energy Power Eng.*, vol. 09, no. 04, pp. 120–126, 2017.

[20] Y. T. Chae, R. Horesh, Y. Hwang, and Y. M. Lee, ‘Artificial neural network model for forecasting sub-hourly electricity usage in commercial buildings’, *Energy Build.*, vol. 111, pp. 184–194, Jan. 2016.

[21] S. Ferlito *et al.*, ‘Predictive models for building’s energy consumption: An Artificial Neural Network (ANN) approach’, in *2015 XVIII AISEM Annual Conference*, Trento, Italy, 2015, pp. 1–4.

[22] J. P. S. Catalao, S. J. P. S. Mariano, V. M. F. Mendes, and L. A. F. M. Ferreira, ‘An Artificial Neural Network Approach for Short-Term Electricity Prices Forecasting’, in *2007 International Conference on Intelligent Systems Applications to Power Systems*, Kaohsiung, Taiwan, 2007, pp. 1–6.

[23] Y. Wang, Y. Liu, M. Wang, and R. Liu, ‘LSTM Model Optimization on Stock Price Forecasting’, in *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Wuxi, China, 2018, pp. 173–177.

[24] D. L. Marino, K. Amarasinghe, and M. Manic, ‘Building energy load forecasting using Deep Neural Networks’, in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, Italy, 2016, pp. 7046–7051.

[25] Z. Chang, Y. Zhang, and W. Chen, ‘Effective Adam-Optimized LSTM Neural Network for Electricity Price Forecasting’, in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2018, pp. 245–248.

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, ‘Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling’, *ArXiv14123555 Cs*, Dec. 2014.

[27] G. Xiuyun, W. Ying, G. Yang, S. Chengzhi, X. Wen, and Y. Yimiao, ‘Short-term Load Forecasting Model of GRU Network Based on Deep Learning Framework’, in *2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2)*, Beijing, 2018, pp. 1–4.

[28] L. Sehovac, C. Nesen, and K. Grolinger, ‘Forecasting Building Energy Consumption with Deep Learning: A Sequence to Sequence Approach’, in *2019 IEEE International Congress on Internet of Things (ICIOT)*, Milan, Italy, 2019, pp. 108–116.

[29] ‘Weather API | JSON Weather | World Weather Online’. [Online]. Available: https://www.worldweatheronline.com/developer/. [Accessed: 15-Dec-2019].