

**A NEURAL NETWORK FORECASTING
APPROACH FOR THE SMART GRID
DEMAND RESPONSE MANAGEMENT
PROBLEM**

BY

SARA ABDULNASSER AL-ABDALLAH

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

MATHEMATICS

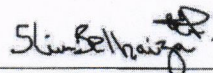
MAY 2023

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

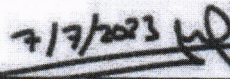
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **SARA ABDULNASSER AL-ABDALLAH** under the direction of her thesis adviser and approved by her thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN MATHEMATICS**.

Thesis Committee



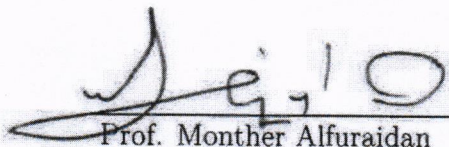
Dr. Slim Belhaiza (Adviser)


7/7/2023

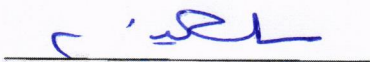
Dr. Mohammed Alshahrani (Member)



Dr. Wessam Mesbah (Member)



Prof. Monther Alfuraidean
Department Chairman


Prof. Suliman Al-Homidan
Dean of Graduate Studies

Date



©Sara Al-Abdallah
2023

Dedication

I dedicate my thesis work to my family. You all have been supportive, encouraging, and interested in my thesis work. I am especially grateful to my dear parents, whose words of support and encouragement keep inspiring me, who have always loved me without conditions and taught me to work faithfully for the goals I strive to obtain. Mom thanks for being there for me throughout the entire master's program. Thanks for listening to my problems and providing perspective.

This work is also dedicated to my husband, who has been a constant source of support and motivation during the challenges of graduate studies. Thank you for your understanding and supporting. My wonderful kids (Faisal and Masa), I am truly thankful for having you in my life during this journey. I would not be who am I today without you all.

Finally, I give special thanks to my friends who have supported me throughout the process. I will always appreciate all they have done.

ACKNOWLEDGMENTS

I am so glad to have completed my graduate work at KFUPM. Therefore, there are a lot of people to thank for their contributions to my success.

First of all, I would like to thank all the faculty and staff members of the Department of Mathematics and the Deanship of Graduate Studies at KFUPM. In particular, I would like to send my special thanks to the department Chairman Prof. Monther Alfuraidan, to the Dean of Graduate Studies Prof. Suliman Al-Homidan, and all my course instructors. They deserve a lot of gratitude for their assistance throughout my master's program, which has been essential to this thesis's achievement.

I send my gratitude to my thesis advisor Dr. Slim Belhaiza for his valuable advice and support. He provided me with a lot of guidance and encouragement as I finished this study and wrote this thesis.

I also want to thank my thesis committee members, Dr. Mohammed Alshahrani and Dr. Wessam Mesbah, for their support in this work and for giving very useful feedback and recommendations.

Finally, I want to express my appreciation to everyone who has helped me during this journey.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
ABSTRACT (ENGLISH)	xiii
ABSTRACT (ARABIC)	xiv
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 LITERATURE REVIEW	4
2.1 Model Categories	5
2.2 Time Series Models	7
2.3 Auto-Regressive Models	9
2.4 Machine Learning Models	11
CHAPTER 3 PRE-PROCESSING AND DATA FORMATTING	15
3.1 Data Sets	15
3.2 Data Cleaning	16
3.3 Variance and Component Analysis	18
3.3.1 Covariance Matrix	18
3.3.2 Singular Value Decomposition	23
3.3.3 Principal Component Analysis	26
3.4 Trend and Seasonality	34

3.5	K-Means Clustering	37
CHAPTER 4 ARTIFICIAL NEURAL NETWORK FORECASTING		
	APPROACH	43
4.1	Statistical Models and Network Architecture	44
4.1.1	Statistical Models	44
4.1.2	Network Architectures	48
4.2	Gradient Descent Methods	49
4.2.1	SGD	52
4.2.2	Adam	53
4.2.3	Adagrad	55
4.2.4	Adamax	55
CHAPTER 5 FORECASTING EXPERIMENTAL RESULTS		57
5.1	Forecasting Results on Data Set 1	58
5.2	Forecasting Results on Data Set 2	60
5.3	Summary of Results	67
CHAPTER 6 CONCLUSION		69
REFERENCES		71
VITAE		77

LIST OF TABLES

2.1	Top 10 rank of the forecasting techniques and models	10
3.1	Covariance matrix on data set 1	19
3.2	Covariance matrix on data set 2, residential area 1	20
3.3	Covariance matrix on data set 2, residential area 2	20
3.4	Covariance matrix on data set 2, residential area 3	21
3.5	Covariance matrix on data set 2, residential area 4	21
3.6	Covariance matrix on data set 2, residential area 5	22
3.7	Covariance matrix on data set 2, residential area 6	22
5.1	Three-Layer ANN Forecasting Results on Data Set 1	58
5.2	Forecasting Results on Residential 1	60
5.3	Forecasting Results on Residential 2	61
5.4	Forecasting Results on Residential 3	63
5.5	Forecasting Results on Residential 4	64
5.6	Forecasting Results on Residential 5	65
5.7	Forecasting Results on Residential 6	67

LIST OF FIGURES

1.1	Smart grid as a network of intelligent devices	2
2.1	Differences between AI - ML - Deep Learning	12
3.1	Explained variance plot data set 2	31
3.2	Explained variance plot data set 1	34
3.3	Additive Trend-Seasonality Decomposition Results	37
3.4	Elbow curve data set 1	41
3.5	3 clusters on data set 1	42
3.6	4 clusters on data set 1	42
4.1	Inputs for Time Series Model	46
4.2	Inputs for Auto-Regressive Model	47
4.3	Inputs for Hybrid Model	48
4.4	ANN First layer	50
5.1	Sample Measurement vs. Prediction on Data Set 1	59

THESIS ABSTRACT

NAME: Sara Abdulnasser Al-Abdallah

TITLE OF STUDY: A NEURAL NETWORK FORECASTING APPROACH
FOR THE SMART GRID DEMAND RESPONSE MAN-
AGEMENT PROBLEM

MAJOR FIELD: Mathematics

DATE OF DEGREE: May 2023

Demand response management (DRM) is crucial to the development of smart grids in the future. The ability to precisely estimate the electricity demand of individual houses is critical to the best operation and planning of the power system. Forecasting the needed components is essential because it can significantly affect the final cost, help to reduce risks, and make better decisions. This thesis proposes a Neural Network forecasting approach for smart grid demand side management. It explores different improved ANN architectures for the forecasting of smart grid consumption. Experimental results show the ability of the proposed neural network framework to provide accurate and reliable energy demand forecasting.

ملخص الرسالة

الاسم:

سارة عبد الناصر العبد الله

عنوان الدراسة:

طريقة التنبؤ من خلال الشبكة العصبية لمشكلة إدارة الاستجابة للطلب على الشبكة الذكية

التخصص:

الرياضيات

تاريخ الدرجة العلمية: ذو القعدة 1444

تعد إدارة الاستجابة للطلب على الطاقة أمراً بالغ الأهمية لتطوير الشبكات الذكية، حيث تعد القدرة على تقدير الطلب على الكهرباء من المنازل مهمة جداً لتحقيق أفضل تشغيل وتخطيط لنظام الطاقة. إن التنبؤ بالعناصر المطلوبة يعد مهمة ضرورية لأنه يمكن أن يؤثر بشكل كبير على التكلفة النهائية و يساعد في تقليل المخاطر و اتخاذ قرارات أفضل. تقترح هذه الرسالة طريقة للتنبؤ من خلال شبكة العصبية لإدارة جانب الطلب على الطاقة في الشبكات الذكية. كما تستكشف بنى محسنة و مختلفة للشبكة العصبية الصناعية لإدارة جانب الإستهلاك في الشبكات الذكية. و تظهر من خلال النتائج التجريبية قدرة إطار الشبكة العصبية المقترح على توفير تنبؤ دقيق وموثوق للطلب على الطاقة.

CHAPTER 1

INTRODUCTION

Smart grids ensure that systems operate safely efficiently and reliably, and also help reduce power loss in the electricity network. Modern smart grids encounter several economic and technical obstacles in their quest to deliver energy to customers securely and cost-effectively. Among the most significant of these obstacles are: load-flow analysis, scheduling, and electric energy system control. An accurate prediction model is needed for the power system's best operation and planning. During the last decade, load forecasting has become one of the most developing fields of interest and research on smart grids. Several forecasting techniques for power system load have been suggested and the usage of mathematical models seems to have been successful. Load forecasting techniques aim to minimize the estimation errors between predicted and measured future values in energy demands. The importance of smart grids' demand forecasting is as follows:

- Reduce unit production costs and preserve the efficiency of power facilities.
- It can be used to monitor high-risk maintenance operations and energy reserves.

- Provide the most significant data for planning and power delivery.



Figure 1.1: Smart grid as a network of intelligent devices

This thesis aims to select the most accurate forecasting model and neural network architecture to apply in different real-life cases. The selected forecasting model and the neural network would aid in energy conservation, demand, and supply management, and efficient financial support planning for users starting their power generation. For this, obtaining precise results, achieving maximum performance during training, and minimizing the overall prediction error are the three key challenges facing artificial neural networks (ANN). To do so, this thesis explores three different statistical models:

- (1) Time series model where several input measurements (hour, period, day, season, month, number of appliances) are used to forecast energy consumption.
- (2) Auto-regressive model where older energy consumption (within a single period range) is used to forecast energy consumption.
- (3) Hybrid auto-regressive model where input measurement and older energy consumption are used to forecast energy consumption.

The performance of the three different models is tested using different multi-layer neural network architectures. The proposed ANN framework is tested on two different real-life smart grid data sets to come out with general recommendations.

Chapter 2 describes the smart grid demand response management (DRM) problem and reviews some important mathematical models and machine learning models proposed for DRM in the literature. Chapter 3 details the data formatting and pre-processing methods used in our context. Chapter 4 describes the statistical models and the artificial neural network (ANN) approach for the DRM. Chapter 5 presents the experimental result obtained with the three different models using different ANN architectures on two smart grid data sets. Finally, chapter 6 summarizes and concludes the thesis.

CHAPTER 2

LITERATURE REVIEW

Demand response management (DRM) plays a key role in the future of smart grids by improving different aspects of both supply and demand sides. Demand forecasting in the power sector has become an important part of modern demand management and response systems. In smart grids, electric load forecasting is critical, it is about predicting power demand using forecasting techniques. Therefore, an accurate prediction model is required for optimal power system operation and planning. Short-term electric load forecasting predicts the load for the next few hours to weeks. Accurate forecasting is difficult in small-scale power systems because the short-term electric load time series is nonlinear and non-stationary.

The use of mathematical models has been shown to be successful for DRM. Various strategies for forecasting power system load have been proposed in recent decades.

2.1 Model Categories

One of the most common model categorizations distinguishes between linear and non-linear models [1].

Linear models are separated into statistic time series and dynamic time series models. Dynamic models are such that factors and random inputs are taken into consideration dynamically. They make use of time series for modeling dynamical behavior. They are divided into auto-regressive and moving average models and state-space models.

- Auto-Regressive and Moving Average (ARMA) models combine auto-regressive and moving average models. Auto-regressive models use the previous values to forecast future values. Moving average (MA) models calculate the residuals or errors of previous values and determine the future values. In ARMA models, the residuals and the effects of previous values are taken into account when predicting future values. Many modifications to the ARMA model can be found in the literature under other names like Auto-Regressive Integrated Moving Average (ARIMA), which is quite similar to the ARMA model in the use of previous values and residuals to predict future values, other than the fact that it includes one more factor known as Integrated (I). The integration component in the ARIMA model converts the non-stationary data into stationary data, where integration is the number of times needed to differentiate a series to achieve stationarity. This model combines several changes that have already been made to the model to keep it stationary. ARMA models work well on stationary data, whereas the

ARIMA model works well on non-stationary data.

- State-space models are used when dealing with dynamic time series issues. They use a set of input, output, and state variables to represent a physical system mathematically. The state variables are employed to describe a system with a set of first-order differential equations. State-space models are commonly used to analyze ecological and biological time-series data.

Non-linear models are more likely to be chosen when more prediction variables are combined in a non-linear way to model observable data. Non-linear models are further divided into two categories: Support vector machine-based (SVM) models and Markov chain-based models.

- Support vector machine-based (SVM) models are one of the most widely used algorithms for supervised learning. They are applied to both classification and regression issues. The SVM algorithm's objective is to establish the best line or decision boundary that can divide the space with n dimensions into classes which help in classifying the new data. Models Works based on the support vector machine are more accurate.
- Markov chain-based models are stochastic models that represents a series of probable events where each event's based on the state obtained in the preceding event. They are frequently employed to forecast economic events, such as market declines.

Meanwhile, due to their complexity, developing non-linear models is extremely difficult.

The following subsections focus on time series analysis and auto-regressive techniques.

2.2 Time Series Models

Time series analysis is a particular method of investigating a set of data points gathered over a period of time. Models of time series analysis include the following:

- Classification: categorizes the data and assigns those categories.
- Curve fitting: studies the relationships between the variables within the data by plotting the data along a curve.
- Descriptive analysis: identifies trends, cycles, or seasonal variation in time series data.
- Explanative analysis: makes an effort to comprehend the data, and the links between the data.
- Exploratory analysis: highlights the key elements of the time series data.
- Forecasting: Predicts future data.
- Intervention analysis: investigates the impact of an event on the data.
- Segmentation: separates the data into segments to display the fundamental properties of the source data.

Time series forecasting is the analysis of time series data using statistics and modeling to generate predictions. The forecast is generally based on data with a historical time stamp.

Time series forecasting techniques can be categorized according to numerous parameters [2]. Categorization according to the **forecasting horizon** distinguishes between the time frame of the forecast :

- Very short-term load forecasting (VSTLF): from a few seconds to several hours.

These models are commonly used in flow management.

- Short-term load forecasting (STLF): ranging from hours to weeks. These models are commonly used to balance supply and demand.
- Medium-term and long-term load forecasting (MTLF and LTLF): between months and years normally. These models are used to plan resource usage.

The magnitude of the variables employed is the key distinction between these three forecasting horizons, not taking into consideration the model that was used [3]. Meanwhile, there are some limitations on time series forecasting. For instance, time series are not useful for all situations as not all models can fit all sets of data. It is up to data teams and analysts to understand the limits of their analysis and what their models can support. Auto-regressive techniques are easier to implement and less expensive to develop.

2.3 Auto-Regressive Models

Applying prediction models that rely on explanatory factors to forecast the load on systems is expensive and time-consuming. Using auto-regressive methods that rely only on the earlier evolution of the examined parameter lowers costs and saves time. A comprehensive table outlining the findings of the review of 47 publications detailing 264 forecasting models from 1997 to 2018 is presented in [4]. It summarizes the status of research on short-term power demand forecasting for power systems using auto-regressive and non-auto-regressive approaches and models. In addition, the authors provide a new method for creating literature reviews when choosing the most probable forecasting models. Due to the chosen forecast quality measure Mean Average Percentage Error (MAPE), the ranking of forecasting models Table 2.1 was used in the proposed new approach. That study was designed to provide an answer to the question of whether using auto-regressive approaches can improve short-term forecasting of electricity demand in power systems while remaining inexpensive and simple to apply. Analysis was conducted on the effectiveness of the forecasting models as determined by the MAPE measure. To compute it, we add all the absolute percent errors together and divide the sum by the number of errors. In this categorization, the ten lowest percentage results were chosen. Table 2.1 lists the top 10 forecasting techniques and models: DEA, FR, GRM, GA, ANFIS, ANN, FGRM, WANN, ANN, and FL, where the repeated ANN belongs to different models [4].

FGRM and GRM employ the explanatory variables, while the remaining eight models were auto-regressive. To establish whether auto-regressive approaches are still

Table 2.1: Top 10 rank of the forecasting techniques and models

Ranking	Authors	Year	Method, Model	MAPE
1	Kheirkhah <i>et al.</i> [5]	2013	DEA (Data Envelopment Analysis)	0.01%
2	Kheirkhah <i>et al.</i> [5]	2013	FR (Fuzzy Regression)	0.08%
3	Wang <i>et al.</i> [6]	2018	GRM (General Regression Model)	0.10%
4	Kheirkhah <i>et al.</i> [5]	2013	GA (Genetic Algorithm)	0.14%
5	Kheirkhah <i>et al.</i> [5]	2013	ANFIS (Adaptive Neuro Fuzzy Inference System)	0.15%
6	Kheirkhah <i>et al.</i> [5]	2013	ANN (Artificial Neural Network)	0.16%
7	Wang <i>et al.</i> [6]	2018	FGRM (Full General Regression Model)	0.20%
8	Rana <i>et al.</i> [7]	2016	WANN (Wavelet Artificial Neural Network)	0.27%
9	Rana <i>et al.</i> [7]	2016	ANN (Artificial Neural Network)	0.28%
10	Rana <i>et al.</i> [7]	2016	FL(Fuzzy Logic)	0.29%

interesting, certain essential papers between 2019 and 2021 were also included in [4].

The results of the reviews supported the auto-regressive approach's great potential for forecasting power demand. The employment of auto-regressive models may assist the transmission system operator to achieve improved forecasting efficiency.

There are various methods for predicting time series. One of the most common in forecasting is the auto-regressive integrated moving average (ARIMA) model. The advantage of using ARIMA model is that it is simple to use and offers good forecasts over a short period. It has two basic limitations which are listed below:

- Linear limitation: it is supposed that a variable's future value will be a linear function of several previous data and random errors. If the implicit mechanism is nonlinear, the ARIMA models' estimation may be completely unsuitable. However, since non-linearity is a common feature of real-world systems[8], it is illogical to assume that a given implementation of a time series is the result of a linear process.
- Data limitation: for ARIMA models to produce the desired results, a lot of historical data is required. Data limitation dictates that ARIMA models need at least fifty, and preferably 100 or more, data to get the required results.

Using hybrid models or combining multiple models can be an efficient strategy to enhance forecasting performance. The fundamental concept behind model combining in forecasting is to remove its limits in order to create a more comprehensive model with more accurate outcomes. The fundamental ideas of ARIMA, ANNs, and fuzzy regression models are used to formulate a new approach to forecasting. In that model, the special ability of ANNs in nonlinear estimating is employed to go over the ARIMA models' linear limitations and create a more accurate model. Additionally, fuzzy logic is used to get beyond the ARIMA models' data limitations and provide a model which is more adaptable [9].

2.4 Machine Learning Models

First of all, let us clarify the differences between artificial intelligence (AI), machine learning (ML), and deep learning (DL). The following literature review focuses on energy demand prediction using neural network approaches. There are three main groups of algorithms in ML:

- Supervised learning: the process used to generate predictions using a labeled data set, which is split into training and test data sets. This is what we will use in this thesis.
- Unsupervised learning: the process of identifying hidden patterns in an unlabeled data set, like PCA and K-means clustering.
- Reinforced learning: a type of learning that is based on interaction with the

environment.

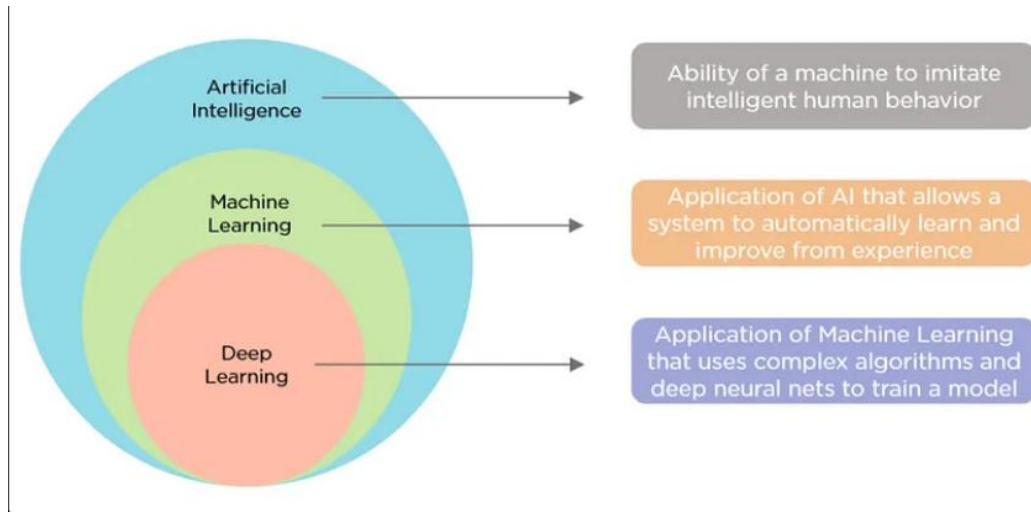


Figure 2.1: Differences between AI - ML - Deep Learning

Neural networks define an artificial intelligence technique that trains computers to analyze data in a manner modeled like a human brain. It is a type of machine learning process, called deep learning, that employs neurons that are linked together in a layered structure that looks like the human brain. Artificial neural networks (ANN) are now widely accepted for pattern recognition and forecasting in real-world problems. Using proper training, evaluation, and checking procedures, neural networks generate an almost perfect solution to these problems. They are used to learn the relationship between past, present, and future temperatures and loads. The main challenge with ANN is to minimize the overall prediction error, obtain precise outcomes, and achieve the highest performance during training. ANN-based models, fuzzy neural network-based models, and stochastic distribution-based models are all examples of NN models.

A recurrent neural network (RNN) is a type of artificial neural network in which

connections between nodes can form a cycle, allowing the output of some nodes to influence subsequent input to the same nodes. Long short-term memory (LSTM) is a sort of recurrent neural network (RNN) that has more memory ability to remember each node's output for a longer time and generate the result for the next node efficiently. LSTM networks solve the long-term reliance issues or vanishing gradients problem of RNNs. For more accurate forecasting, a long short-term memory-based RNN can take advantage of long-term dependencies in the electric load time series. Experiments demonstrated that LSTM-based RNN is capable of forecasting accurately the complex electric load time series with a long forecasting horizon.

Since the second half of the 1980s, non-linear ANN-based models have gotten a lot of interest and academics were able to make significant progress on ANNs. Anderson *et al.* [10] were among them as they introduced a model called Brain-State-in-a-Box, which is founded on the premise that the connections between neurons are reinforced every time they are activated. Teuvu Kohonen presented a refinement of Anderson's model [11]. Fakushima suggested a network model that relied on pattern recognition [12]. John Hopfield described and demonstrated the operation of a network using mathematics [13]. The use of ANN-based machine learning algorithms to estimate electricity consumption dates back to the 1990s, and it is still under investigation today. Park *et al.* [14] provided an artificial neural network (ANN)-based technique to predict electric load. Experimental results on actual smart grid electrical data showed an average absolute error on one-hour and 24-hour ahead forecasts, respectively, were about 1.40% and 2.06% percent. This is compared with the 4.22% percent average error for 24-hour projections based on commonly used forecasting techniques on the

same data.

Amjady *et al.* [15] used a new neural network technique for STLF. They compared ANN Modified Harmony Search (MHS) learning algorithm to other various approaches, and were able to obtain a forecasting framework with excellent short-term load forecasting (STLF) accuracy. Macedo *et al.* [16] investigated the use of DRM in power systems and presented a model that takes data from digital meters to construct load curve patterns. They designed and tested an ANN using these load curve patterns, and categorized new data, and used these specified patterns. Baliyan *et al.* [17] survey artificial neural networks applied to short-term load forecasting.

Muralitharan *et al.* [18] proposed a neural network-based optimization model to forecast energy demand. They used a conventional neural network to forecast energy demand on the customers' side. They tested a genetic algorithm (GA) and Particle Swarm Optimization (PSO) to optimize the weights of the neural network. They observed that PSO was more appropriate for estimating long-term energy demand, and GA was more appropriate for short and medium-term energy demand.

A machine learning (ML)-based forecasting model was used by Li [19] for short-term load forecasting. The model included extensive data mining and multi-step rolling forecasting. A de-noising method based on deconstruction and regeneration was employed to reduce the turmoil of short-term demand.

CHAPTER 3

PRE-PROCESSING AND DATA FORMATTING

This chapter describes our pre-processing and data formatting methods. We first describe our real-life data sets. We then dive into the formatting and pre-processing in four main steps: data cleaning, principal component analysis, trend and seasonality, and K -means clustering.

3.1 Data Sets

This thesis studies two real-life data sets: Toronto Data set 1 and Oklahoma Data set 2. Toronto Data set 1 contains 1082 households energy consumption information collected daily over 36 consecutive months, during the past 3 years: 2019, 2020, 2021. The collected data indicates the geographic coordinates of the household, its current index in the data set, the 'number of electric appliances' inventoried, the 'day order' within the current year, the 'season', the 'day of the week' and the 'period' of the

'measurement'. We have only one observation for each household.

Data set 2 contains over 30000 observations for 6 selected residential areas. Energy consumption information was collected daily over 60 consecutive months, during the past 5 years: 2017, 2018, 2019, 2020, and 2021. The collected data indicates the 'number of electric appliances' inventoried, the 'day order' within the current year, the 'season', the 'day of the week', the 'period', and the 'hour' of the 'measurement'.

The day order is the order of the measurement day in the corresponding year. We divide both data sets into three four-month seasons: Season 1 spans November to February, Season 2 spans March to June, and Season 3 spans July to October. Days of the week are represented by integers, with Monday being assigned the value of 1 and Tuesday being assigned the value of 2, and so on. Time periods are also represented by integers, with 1 indicating the time between 0:00 and 5:59, 2 indicating the time between 6:00 and 11:59, 3 indicating the time between 12:00 and 17:59, and 4 indicating the time between 18:00 and 23:59. Data set 2 measurements are taken over six hours, from 1 to 6, for each time period.

3.2 Data Cleaning

The acquired data needed to be transformed into input-output times series samples to be able to train the model. In particular, for data set 1, we had to clean the inputs of 14 out of 1082 data entries. The 14 data entries had generally 0 in the (kilowatt per hour) KWH electricity consumption column value, most likely due to measurement errors. The entries in the corresponding cells were then replaced by the average values

observed for the demand, particularly taking into account the average demand on similar weekdays, seasons, and periods [20].

Data cleaning is about finding and eliminating mistakes and discrepancies to increase the data's quality. Data entry errors, information gaps, and other types of incorrect data can cause data quality issues. Cleaning the data before working on it is very important. Here are some of its benefits:

- Utilizing several data sources will be simple because we can correct the errors when numerous data sources are involved.
- Allowing users to identify where issues are coming from by monitoring mistakes makes resolving inaccurate data for future applications easier.
- Giving access to the best information possible to take decisions.

The following are the different steps involved in the data cleaning method:

- Eliminating redundant or useless observations is required to get accurate information. Duplicate observations may occur when data sets from many sources are combined.
- Correcting structural mistakes, such as typos, or wrong capitalization.
- Remove unacceptable outliers, which are odd observations that do not seem to fit the data.
- Working on missing data, by filling these missing values based on other observations, or dropping missing-value observations, but the information will be lost in the process.

3.3 Variance and Component Analysis

Numerous applications including control in dynamical systems, power networks, image processing, stability, and data mining, depend on variance and component analysis. Variance analysis aims to find a correlation between features and outputs. Principal component analysis (PCA) aims to find patterns in data collection and then reduce the variables to the most crucial features, without losing any important ones. To conduct PCA, there are certain mathematical and statistical requirements:

- Compute the covariance matrix.
- Find the eigenvectors and eigenvalues of $A^t A$ and AA^t .
- Perform singular vectors decomposition (SVD).

3.3.1 Covariance Matrix

The covariance matrix adds a lot of indications on the relation between the input features (number of appliances, day order, season, day of the week, period) and the outputs we aim to study and forecast, which is consumption (KWH). It is a symmetric square matrix that summarizes the relation between features and consumption by looking at the last row or last column. Positive covariance values mean that power consumption and input features increase together or decrease together. But if the values are negative that means they will act oppositely.

This is how the covariance between two features is mathematically calculated:

$$Cov(X, Y) = \frac{\sum (x - \bar{x})(y - \bar{y})}{N - 1}$$

where $\bar{x}$ and $\bar{y}$ are, respectively, the means of x and y .

The data matrix A has m columns and n rows. If we center this matrix by subtracting the mean from all m entries in each, row the matrix AA^t will contain the variance and covariance values. The variance values located in the diagonal of the covariance matrix are obtained when we find the relation between the feature and itself $Cov(x, x)$. The covariance matrix is then defined by:

$$C = \frac{AA^t}{n - 1}$$

where n is the number of columns or rows. The orthogonal eigenvectors of C are the singular vectors (principal components) of A .

Table 3.1: Covariance matrix on data set 1

Cov.	Longitude	Latitude	NbrAppliances	DayOrder	Season	DayOfWeek	Period	KWH
Longitude	1.000	-0.998	0.159	0.201	0.277	0.220	0.243	0.123
Latitude	-0.998	1.000	-0.159	-0.200	-0.276	-0.218	-0.243	-0.124
NbrAppliances	0.159	-0.159	1.000	0.003	0.016	-0.004	-0.002	0.899
DayOrder	0.201	-0.200	0.003	1.000	0.943	0.068	0.024	-0.053
Season	0.277	-0.276	0.016	0.943	1.000	0.075	0.031	-0.053
DayOfWeek	0.220	-0.218	-0.004	0.068	0.075	1.000	0.003	-0.008
Period	0.243	-0.243	-0.002	0.024	0.031	0.003	1.000	0.015
KWH	0.123	-0.124	0.899	-0.053	-0.053	-0.008	0.015	1.000

The obtained covariance matrix for data set 1 in table 3.1 shows that the covariance between the consumption (KWH) and the number of appliances is the highest. This means that the number of appliances is the feature with the most important effect on consumption. On the other hand, the covariance between the consumption and the

day order or the period has negative values. Which means that they act oppositely. If the day order or season has a small number like in January which belongs to season (1) with low numbers of day order, the consumption will increase because the winter is very cold in Canada. The longitudes and latitudes would not have a direct impact on household electricity consumption.

Table 3.2: Covariance matrix on data set 2, residential area 1

Cov.	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
DayOrder	1.000	0.319	0.346	0.450	0.328	0.204
DayOfWeek	0.319	1.000	0.378	0.485	0.358	0.244
Period	0.346	0.378	1.000	0.524	0.383	0.243
NbrAppliances	0.450	0.485	0.534	1.000	0.463	0.449
Hour	0.328	0.358	0.383	0.463	1.000	0.186
KWH	0.204	0.244	0.243	0.449	0.186	1.000

For data set 2, the obtained covariance matrix for residential area 1 displayed in Table 3.2 shows that the correlation between consumption (KWH) and the number of appliances is the highest. The covariance between the consumption and the other features is less than the covariance with the number of appliances. All the features have a similar positive effect on consumption, but their impact is less than the impact of the number of appliances.

Table 3.3: Covariance matrix on data set 2, residential area 2

Cov	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
DayOrder	1.000	0.485	0.512	0.599	0.487	0.316
DayOfWeek	0.485	1.000	0.546	0.597	0.524	0.327
Period	0.512	0.546	1.000	0.635	0.552	0.330
NbrAppliances	0.599	0.597	0.635	1.000	0.591	0.544
Hour	0.487	0.524	0.552	0.591	1.000	0.284
KWH	0.316	0.327	0.330	0.544	0.284	1.000

For data set 2, the obtained covariance matrix for residential area 2 displayed in Table 3.3 shows that the correlation between consumption (KWH) and the number of appliances is the highest. The covariance between the consumption and all other features is less than the covariance with the number of appliances. All the features have a similar positive effect on consumption, but their impact is less than the impact of the number of appliances.

Table 3.4: Covariance matrix on data set 2, residential area 3

Cov	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
DayOrder	1.000	0.005	0.003	-0.046	0.002	-0.020
DayOfWeek	0.005	1.000	0.003	-0.009	0.003	-0.005
Period	0.003	0.003	1.000	0.155	0.003	0.115
NbrAppliances	-0.046	-0.009	0.155	1.000	-0.007	0.389
Hour	0.002	0.003	0.003	-0.007	1.000	0.020
KWH	-0.020	-0.005	0.115	0.389	0.020	1.000

For data set 2, the obtained covariance matrix for residential area 3 displayed in Table 3.4 shows that the correlation between consumption (KWH) and the number of appliances is the highest. The covariance between the consumption and the other features is in absolute value less than the covariance with the number of appliances. The covariance between the consumption and the day order or the day of the week has negative values, which means that they act oppositely.

Table 3.5: Covariance matrix on data set 2, residential area 4

Cov.	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
DayOrder	1.000	0.470	0.494	0.512	0.475	0.215
DayOfWeek	0.470	1.000	0.540	0.578	0.519	0.276
Period	0.494	0.540	1.000	0.584	0.546	0.295
NbrAppliances	0.512	0.578	0.584	1.000	0.575	0.486
Hour	0.475	0.519	0.546	0.575	1.000	0.274
KWH	0.215	0.276	0.295	0.486	0.274	1.000

For data set 2, the obtained covariance matrix for residential area 4 displayed in Table 3.5 shows that the correlation between consumption (KWH) and the number

of appliances is the highest. The covariance between the consumption and all other features is less than the covariance with the number of appliances. All the features have a similar positive effect on consumption, but their impact is less than the impact of the number of appliances.

Table 3.6: Covariance matrix on data set 2, residential area 5

Cov.	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
DayOrder	1.000	0.393	0.415	0.401	0.396	0.226
DayOfWeek	0.393	1.000	0.470	0.480	0.449	0.288
Period	0.415	0.470	1.000	0.546	0.476	0.458
NbrAppliances	0.401	0.480	0.546	1.000	0.479	0.632
Hour	0.396	0.449	0.476	0.479	1.000	0.285
KWH	0.226	0.288	0.458	0.632	0.285	1.000

For data set 2, the obtained covariance matrix for residential area 5 displayed in Table 3.6 shows that the correlation between consumption (KWH) and the number of appliances is the highest. The covariance between the consumption and all other features is less than the covariance with the number of appliances. All the features have a similar positive effect on consumption, but their impact is less than the impact of the number of appliances.

Table 3.7: Covariance matrix on data set 2, residential area 6

Cov.	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
DayOrder	1.000	0.453	0.475	0.489	0.456	0.197
DayOfWeek	0.453	1.000	0.538	0.594	0.517	0.243
Period	0.475	0.538	1.000	0.632	0.544	0.286
NbrAppliances	0.489	0.594	0.632	1.000	0.610	0.467
Hour	0.456	0.517	0.544	0.610	1.000	0.252
KWH	0.197	0.243	0.286	0.467	0.252	1.000

For data set 2, the obtained covariance matrix for residential area 6 displayed in Table 3.7 shows that the correlation between consumption (KWH) and the number of appliances is the highest. The covariance between the consumption and all other features is less than the covariance with the number of appliances. All the features

have a similar positive effect on consumption, but their impact is less than the impact of the number of appliances.

3.3.2 Singular Value Decomposition

Singular value decomposition (SVD) is a generalization of eigen decomposition, which is a matrix factorization approach. It consists in decomposing a square matrix into the product of its eigenvalues and eigenvectors, as in the equation:

$$Ax = \lambda x.$$

But in our case, the matrix A is not square. However, the matrices $A^t A$ and AA^t are square. The eigenvalue-eigenvector equation for the square matrix $(A^t A)$ can be written:

$$(A^t A - \lambda I)x = 0, \text{ with } x \neq 0.$$

Since each λ is a solution to $\det(A^t A - \lambda I) = 0$, the matrix $A^t A - \lambda I$ is singular. Let Λ be a diagonal matrix containing the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ of a matrix $A^t A$ on its diagonal. Let also $x_1, x_2, \dots, x_n$ be the set of corresponding eigenvectors. After normalizing the eigenvectors, we get the unit norms vectors $u_1, u_2, \dots, u_n$ that define the columns of a square matrix U such that $A^t A u_i = \lambda_i u_i, \forall i = 1, \dots, n$. In matrix form, it is written as:

$$A^t A U = U \Lambda.$$

Because the unit norm vectors $u_1, u_2, \dots, u_n$ are orthogonal and linearly indepen-

dent, U^{-1} exists and $A^t A U U^{-1} = U \Lambda U^{-1}$ yields:

$$A^t A = U \Lambda U^{-1}.$$

By applying the same methodology to AA^t , we get $AA^t = V \Lambda V^{-1}$, where V is a square matrix with orthogonal columns $v_1, v_2, \dots, v_n$ which are normalized eigenvectors of AA^t .

To comprehend the information in a data matrix A , principal component analysis (PCA) employs the largest eigenvalues connected to $A^t A$ and AA^t (common between them). These eigenvalues are called singular values, and their corresponding eigenvectors are called singular vectors. These eigenvectors are naturally orthonormal and define the matrices U and V . The orthonormal columns of U and V are the singular vectors of A and also its principal components [21].

Any two orthogonal eigenvectors v_i, v_j , such that $i \neq j$ satisfy:

$$(Av_i)^t(Av_j) = v_i^t(A^t A)v_j = v_i^t(\lambda_j v_j) = \lambda_j(v_i^t v_j) = 0.$$

Hence, $Av_1, \dots, Av_r$ form an orthogonal basis of R^n . If we normalize them, we get the left singular vectors:

$$u_i = \frac{Av_i}{\|Av_i\|} = \frac{1}{\sqrt{\lambda_i}} Av_i = \frac{1}{\sigma_i} Av_i,$$

where $\sigma_i = \sqrt{\lambda_i}$ and the eigenvalues of $A^t A$ are $\lambda_i = \sigma_i^2$.

For the singular vectors of a matrix $A \in R^{m \times n}$ with rank r , where $r \leq n$, each

$$Av = \sigma u: Av_1 = \sigma_1 u_1; Av_2 = \sigma_2 u_2; \dots; Av_r = \sigma_r u_r; Av_{r+1} = 0; \dots; Av_n = 0$$

If $A = U\Sigma V^t$, then

- $A^t A = (V\Sigma^t U^t)(U\Sigma V^t) = V\Sigma^t \Sigma V^t$.
- $AA^t = (U\Sigma V^t)(V\Sigma^t U^t) = U\Sigma \Sigma^t U^t$.

The matrices $A^t A$ and AA^t are different. While $\Sigma\Sigma^t$ or $\Sigma^t\Sigma$ contain the eigenvalues, $\Sigma^t\Sigma = \Sigma^2$ because Σ is a diagonal matrix ($\Sigma = \Sigma^t$).

Example 3.3.1 *This example shows how to find the matrices U , Σ , V for*

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

We have $A^t A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ and $AA^t = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$. These two matrices have the common eigenvalues $\lambda_1 = \sigma_1^2 = 1$ and $\lambda_2 = \sigma_2^2 = 3$. The remaining eigenvalue of $A^t A$ is $\lambda_3 = 0$.

$$\text{The eigenvectors of } A^t A \text{ are: } x_1 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, x_2 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \text{ and } x_3 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$$

$$\text{Therefore, the right singular vectors are } v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, v_2 = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \text{ and } v_3 =$$

$$\frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}.$$

The left singular vectors of A are obtained from $u_i = \frac{Av_i}{\sigma_i}$. Hence $u_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$$\text{and } u_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

$$\text{Therefore } U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{3} & 0 \end{pmatrix} \text{ and}$$

$$V^t = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ 2/\sqrt{6} & 1/\sqrt{6} & 1/\sqrt{6} \\ 1/\sqrt{3} & -1/\sqrt{3} & -1/\sqrt{3} \end{pmatrix}.$$

The matrix $A = U\Sigma V^t$ splits into two rank-one matrices, in columns times rows form, with: $A = \sigma_1 u_1 v_1^t + \sigma_2 u_2 v_2^t = \frac{1}{2} \begin{bmatrix} 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 2 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}.$

The matrices U and V contain orthonormal bases for the column space and the row space of A . The real achievement is that those two bases diagonalize A : $AV = U\Sigma$.

3.3.3 Principal Component Analysis

Reducing the dimension of the observation space in which certain objects are examined is the primary goal of principal component analysis (PCA). We can reduce

the dimension by developing new linear combinations of variables that describe the items under study. The SVD became the focal point of data science because PCA is "unsupervised" learning. PCA approximates a general matrix by a sum of rank one components. PCA uses SVD with the unique characteristic that those components are ordered by significance from largest to least. The best rank k approximation to A is $A_k = \sigma_1 u_1 v_1^t + \sigma_2 u_2 v_2^t + \dots + \sigma_k u_k v_k^t = U_k \Sigma_k V_k^t$.

The first component, $\sigma_1 u_1 v_1^t$, is the rank one matrix that is closest to A . Furthermore, the greatest sum for rank k is the sum of the first k components, where $(k < r)$.

PCA with SVD is frequently applied to the covariance matrix of standardized data to get the impact of every feature on the total variance. The correlation matrix of non-standardized data and the covariance matrix of standardized data are identical. We standardize (scale) the data because singular values are extremely sensitive to the relative ranges of original features. There are many advantages of using PCA, such as reducing the neural network training time, simple displaying of the data set with two or three main components, and improving the performance.

The steps below illustrate the PCA process using SVD. After each step, we display the result obtained by applying PCA on data set 2, residential area 2:

- Step(1): Obtain the data.
- Step(2): Standardize data. For each value of each variable, this can be achieved by subtracting the mean and dividing by the standard deviation.
- Step(3): Calculate the covariance matrix of standardized data.

$$Cov - matrix = \begin{bmatrix} 1.000 & 0.485 & 0.512 & 0.599 & 0.487 & 0.316 \\ 0.485 & 1.000 & 0.546 & 0.597 & 0.524 & 0.327 \\ 0.512 & 0.546 & 1.000 & 0.635 & 0.552 & 0.330 \\ 0.599 & 0.597 & 0.635 & 1.000 & 0.591 & 0.544 \\ 0.487 & 0.524 & 0.552 & 0.591 & 1.000 & 0.284 \\ 0.316 & 0.327 & 0.330 & 0.544 & 0.284 & 1.000 \end{bmatrix}$$

- Step(4): Find eigenvalues and eigenvectors:

Eigenvalues :

3.484 , 0.799 , 0.274 , 0.529 , 0.440 , 0.474

Eigenvectors :

$$\begin{bmatrix} -0.402 \\ -0.413 \\ -0.426 \\ -0.471 \\ -0.409 \\ -0.312 \end{bmatrix}, \begin{bmatrix} -0.154 \\ -0.186 \\ -0.197 \\ 0.157 \\ -0.311 \\ 0.884 \end{bmatrix}, \begin{bmatrix} 0.206 \\ 0.125 \\ 0.249 \\ -0.865 \\ 0.160 \\ 0.327 \end{bmatrix}, \begin{bmatrix} 0.865 \\ -0.361 \\ -0.155 \\ 0.027 \\ -0.301 \\ -0.071 \end{bmatrix}, \begin{bmatrix} -0.133 \\ -0.261 \\ 0.825 \\ 0.053 \\ -0.474 \\ -0.070 \end{bmatrix}, \begin{bmatrix} -0.075 \\ -0.762 \\ 0.110 \\ 0.044 \\ 0.629 \\ 0.065 \end{bmatrix}$$

- Step(5): Use SVD to obtain the covariance matrix's singular values and singular vectors.

$$U = \begin{bmatrix} -0.402 & 0.153 & -0.865 & 0.075 & 0.133 & -0.206 \\ -0.413 & 0.186 & 0.361 & 0.762 & 0.261 & -0.125 \\ -0.426 & 0.197 & 0.155 & -0.110 & -0.825 & -0.249 \\ -0.471 & -0.157 & -0.027 & -0.044 & -0.053 & 0.865 \\ -0.409 & 0.311 & 0.301 & -0.629 & 0.474 & -0.160 \\ -0.312 & -0.884 & 0.071 & -0.065 & 0.070 & -0.327 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 3.484 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.799 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.529 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.474 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.440 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.274 \end{bmatrix}$$

$$V^t = \begin{bmatrix} -0.402 & -0.413 & -0.426 & -0.471 & -0.409 & -0.312 \\ 0.153 & 0.186 & 0.197 & -0.157 & 0.311 & -0.884 \\ -0.865 & 0.361 & 0.155 & -0.027 & 0.301 & 0.071 \\ 0.075 & 0.762 & -0.110 & -0.044 & -0.629 & -0.065 \\ -0.133 & 0.261 & -0.825 & -0.053 & 0.474 & 0.070 \\ -0.206 & -0.125 & -0.249 & 0.865 & -0.160 & -0.327 \end{bmatrix}$$

- Step(6): Convert singular values to the variance explained. Where each singular value represents the amount of variance given by each component. The overall amount of variation in the data is equal to the sum of these singular values. Singular values must be converted to the variance explained to calculate these items.

$$\begin{bmatrix} 58.060 & 13.316 & 8.814 & 7.897 & 7.339 & 4.574 \end{bmatrix}$$

This means that the first component captures 58.1% of the variance in the data.

While the second component captures 13.3% of the variance in the data.

- Step(7): Choose the ideal amount of components by visualizing singular values. Not each component of the model contributes equally. We can eliminate the components that fail to represent the data's variation and retain only the most crucial ones.

In the figure 3.1 below, it is clear that the first 2 components alone capture about 71.4% of the variance in the data.

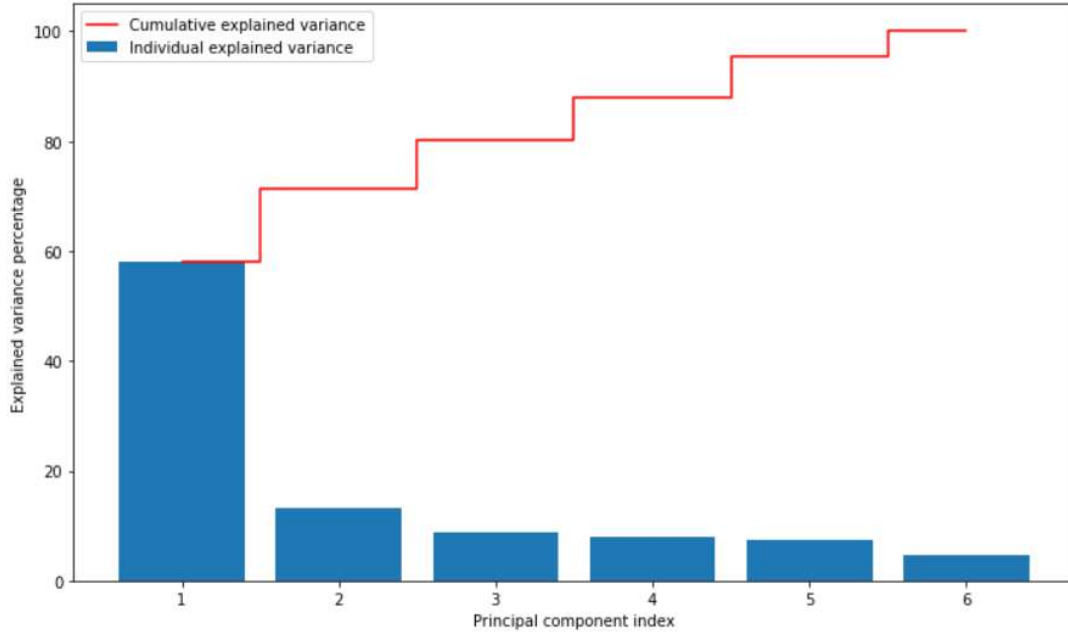


Figure 3.1: Explained variance plot data set 2

For data set (1), after cleaning and filling in some missing data, we obtain the data and standardize it. The covariance matrix of the standardized data is:

$$Cov-matrix = \begin{bmatrix} 1.000 & -0.998 & 0.159 & 0.201 & 0.277 & 0.220 & 0.243 & 0.123 \\ -0.998 & 1.000 & -0.159 & -0.200 & -0.276 & -0.218 & -0.243 & -0.124 \\ 0.159 & -0.159 & 1.000 & 0.003 & 0.016 & -0.004 & -0.002 & 0.899 \\ 0.201 & -0.200 & 0.003 & 1.000 & 0.943 & 0.068 & 0.024 & -0.053 \\ 0.277 & -0.276 & 0.016 & 0.943 & 1.000 & 0.075 & 0.031 & -0.053 \\ 0.220 & -0.218 & -0.004 & 0.068 & 0.075 & 1.000 & 0.003 & -0.008 \\ 0.243 & -0.243 & -0.002 & 0.024 & 0.031 & 0.003 & 1.000 & 0.015 \\ 0.123 & -0.124 & 0.899 & -0.053 & -0.053 & -0.008 & 0.015 & 1.000 \end{bmatrix}.$$

The eigenvalues are: 2.611 , 1.952 , 1.490 , 0.803 , 0.995 , 0.003 , 0.054 and 0.100.

The eigenvectors are:

$$\begin{bmatrix} -0.528 \\ 0.528 \\ -0.203 \\ -0.377 \\ -0.410 \\ -0.178 \\ -0.175 \\ -0.169 \end{bmatrix}, \begin{bmatrix} 0.055 \\ -0.056 \\ 0.596 \\ -0.360 \\ -0.352 \\ -0.036 \\ 0.016 \\ 0.619 \end{bmatrix}, \begin{bmatrix} 0.363 \\ -0.363 \\ -0.313 \\ -0.478 \\ -0.440 \\ 0.201 \\ 0.304 \\ -0.288 \end{bmatrix}, \begin{bmatrix} 0.290 \\ -0.292 \\ -0.062 \\ -0.096 \\ -0.049 \\ -0.587 \\ -0.680 \\ -0.092 \end{bmatrix}, \begin{bmatrix} -0.009 \\ 0.008 \\ -0.010 \\ 0.044 \\ 0.042 \\ -0.763 \\ 0.644 \\ -0.000 \end{bmatrix}, \begin{bmatrix} 0.707 \\ 0.707 \\ -0.005 \\ 0.000 \\ -0.001 \\ -0.001 \\ -0.000 \\ 0.005 \end{bmatrix}, \begin{bmatrix} -0.031 \\ 0.031 \\ -0.072 \\ -0.692 \\ 0.712 \\ 0.008 \\ 0.009 \\ 0.078 \end{bmatrix}, \begin{bmatrix} 0.010 \\ -0.020 \\ -0.704 \\ 0.095 \\ -0.054 \\ -0.007 \\ -0.022 \\ 0.701 \end{bmatrix}$$

Which define the singular values and singular vectors matrices:

$$U = \begin{bmatrix} -0.528 & 0.055 & 0.363 & -0.009 & -0.290 & -0.010 & 0.031 & -0.707 \\ 0.528 & -0.056 & -0.363 & 0.008 & 0.292 & 0.020 & -0.031 & -0.707 \\ -0.203 & 0.596 & -0.313 & -0.010 & 0.062 & 0.704 & 0.072 & 0.005 \\ -0.377 & -0.360 & -0.478 & 0.044 & 0.096 & -0.095 & 0.692 & -0.000 \\ -0.410 & -0.352 & -0.440 & 0.042 & 0.049 & 0.054 & -0.712 & 0.001 \\ -0.178 & -0.036 & 0.201 & -0.763 & 0.587 & 0.007 & -0.008 & 0.001 \\ -0.175 & 0.016 & 0.304 & 0.644 & 0.680 & 0.022 & -0.009 & 0.000 \\ -0.169 & 0.619 & -0.288 & -0.000 & 0.092 & -0.701 & -0.078 & -0.005 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 2.611 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.952 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.490 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.995 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.803 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.054 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.003 \end{bmatrix}$$

$$V^t = \begin{bmatrix} -0.528 & 0.528 & -0.203 & -0.377 & -0.410 & -0.178 & -0.175 & -0.169 \\ 0.055 & -0.056 & 0.596 & -0.360 & -0.352 & -0.036 & 0.016 & 0.619 \\ 0.363 & -0.363 & -0.313 & -0.478 & -0.440 & 0.201 & 0.304 & -0.288 \\ -0.009 & 0.008 & -0.010 & 0.044 & 0.042 & -0.763 & 0.644 & -0.000 \\ -0.290 & 0.292 & 0.062 & 0.096 & 0.049 & 0.587 & 0.680 & 0.092 \\ -0.010 & 0.020 & 0.704 & -0.095 & 0.054 & 0.007 & 0.022 & -0.701 \\ 0.031 & -0.031 & 0.072 & 0.692 & -0.712 & -0.008 & -0.009 & -0.078 \\ -0.707 & -0.707 & 0.005 & -0.000 & 0.001 & 0.001 & 0.000 & -0.005 \end{bmatrix}.$$

We then convert the singular values to the variance explained and get:

$$\begin{bmatrix} 32.606 & 24.374 & 18.610 & 12.424 & 10.024 & 1.246 & 0.677 & 0.039 \end{bmatrix}.$$

This means that the first component captures 32.6% of the variance in the data. While

the second component captures 24.4% of the variance. The third component captures 18.6% of the variance.

In the figure 3.2 below, it is clear that the first 3 components capture about 75.6% of the variance in the data.

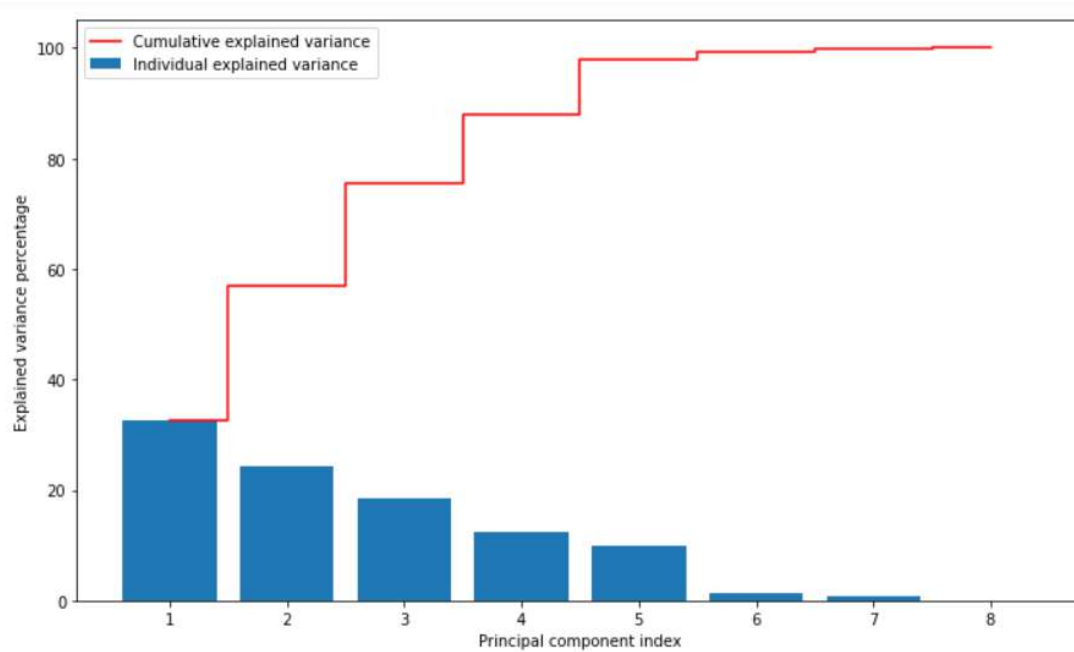


Figure 3.2: Explained variance plot data set 1

3.4 Trend and Seasonality

Seasonality is a periodical and recurring pattern produced by elements including the climate, holidays, recurring sales, and the actions of economic agents. Seasonal changes are likely the most important factor in a seasonal time series. A trend is a pattern in data that demonstrates how a series moves over time to relatively greater or lower values. In other words, it can be seen when the time series has a growing

or declining slope. Typically, it lasts for a while before disappearing and it does not repeat. Which can be discovered by either simple visualization or data set decomposition. The stochastic trend is frequently present and can significantly affect different forecasting techniques [22]. Stationary time series satisfy that, the mean, and variance do not change over time and the trend and seasonality do not exist. A time series with a trend is regarded as non-stationary and frequently must be converted to stationary before the modeling and forecasting operations can take place. Seasonal time series are typically modeled to use certain seasonal adjustment methods to remove seasonal changes, and then the models are re-scaled using the predicted seasonal variations for forecasting purposes. If there is a seasonality, the time series is non-stationary and we can not use some models like ARIMA which are based on the assumption that the data is stationary. One of the methods used to make a time series stationary is differencing. This method involves removing the trend component from the data by subtracting consecutive observations from one another. Another method is the seasonal decomposition of time series, which separates the time series into trend, seasonal, and residual components.

Trend and seasonality decomposition, whether additive or multiplicative, allows the detection of inner data phenomena specific to the problem. An additive decomposition is used if the variation around the trend cycle or the size of seasonal variations does not change with the time series level. Multiplicative decomposition is preferable when such variance is shown to be proportional to the time series level. The additive

formula of time series decomposition is:

$$y_t = T_t + S_t + R_t.$$

While the multiplicative formula is:

$$y_t = T_t \times S_t \times R_t,$$

where y_t means for the observed series, T_t for trend component, S_t for seasonal component, and R_t for irregular component (residual), all of them at period t [23]. For instance, the additive seasonality decomposition in Figure 3.3 shows the non-existence of a major seasonal component for electricity consumption for data set 1. It shows the existence of some particular trend related to the increase in electricity consumption during the first season corresponding to the hard winter in Canada. The Dickey-Fuller test, with p-value= 0.0001 less than 0.05, on the electricity consumption, confirms the absence of any major seasonality.

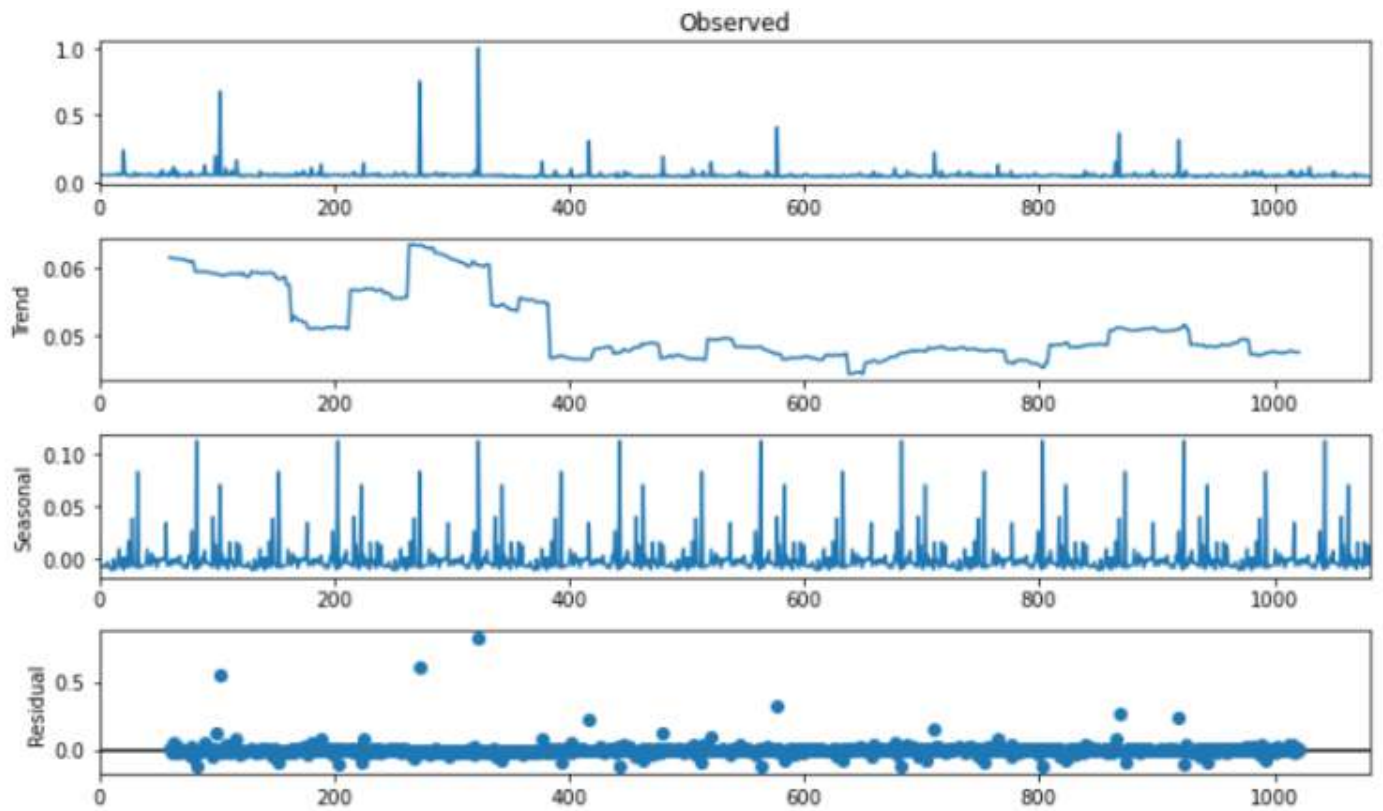


Figure 3.3: Additive Trend-Seasonality Decomposition Results

3.5 K-Means Clustering

Data can be separated into groups or clusters, using a variety of processes called clustering. Clusters are collections of data objects that share more similarities between

them than with those in other clusters. Because there are so many options, choosing the right clustering algorithm for the data set can be challenging. Some significant aspects influence this choice, like the properties of the clusters, the data set's characteristics, the number of outliers, the features, and the number of data objects. K-means clustering is an unsupervised machine-learning technique used in data mining to identify collections of data objects. Because this approach is centroid-based, each cluster has a centroid given to it [24]. The aim of using K -means clustering algorithm is to:

- Reduce the total distance between each data point and its corresponding centroid.
- Select the ideal value for K centroids using an iterative method.
- Match each data point with the closest K center. The data points that are near a certain K center group together to form a cluster.

The following stages illustrate how the K-means algorithm performs, where each cluster $N_j \in N_1, N_2, \dots, N_k$ has a centroid C_j $C_1, C_2, \dots, C_k = C_j$ obtained through norm l_2 minimization between each data point $i \in N_j$ and its corresponding centroid:

- Step 1: Select K to get the number of clusters.
- Step 2: Randomly select K centroids.
- Step 3: The K clusters will be generated by assigning each data point to its nearest centroid, resulting in the desired number of clusters. We should minimize the quadratic sum of cluster inaccuracies M for all cluster points and their mean

values:

$$M = \sum_{j=1}^k \sum_{i \in N_j} ||a_i - C_j||^2.$$

- Step 4: Find the mean and move the centroid of each cluster. The centroid value is the calculated mean value for each cluster:

$$C_j = \frac{1}{S} \sum_{i \in N_j} a_i,$$

where S is the number of the points in each cluster.

- Step 5: Repeat the third step to reassign each data point to the updated centroid of each cluster.
- Step 6: Move to step 4 if there is a reassignment; otherwise, finalize the model.
- Step 7: Finalize the model.

The following are the steps that must be taken for implementation (in the code):

- (1) Process the data by importing libraries and data sets.
- (2) Choose the appropriate number of clusters using the elbow method, which is one of the most common methods for figuring out the optimal number of clusters. The elbow method uses the total variations inside a cluster WCSS (Within Cluster Sum of Squares). The elbow approach goes through the procedure listed below to determine the best value of clusters:
 - a) Apply K -means clustering on a given data set for various K values (ranges of the number of clusters is between two and six).

- b) Calculate the WCSS value for each value of K .
- c) Sketches a curve from the estimated WCSS values to the K -cluster value.
- d) The sharp point of a bend, such as an elbow, is considered to have the K value.
- (3) Use the training data set to run the K -means algorithm and making the clusters clear.
- (4) Visualize the clusters: while our model consists of 3 clusters, we will visualize each cluster separately.

The figures below display the elbow curve around the critical K-value(s) and the clusters' centroids when $k = 3$ and $k = 4$ on the Toronto data set 1. In Figure 3.4 the x -axis describes the range of the number of clusters we choose which is between 2 and 6, while the y -axis shows the sum of squared error. We determine the number of clusters (K) according to the sharp point in the graph as shown in the figure we have 2 sharp points at 3 and 4. The x -axis shows the number of appliances which is the most effective feature for consumption. The y -axis shows the consumption values. In Figure 3.5 we have 3 groups in different colors. In Figure 3.6 we have 4 groups.

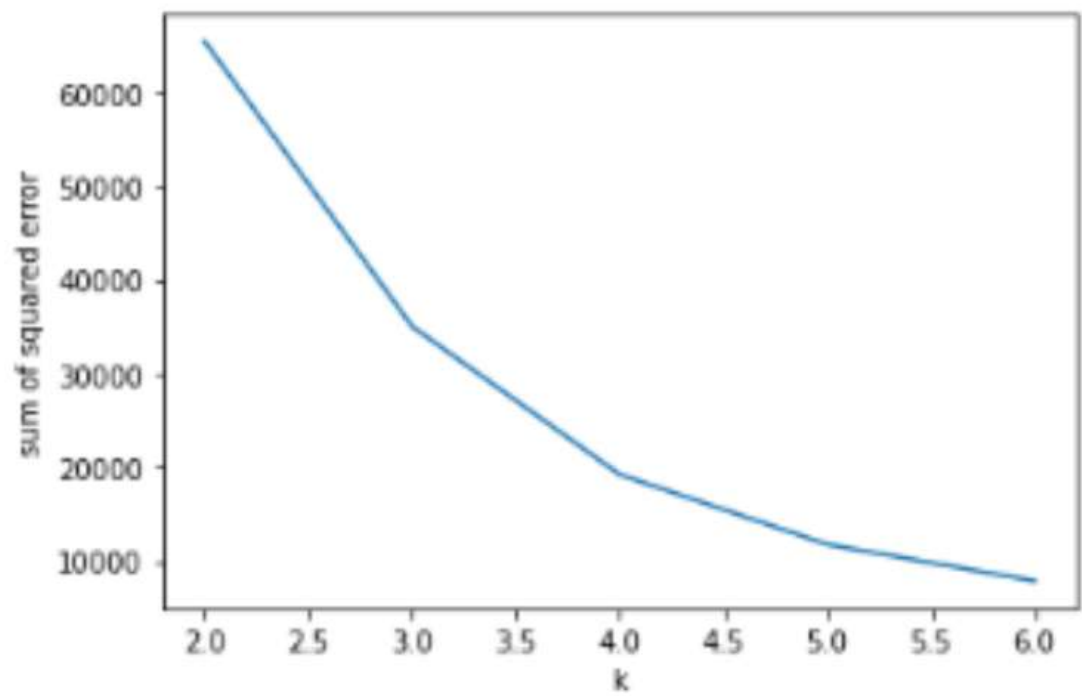


Figure 3.4: Elbow curve data set 1

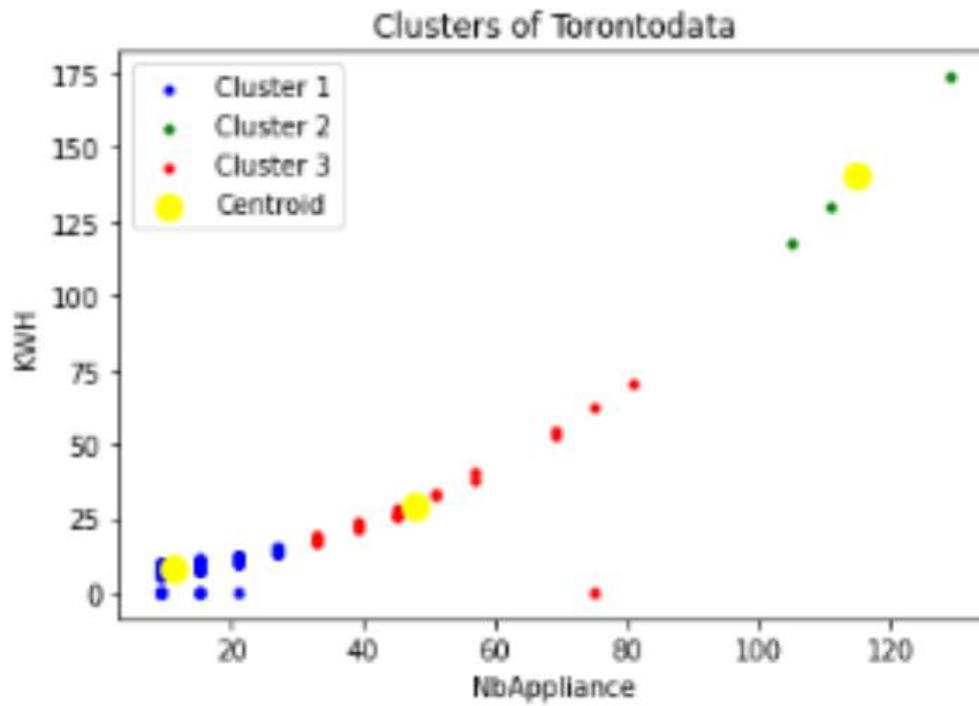


Figure 3.5: 3 clusters on data set 1

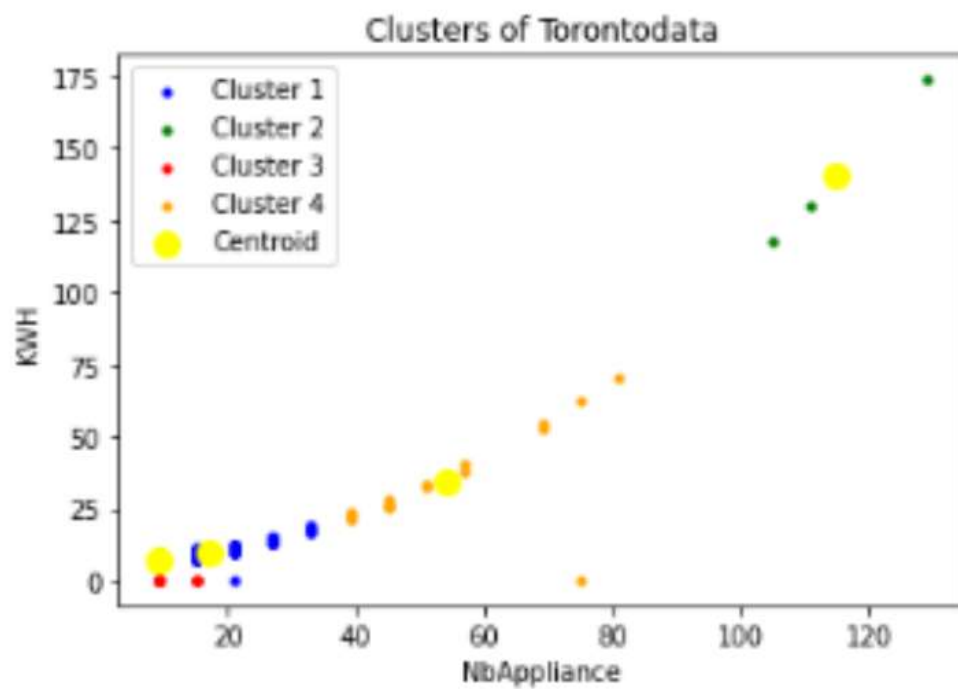


Figure 3.6: 4 clusters on data set 1

CHAPTER 4

ARTIFICIAL NEURAL NETWORK FORECASTING APPROACH

This chapter provides a detailed account of the statistical and mathematical models employed in our Artificial Neural Network (ANN) approach and outlines the use of the gradient descent method for weight optimization. Our Neural Network (NN) approach for forecasting energy demands is tested using three statistical models: a time series model, an auto-regressive model, and a hybrid model. To do so, it applies different network architectures in terms of the number of neurons at each layer, activation function types, and gradient descent methods, i.e., Adam, Adagrad, or Adamax.

4.1 Statistical Models and Network Architecture

This thesis studies the performance of three different models on different neural network architectures tested on benchmark and two real-life instances: spatial and temporal data sets. First of all, this section presents the statistical models used: times series, auto-regressive and hybrid models. Secondly, it presents the different network architectures tested. Finally, it details the gradient descent methods used.

4.1.1 Statistical Models

For our proposed artificial neural network implementation, and each household i , the input variables are denoted by p_i^t and the output variables correspond to the load demand y_i^t in kilowatt per hour (KWH).

(A) Time Series Model

Time series models are statistical tools that enable the examination and prediction of data collected over a period. These models are applicable in various fields, including finance, economics, weather forecasting, and more. The primary objective of time series modeling is to comprehend and delineate the underlying patterns and trends in the data and provide precise predictions of future values. In this thesis, the time series model is tested on both data sets 1 and 2. For the first layer, we apply the activation function on the data set inputs. The output obtained from the first layer neurons is then as used an input for the second layer. This continues till we reach the last layer L . It considers:

$$\hat{y}^t = F(p^t) = f_L(\dots f_l(\dots f_1(p^t))).$$

Toronto data set 1 contains 1082 households energy consumption information collected daily over 36 consecutive months, during the past 3 years: 2019, 2020, 2021. The collected data indicates the geographic coordinates of the household and its current index in the data set. Hence, for data set 1 the t index is a spatial index rather than a time index.

Data set 2 contains over 30000 observations for 6 selected residential areas. Energy consumption information was collected daily over 60 consecutive months, during the past 5 years: 2017, 2018, 2019, 2020, and 2021. Therefore, for data Set 2 the t index is a genuine time index. Figure 4.1 displays a time series input-output data sample from residential area 1.

(B) Auto-Regressive Model

Auto-regressive models (AR) are a category of time series models that leverage previous values of a variable to forecast its future values. These models assume that the present value of the variable is a product of its past values and a random error term. The order of an AR model, marked as ' r ', represents the number of preceding values utilized to predict the present value. For instance, an AR(1) model utilizes the past value of the variable to predict the current value, while an AR(2) model considers the last two values. In this thesis, our auto-regressive model considers:

$$\hat{y}^t = F(y^{t-r}) = f_L(\dots f_l(\dots f_1(y^{t-r}))),$$

where r represents the number of preceding values of y used, as well as the time delay in hours needed between the last measurement and the output forecasting.

	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH
0	345	4	4	0	1	0.000
1	345	4	4	0	2	0.002
2	345	4	4	2	3	0.088
3	345	4	4	2	4	0.111
4	345	4	4	2	5	0.109
...	...	...	...	...	...	...
33207	268	2	2	3	4	2.501
33208	268	2	2	3	5	3.870
33209	268	2	2	3	6	4.125
33210	268	2	3	3	1	5.178
33211	268	2	3	3	2	3.455

33212 rows $\times$ 6 columns

Figure 4.1: Inputs for Time Series Model

For data set 1, one single observation is measured for every household, therefore this model cannot be applied. Therefore, the auto-regressive model is tested only on data set 2, with a maximum delay of $r = 6$ hours corresponding to a full period. Figure 4.2 displays an auto-regressive data sample from residential area 1.

(C) Hybrid Model

Hybrid models, also called mixed models, refer to models that blend multiple types of models to make predictions. A hybrid time series model can integrate diverse models, such as combining an auto-regressive model with a moving average model or

	KWH	KWH-1	KWH-2	KWH-3	KWH-4	KWH-5	KWH-6
0	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.002	0.000	0.000	0.000	0.000	0.000	0.000
2	0.088	0.002	0.000	0.000	0.000	0.000	0.000
3	0.111	0.088	0.002	0.000	0.000	0.000	0.000
4	0.109	0.111	0.088	0.002	0.000	0.000	0.000
...	...	...	...	...	...	...	...
33207	2.501	1.029	0.091	0.008	0.007	0.013	0.008
33208	3.870	2.501	1.029	0.091	0.008	0.007	0.013
33209	4.125	3.870	2.501	1.029	0.091	0.008	0.007
33210	5.178	4.125	3.870	2.501	1.029	0.091	0.008
33211	3.455	5.178	4.125	3.870	2.501	1.029	0.091
33212 rows × 7 columns							

Figure 4.2: Inputs for Auto-Regressive Model

an auto-regressive integrated moving average (ARIMA) model with a neural network model. The objective of a hybrid model is to capitalize on the strengths of different models and compensate for their limitations. For instance, an auto-regressive model may work proficiently for short-term predictions, but not for long-term predictions. In such a scenario, a hybrid model that merges an auto-regressive model with a moving average model could result in better outcomes. In this thesis, our hybrid model combines the time series and the auto-regressive models as it considers:

$$y^t = F(p^t, y^{t-r}) = f_L(\dots f_l(\dots f_1(p^t, y^{t-r}))).$$

Similarly to the auto-regressive model, the hybrid model is tested only on data set

2, with a maximum delay of $r = 6$ hours corresponding to a full period. Figure 4.3 displays an auto-regressive data sample from residential area 1.

	DayOrder	DayOfWeek	Period	NbrAppliances	Hour	KWH	KWH-1	KWH-2	KWH-3	KWH-4	KWH-5	KWH-6
0	345	4	4	0	1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	345	4	4	0	2	0.002	0.000	0.000	0.000	0.000	0.000	0.000
2	345	4	4	2	3	0.088	0.002	0.000	0.000	0.000	0.000	0.000
3	345	4	4	2	4	0.111	0.088	0.002	0.000	0.000	0.000	0.000
4	345	4	4	2	5	0.109	0.111	0.088	0.002	0.000	0.000	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...
33207	268	2	2	3	4	2.501	1.029	0.091	0.008	0.007	0.013	0.008
33208	268	2	2	3	5	3.870	2.501	1.029	0.091	0.008	0.007	0.013
33209	268	2	2	3	6	4.125	3.870	2.501	1.029	0.091	0.008	0.007
33210	268	2	3	3	1	5.178	4.125	3.870	2.501	1.029	0.091	0.008
33211	268	2	3	3	2	3.455	5.178	4.125	3.870	2.501	1.029	0.091

33212 rows × 12 columns

Figure 4.3: Inputs for Hybrid Model

4.1.2 Network Architectures

Our current implementation of ANN for forecasting in the smart grid DRM problem has three layers of neurons. We tested the first layer (input layer) and the second layer (hidden layer) using four activation functions: the log-sigmoid function, the hyperbolic tangent function (tanh), the sigmoid linear unit (selu), and the rectified linear unit (relu) function:

- Sigmoid: $f_{1,2}(X) = \frac{1}{1+e^{-X}}$
- Tanh: $f_{1,2}(X) = \frac{e^X - e^{-X}}{e^X + e^{-X}}$
- Selu: $f_{1,2}(X) = \frac{X}{1+e^{-X}}$
- Relu: $f_{1,2}(X) = \max(0, X)$.

The last layer only used the relu activation function, because we want to get positive values for the consumption. The number of neurons S_1 varies from 32 to 256 for layer 1. The number of neurons S_2 varies from 16 to 128 for layer 2. Finally, the number of neurons S_3 varies from 8 to 64 for layer 3.

4.2 Gradient Descent Methods

Assuming p_r^t as the input(s) and y^t as the output(s) of DRM, with t as the observation index, suppose that the training data set contains T elements and that the total number of input variables is denoted by R .

In a given neuron, an activation (transfer) function f is used, as illustrated in Figure 4.4, where $p^t = (p_1^t, \dots, p_r^t, \dots, p_R^t, 1)$ and $f(p^t) = \hat{y}^t$ approximates the output measure y^t at the time or observation index t .

Activation functions commonly used in artificial neural networks, such as the logistic sigmoid function $f(x) = \frac{1}{1+e^{-x}}$, assume that $x^t = \sum_{r=1}^R w_r p_r^t + b$, where $w = (w_1, \dots, w_r, \dots, w_R, b)$ is the input weights and bias b . The neurons at each layer l ($l = 1, \dots, L$) of the ANN are then applied with their activation function f_l . The output prediction after a complete forward pass through the ANN is given by $\hat{y}^t = F(p^t) = f_L(\dots f_l(\dots f_1(p^t)))$, where F is the resulting composed function. Thus,

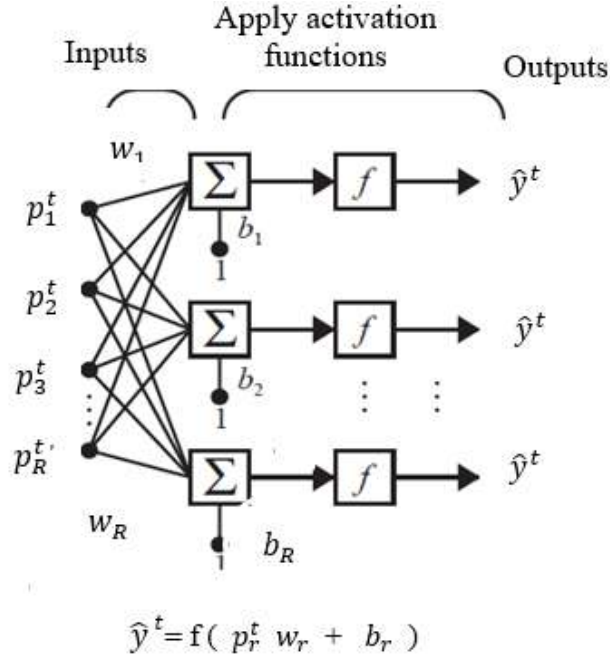


Figure 4.4: ANN First layer

the overall error function for estimating each output obtained after a complete forward propagation through the L layers of the artificial neural network is referred to as

$$E(T) = \sum_{t=1}^T (y^t - \hat{y}^t)^2 = \sum_{t=1}^T (y^t - F(p^t))^2.$$

The use of an optimizer allows us to maximize production efficiency or minimize error functions (loss functions). The optimizer is a learning algorithm that updates the parameters [w's (weights/coefficients), b (bias)] of the model to calculate an accurate target value $\hat{y}^t$ from the set of inputs p^t [25]. We want to reach the optimal weight parameters values which minimize the error. In a typical steepest descent approach, the weights are updated after each forward pass using the following sequence:

$$w_r^{h+1} = w_r^h - \mu \frac{1}{\|\nabla E(T)\|} \frac{\partial E(T)}{\partial w_r},$$

$$b^{h+1} = b^h - \mu \frac{1}{\|\nabla E(T)\|} \frac{\partial E(T)}{\partial b},$$

where μ is a pre-defined parameter adjusted according to specific rules, and $\|\nabla E(T)\|$ represents the magnitude of $\nabla E(T)$. In vector notation, the steepest gradient descent scheme can be expressed as:

$$w^{h+1} = w^h - \frac{\mu}{\|\nabla E(T)\|} \cdot \nabla E(T).$$

For simplification purposes, we abstract the input and output indices in the following mathematical derivation. Let T be the size of the training set, the estimation error function for each output can be expressed as

$$E(T) = \sum_{t=1}^T (y_t - \hat{y}_t)^2 = \sum_{t=1}^T (y_t - F)^2.$$

The partial derivative of $E(T)$ with respect to each input weight w_j can be given as:

$$\frac{\partial E(T)}{\partial w_j} = \sum_{t=1}^T -2[y_t - F] \frac{\partial F}{\partial w_j}.$$

Similarly, the partial derivative of $E(T)$ with respect to each bias b can be expressed as:

$$\frac{\partial E(T)}{\partial b} = \sum_{t=1}^T -2[y_t - F] \frac{\partial F}{\partial b}.$$

The gradient vector can be represented as:

$$\nabla E(T) = \left[\left(\frac{\partial E(T)}{\partial w_j} \right), \frac{\partial E(T)}{\partial b} \right]^t.$$

In the following chapter 4, we evaluate the performance of our ANN by employing various gradient optimizers, such as adaptive moment estimation (Adam), adaptive gradient algorithm (Adagrad) [26], and an extension of Adam known as Adamax [26]. Therefore, it is important to detail these algorithms. Stochastic Gradient Descent (SGD) [27] lays the foundation of some of these optimizers. Although it did not provide usable results in our current context, it is detailed in the next subsection for its importance.

4.2.1 SGD

In machine learning, stochastic gradient descent (SGD) is commonly used to minimize the loss function of a model. It is a variant of the gradient descent algorithm that is specifically designed to handle large data sets. While traditional gradient descent updates model parameters based on the average gradient of the entire data set, SGD updates parameters using the gradient of a randomly selected subset of the data (known as a "mini-batch"). This approach leads to faster convergence and enables the algorithm to handle much larger data sets than traditional gradient descent. The learning rate is denoted by η . The precision parameter is denoted by $\epsilon = 10^{-8}$, and the sample size is denoted by n_s . An approximation of the gradient ∇E_h for sample j at iteration h is denoted by ∇Q_j^h . The pseudo-code for SGD is detailed in Algorithm

1.

Algorithm 1: SGD

Start: Set $h \leftarrow 0$

Do While $\|w_{h+1} - w_h\| > \epsilon$, w_h did not converge yet:

- $h \leftarrow h + 1$ (update the iteration index)
- Randomly shuffle samples in training set
- Select $j \in \{1, 2, \dots, n_s\}$ (choose a random subset)
- $w_{h+1} \leftarrow w_h - \eta \nabla Q_j(w_h)$ (update the weights)

End Do While

End: return w_h .

SGD estimates the true gradient of the loss function using a small amount of data, then updates the model parameters in the direction of the negative gradient to minimize the loss function. It is computationally efficient and suitable for large-scale machine learning tasks but may not converge to the global minimum since it is not working well with sparse gradient [27].

4.2.2 Adam

Adaptive moment estimation (Adam) is among the most famous deep learning optimization techniques. It is a method for efficient stochastic optimization that just demands first-order gradients and takes less memory. It uses estimates of the initial and second moments of the gradients to calculate individual adaptive learning rates for a variety of parameters. Adam relies on updated running averages of both the gradients and the second moments of the gradients. The moments are used to recall

the updates at previous iterations.

Adam offers lots of benefits, such as the fact that the magnitudes of parameter updates are invariant to gradient scaling. It is easy to use and does not require a constant objective. It works with sparse gradients and automatically performs a type of step-size annealing. It also has excellent computing performance and little memory requirement [26], [27].

To describe Adam in detail, let us first define α as the step size, β_1 and $\beta_2 \in [0, 1)$ as the exponential decay rates for the moment estimates, and h as the iteration index. Let us also initialize the weight vector w_0 , the initial first-moment vector $m_0 = 0$, and the initial second-moment vector $v_0 = 0$. Default settings may consider $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Adam's algorithm can be stated as follows.

Algorithm 2: Adam

Start: Set $h \leftarrow 0$

Do While $\|w_{h+1} - w_h\| > \epsilon$, w_h did not converge yet:

- $h \leftarrow h + 1$ (update the iteration index)
- $\nabla E_h \leftarrow \nabla E(w_{h-1})$ (update the gradient)
- $m_h \leftarrow \beta_1 \cdot m_{h-1} + (1 - \beta_1) \cdot \nabla E_h$ (update biased first moment estimate)
- $v_h \leftarrow \beta_2 \cdot v_{h-1} + (1 - \beta_2) \cdot \nabla E_h^2$ (update biased second moment estimate)
- $\hat{m}_h \leftarrow \frac{m_h}{(1 - \beta_1^h)}$ (compute biased-corrected first-moment estimate)
- $\hat{v}_h \leftarrow \frac{v_h}{(1 - \beta_2^h)}$ (compute biased-corrected second-moment estimate)
- $w_h \leftarrow w_{h-1} - \alpha \cdot \hat{m}_h / (\sqrt{\hat{v}_h} + \epsilon)$ (update the weight parameters)

End Do While

End: return w_h .

4.2.3 Adagrad

Adaptive gradient algorithm (Adagrad) modifies the weight parameters learning rate. It uses all of the gradient's previously squared values from each iteration starting at the beginning of training. The Adagrad approach has disadvantages, such as when the sum of squares of the gradients increases, the learning rhythm slows down. The learning rate declines slowly for parameters with low partial derivatives and quickly for those with the highest partial derivative. The algorithm suddenly stops working before reaching the global optimum because the learning rate decreases significantly. In general, Adagrad works well for simple quadratic problems, but it typically ends quickly when training neural networks. Adagrad increases the learning rate for sparser parameter values and decreases it for those with lower sparsity. Indeed, it works well for sparse gradients. Its basic version updates parameters as:

$$w_{h+1} = w_h - \alpha \cdot \frac{\nabla E_h}{\sqrt{\sum_{i=1}^h \nabla E_i^2}}.$$

Note that if we choose β_2 to be very close to 1 from below, then:

$$\lim_{\beta_2 \rightarrow 1} \hat{v}_h = \frac{1}{h} \cdot \sum_{i=1}^h \nabla E_i^2.$$

Adagrad is a kind of a version of Adam with $\beta_1 = 0$, very small $(1 - \beta_2)$, and a replacement of α by an annealed version.

4.2.4 Adamax

The foundation of Adamax is the concept that an L^2 norm-based update rule can be expanded to an L^P norm-based update rule. A straightforward algorithm is obtained

in the specific case where $p \rightarrow \infty$. The Adamax algorithm starts by computing gradients concerning the stochastic objective at iteration h . Then, an exponentially-weighted infinity norm and a biased first-moment estimate are calculated. The weight parameters are then updated.

To describe Adamax, we first define α as the step size, β_1 and $\beta_2 \in [0, 1)$ as the exponential decay rates, β_1^h as β_1 to the power h , $(\frac{\alpha}{1-\beta_1^h})$ as the learning rate with the bias-correction term for the first moment. We initialize $m_0 = 0$ as the initial first moment vector, and $u_0 = 0$ as the initial exponentially weighted infinity norm. Common default settings may consider $\alpha = 0.002$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Adamax algorithm can be stated as follows.

Algorithm 3: Adamax

Start: Set $h \leftarrow 0$

Do While $\|w_{h+1} - w_h\| > \epsilon$, w_h did not converge yet:

- $h \leftarrow h + 1$ (update the iteration index)
- $\nabla E_h \leftarrow \nabla_w E_h(w_{h-1})$ (update the gradient)
- $m_h \leftarrow \beta_1 \cdot m_{h-1} + (1 - \beta_1) \cdot \nabla E_h$ (update biased first-moment estimate)
- $u_h \leftarrow \max(\beta_2 \cdot u_{h-1}, |\nabla E_h|)$ (update the exponentially-weighted infinity norm)
- $w_h \leftarrow w_{h-1} - (\frac{\alpha}{1-\beta_1^h}) \cdot \frac{m_h}{u_h}$ (update the weight parameters)

End Do While

End: return w_h .

CHAPTER 5

FORECASTING

EXPERIMENTAL RESULTS

We conducted our experiments using Python 3 on Jupyter Notebook. The computations were carried out on 64-bit workstations with 2.9 GHz Intel Core i7-7600 processors and 8 GB RAM running under MS Windows.

The accuracy results of the three-layer ANN are summarized in the following tables. The results were obtained by varying the number of neurons in each layer (S_1 , S_2 , and S_3) and the activation functions f_1 and f_2 . The entries in the table indicate the root mean squared error RMSE of the residuals calculated with a training set size that remains constant at 3/4 of the entire data set, while the household consumption is predicted. RMSE is one of the most popular measures for assessing the accuracy of predictions. It illustrates the Euclidean distance between measured true values and forecasts. To compute RMSE, we first calculate the residual (difference between prediction and true value) for each data point, then compute the norm of residual for

each data point. Finally, we compute the mean of residuals and take the square root of that mean using the formula:

$$RMSE = \sqrt{\frac{\sum_{t=1}^T (y^t - \hat{y}^t)^2}{N}}.$$

All the results are obtained with a fixed number of epochs (50). An epoch refers to a single forward and backward pass through the entire training data set.

5.1 Forecasting Results on Data Set 1

Table 5.1: Three-Layer ANN Forecasting Results on Data Set 1

			f_1 : log-sigmoid; f_2 : selu; f_3 : relu			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.
32	16	8	1.828	7.984	7.194	5.669
64	32	16	2.004	7.244	6.967	5.405
128	64	32	2.426	7.214	5.636	5.092
256	128	64	2.987	7.04	4.315	4.781
		avg.	2.311	7.371	6.028	5.237
			f_1 : selu; f_2 : tanh; f_3 : relu			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.
32	16	8	2.907	7.297	6.099	5.434
64	32	16	2.372	7.162	4.500	4.678
128	64	32	3.480	6.889	3.276	4.548
256	128	64	4.002	5.982	2.187	4.057
		avg.	3.190	6.833	4.016	4.679
			f_1 : tanh; f_2 : log-sigmoid; f_3 : relu			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.
32	16	8	3.060	7.217	7.149	5.809
64	32	16	3.628	7.200	7.160	5.996
128	64	32	1.793	7.199	7.031	5.341
256	128	64	2.177	7.182	6.917	5.425
		avg.	2.665	7.200	7.064	5.643
			f_1 : relu; f_2 :relu ; f_3 : relu			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.
32	16	8	1.817	7.560	6.501	5.293
64	32	16	1.201	7.375	5.830	4.802
128	64	32	2.275	7.125	4.313	4.571
256	128	64	1.307	6.465	2.736	3.503
		avg.	1.650	7.131	4.845	4.542

The results in table 5.1 show that the last architecture, with f_1 :relu and f_2 :relu as activation functions, provides, in general, the best prediction and the lowest RMSE

in averages. In particular, the lowest RMSE (1.201) is obtained with $S_1 = 64$ neurons in the first layer, $S_2 = 32$ neurons in the second, and $S_3 = 16$ neurons in the third layer with Adam optimizer. Adam is more stable than Adagrad and Adamax when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3 varies from 8 to 64 neurons on the four architectures. The third architecture, with f_1 :**tanh** and f_2 :**log-sigmoid** as activation functions, provides a larger RMSE on average. Its best RMSE (1.793) is slightly better than that obtained by the best accuracy provided by the first architecture. More detailed results are provided upon request.

Figure 5.1 illustrates a measurement vs. prediction values of consumption in the y -axis (blue vs. green) with the proposed three-layer ANN, with $f_1 : \text{relu}$, $f_2 : \text{relu}$, and Adam optimizer, applied on data set 1. Where the x -axis is used for the index of the observation.

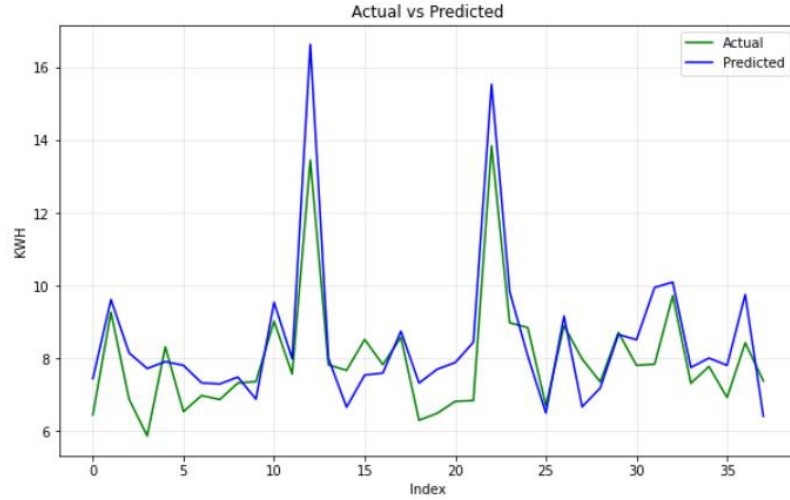


Figure 5.1: Sample Measurement vs. Prediction on Data Set 1

5.2 Forecasting Results on Data Set 2

Table 5.2: Forecasting Results on Residential 1

			Time-series				f_1 : sigmoid; f_2 :selu ; f_3 : relu				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.076	1.629	1.211	1.305	0.766	0.988	0.812	0.855	0.843	1.041	0.894	0.926
64	32	16	0.998	1.56	1.071	1.201	0.736	0.902	0.784	0.807	0.840	0.939	0.890	0.890
128	64	32	1.027	1.54	1.043	1.203	0.727	0.871	0.787	0.795	0.890	0.891	0.904	0.895
256	128	64	1.03	1.538	1.038	1.202	0.775	0.863	0.756	0.798	0.838	0.894	0.878	0.870
		Avg.	1.032	1.567	1.091	1.228	0.751	0.906	0.785	0.814	0.853	0.941	0.892	0.895
			time series				f_1 : tanh; f_2 :sigmoid ; f_3 : relu				Auto regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.002	1.568	1.031	1.2	0.745	0.994	0.787	0.842	0.848	1.029	0.850	0.909
64	32	16	1.007	1.484	1	1.164	0.710	0.907	0.731	0.783	0.855	0.911	0.882	0.883
128	64	32	1.005	1.432	0.993	1.143	0.706	0.870	0.727	0.768	0.824	0.901	0.840	0.855
256	128	64	0.981	1.555	0.997	1.178	0.809	0.851	0.760	0.807	0.862	0.900	0.895	0.886
		Avg.	0.999	1.510	1.005	1.171	0.743	0.906	0.751	0.800	0.847	0.935	0.867	0.883
			Time series				f_1 : tanh; f_2 :selu ; f_3 : relu				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.998	2.221	1.035	1.418	0.764	0.842	0.749	0.785	0.850	0.876	0.833	0.853
64	32	16	0.985	1.335	1.001	1.107	0.749	0.825	0.752	0.775	0.825	0.870	0.821	0.839
128	64	32	0.981	1.286	0.99	1.086	0.734	0.820	0.705	0.753	0.813	0.862	0.819	0.831
256	128	64	1.004	1.226	1.006	1.079	0.733	0.812	0.728	0.758	0.837	0.864	0.877	0.859
		Avg.	0.992	1.517	1.008	1.173	0.745	0.825	0.734	0.768	0.831	0.868	0.838	0.846
			Time series				f_1 : relu; f_2 :relu ; f_3 : relu				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.981	1.381	1.007	1.123	0.701	0.847	0.759	0.769	0.807	0.891	0.843	0.847
64	32	16	0.994	1.346	1.003	1.114	0.727	0.816	0.729	0.757	0.833	0.863	0.840	0.845
128	64	32	0.985	1.263	0.978	1.075	0.702	0.805	0.731	0.746	0.811	0.858	0.833	0.834
256	128	64	0.986	1.158	0.977	1.04	0.699	0.779	0.697	0.725	0.839	0.865	0.839	0.848
		Avg.	0.987	1.287	0.991	1.088	0.707	0.812	0.729	0.749	0.823	0.869	0.839	0.844

Table 5.2 displays the results for the residential area 1 from data set 2. For the time series model, the last architecture, with f_1 :**relu** and f_2 :**relu** as activation functions, provides the lowest RMSE (0.977), obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, $S_3 = 64$ neurons in the third layer and Adamax optimizer.

For the auto-regressive model, with f_1 :**relu** and f_2 :**relu** as activation functions, the lowest RMSE (0.807) is obtained with $S_1 = 32$ neurons in the first layer, $S_2 = 16$ neurons in the second and $S_3 = 8$ neurons in the third layer with Adam optimizer.

For the hybrid model, with f_1 :**relu** and f_2 :**relu** as activation functions, the lowest RMSE (0.697) is obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, $S_3 = 64$ neurons in the third layer and Adamax optimizer.

On the four architectures and all three models, Adam and Adamax are more stable than Adagrad when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3 varies from 8 to 64 neurons. In general, the hybrid model provides the lowest value of RMSE among the three models.

Table 5.3: Forecasting Results on Residential 2

							f_1 : sigmoid; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.268	0.293	0.272	0.278	0.252	0.274	0.246	0.257	0.265	0.355	0.266	0.295
64	32	16	0.257	0.294	0.264	0.272	0.242	0.275	0.256	0.258	0.278	0.290	0.267	0.278
128	64	32	0.255	0.293	0.266	0.271	0.240	0.269	0.252	0.254	0.264	0.281	0.273	0.273
256	128	64	0.259	0.298	0.267	0.274	0.251	0.267	0.251	0.256	0.267	0.277	0.283	0.276
		Avg.	0.260	0.295	0.267	0.274	0.246	0.271	0.251	0.256	0.269	0.301	0.272	0.281
							f_1 : tanh; f_2 :sigmoid ; f_3 : relu							
			time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.254	0.309	0.260	0.274	0.238	0.272	0.248	0.253	0.264	0.282	0.264	0.270
64	32	16	0.252	0.296	0.261	0.270	0.239	0.271	0.248	0.253	0.263	0.274	0.276	0.271
128	64	32	0.253	0.269	0.254	0.267	0.243	0.269	0.242	0.251	0.263	0.271	0.263	0.266
256	128	64	0.253	0.269	0.257	0.279	0.233	0.266	0.239	0.246	0.265	0.273	0.266	0.268
		Avg.	0.253	0.299	0.258	0.270	0.238	0.270	0.244	0.251	0.264	0.275	0.267	0.269
							f_1 : tanh; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.256	0.282	0.274	0.270	0.241	0.260	0.243	0.248	0.273	0.270	0.265	0.269
64	32	16	0.261	0.283	0.257	0.267	0.243	0.260	0.239	0.247	0.264	0.266	0.264	0.265
128	64	32	0.254	0.276	0.253	0.261	0.240	0.254	0.239	0.244	0.268	0.265	0.273	0.269
256	128	64	0.255	0.267	0.254	0.259	0.236	0.252	0.250	0.246	0.268	0.264	0.269	0.267
		Avg.	0.257	0.277	0.260	0.264	0.240	0.257	0.243	0.246	0.268	0.266	0.268	0.268
							f_1 : relu; f_2 :relu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.250	0.284	0.251	0.262	0.238	0.266	0.243	0.249	0.263	0.268	0.264	0.265
64	32	16	0.253	0.277	0.257	0.262	0.234	0.257	0.232	0.241	0.272	0.266	0.263	0.267
128	64	32	0.253	0.272	0.250	0.258	0.242	0.254	0.244	0.247	0.267	0.264	0.262	0.264
256	128	64	0.249	0.265	0.248	0.254	0.246	0.251	0.235	0.244	0.268	0.264	0.264	0.265
		Avg.	0.251	0.275	0.251	0.259	0.240	0.257	0.239	0.245	0.268	0.266	0.263	0.265

For residential area 2, the results in table 5.3 show that the last architecture, with f_1 :**relu** and f_2 :**relu** as activation functions, provides, in general, the best prediction and the lowest average RMSE on the three models.

For the time series model, the lowest RMSE (0.248) is obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, and $S_3 = 64$ neurons in the third layer with Adamax optimizer.

For the auto-regressive model, the lowest RMSE (0.262) is obtained with $S_1 = 128$ neurons in the first layer, $S_2 = 64$ neurons in the second, and $S_3 = 32$ neurons in the third layer with Adamax optimizer.

For the hybrid model, the lowest RMSE (0.232) is obtained with $S_1 = 64$ neurons in the first layer, $S_2 = 32$ neurons in the second, and $S_3 = 16$ neurons in the third layer with Adamax optimizer.

On the four architectures and all three models, Adam and Adamax are more stable than Adagrad when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3 varies from 8 to 64 neurons. In general, the hybrid model provides the lowest value of RMSE among the three models.

For residential area 3, the results in Table 5.4 show that the last architecture, with f_1 :**relu** and f_2 :**relu** as activation functions, provides, in general, the best prediction on all three models.

For the time series model, the lowest RMSE (1.883) is obtained with $S_1 = 128$ neurons in the first layer, $S_2 = 64$ neurons in the second, and $S_3 = 32$ neurons in the third layer with Adamax optimizer.

For the auto-regressive model, the lowest RMSE (1.736) is obtained with $S_1 = 128$ neurons in the first layer, $S_2 = 64$ neurons in the second, and $S_3 = 32$ neurons in the third layer with Adamax optimizer.

For the hybrid model, the lowest RMSE (1.720) is obtained with $S_1 = 64$ neurons

Table 5.4: Forecasting Results on Residential 3

							f_1 : sigmoid; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.907	2.091	1.916	1.971	1.725	2.010	1.724	1.820	1.752	2.002	1.747	1.834
64	32	16	1.902	2.071	1.913	1.962	1.721	2.029	1.742	1.831	1.738	2.121	1.741	1.867
128	64	32	1.898	2.073	1.898	1.965	1.746	2.270	1.721	1.912	1.753	2.311	1.742	1.935
256	128	64	1.894	2.056	1.899	1.950	1.857	2.315	1.731	1.968	1.793	2.360	1.737	1.963
		Avg.	1.900	2.073	1.907	1.960	1.762	2.156	1.730	1.883	1.759	2.199	1.742	1.900
							f_1 : tanh; f_2 :sigmoid ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.899	2.083	1.906	1.963	1.851	2.063	1.777	1.897	1.739	2.219	1.747	1.902
64	32	16	1.892	2.110	1.906	1.969	1.783	2.041	1.838	1.887	1.778	2.344	1.751	1.958
128	64	32	1.884	2.069	1.904	1.952	1.920	2.220	1.825	1.988	1.889	2.151	1.781	1.940
256	128	64	1.889	2.072	1.901	1.954	1.808	2.187	1.751	1.915	1.878	2.053	1.774	1.902
		Avg.	1.891	2.084	1.904	1.960	1.841	2.128	1.798	1.922	1.821	2.192	1.763	1.925
							f_1 : tanh; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.899	2.042	1.905	1.949	1.790	2.243	1.748	1.927	1.739	2.669	1.746	2.051
64	32	16	1.908	2.004	1.906	1.939	1.781	2.111	1.762	1.885	1.761	2.150	1.751	1.887
128	64	32	1.893	1.962	1.899	1.918	1.833	2.151	1.770	1.918	1.831	1.804	1.744	1.793
256	128	64	1.904	1.923	1.904	1.910	1.885	1.876	1.778	1.846	1.818	1.800	1.740	1.786
		Avg.	1.901	1.983	1.904	1.929	1.822	2.095	1.765	1.894	1.787	2.106	1.745	1.879
							f_1 : relu; f_2 :relu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.893	2.035	1.899	1.942	1.757	2.050	1.789	1.865	1.855	2.367	1.738	1.987
64	32	16	1.901	1.998	1.889	1.929	1.855	2.221	1.720	1.932	1.973	2.014	1.759	1.915
128	64	32	1.890	1.958	1.883	1.910	1.763	2.129	1.773	1.888	1.788	1.838	1.736	1.787
256	128	64	1.892	1.922	1.889	1.901	1.770	1.852	1.747	1.790	1.853	1.845	1.893	1.864
		Avg.	1.894	1.978	1.890	1.921	1.786	2.063	1.757	1.869	1.867	2.016	1.782	1.888

in the first layer, $S_2 = 32$ neurons in the second, and $S_3 = 16$ neurons in the third layer with Adamax optimizer.

On the four architectures and all three models, Adam and Adamax are more stable than Adagrad when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3 varies from 8 to 64 neurons. In general, the hybrid model provides the lowest value of RMSE among the three models.

For residential area 4, the results in Table 5.5 show that the last architecture, with f_1 :relu and f_2 :relu as activation functions, provides, in general, the best prediction and the lowest RMSE in average on all three models.

For the time series model, the lowest three-layer ANN RMSE obtained (1.640) is

Table 5.5: Forecasting Results on Residential 4

							f_1 : sigmoid; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.695	3.213	2.120	2.343	1.113	1.625	1.167	1.302	1.206	1.665	1.228	1.366
64	32	16	1.692	3.042	2.033	2.256	1.080	1.459	1.139	1.226	1.207	1.470	1.184	1.287
128	64	32	1.676	3.037	1.724	2.146	1.076	1.330	1.133	1.180	1.193	1.395	1.203	1.264
256	128	64	1.657	3.034	1.708	2.133	1.090	1.284	1.112	1.162	1.183	1.290	1.194	1.222
		Avg.	1.680	3.082	1.896	2.219	1.090	1.425	1.138	1.218	1.197	1.455	1.202	1.285

							f_1 : tanh; f_2 :sigmoid ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.667	2.978	1.679	2.108	1.084	1.613	1.138	1.278	1.186	1.606	1.181	1.324
64	32	16	1.677	3.043	1.697	2.139	1.080	1.331	1.102	1.171	1.171	1.317	1.186	1.225
128	64	32	1.686	3.003	1.665	2.118	1.074	1.263	1.090	1.142	1.205	1.270	1.175	1.217
256	128	64	1.696	2.597	1.651	1.981	1.072	1.246	1.073	1.113	1.197	1.260	1.211	1.223
		Avg.	1.682	2.905	1.673	2.087	1.078	1.363	1.101	1.180	1.190	1.363	1.188	1.241

							f_1 : tanh; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.675	2.919	1.740	2.111	1.078	1.339	1.127	1.181	1.186	1.286	1.209	1.227
64	32	16	1.696	2.625	1.673	1.998	1.075	1.256	1.087	1.139	1.162	1.235	1.183	1.193
128	64	32	1.665	2.498	1.674	1.946	1.080	1.229	1.084	1.131	1.193	1.201	1.179	1.191
256	128	64	1.662	2.388	1.653	1.901	1.106	1.183	1.067	1.119	1.182	1.213	1.187	1.194
		Avg.	1.675	2.608	1.685	1.989	1.085	1.252	1.091	1.142	1.181	1.234	1.190	1.201

							f_1 : relu; f_2 :relu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.640	2.866	1.680	2.062	1.090	1.376	1.086	1.084	1.169	1.251	1.177	1.199
64	32	16	1.694	2.599	1.663	1.986	1.095	1.259	1.095	1.150	1.161	1.253	1.178	1.197
128	64	32	1.666	2.408	1.652	1.909	1.080	1.183	1.080	1.114	1.168	1.215	1.158	1.180
256	128	64	1.643	2.089	1.642	1.791	1.089	1.149	1.064	1.101	1.186	1.203	1.160	1.183
		Avg.	1.661	2.491	1.659	1.937	1.089	1.242	1.081	1.137	1.171	1.231	1.168	1.190

obtained with $S_1 = 32$ neurons in the first layer, $S_2 = 16$ neurons in the second, and $S_3 = 8$ neurons in the third layer with Adam optimizer.

For the auto-regressive model, the lowest RMSE (0.158) is obtained with $S_1 = 128$ neurons in the first layer, $S_2 = 64$ neurons in the second, and $S_3 = 32$ neurons in the third layer with Adamax optimizer.

For the hybrid model, the lowest RMSE (1.064) is obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, and $S_3 = 64$ neurons in the third layer with Adamax optimizer.

On the four architectures and all three models, Adam and Adamax are more stable than Adagrad when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3

varies from 8 to 64 neurons. In general, the hybrid model provides the lowest value of RMSE among the three models.

Table 5.6: Forecasting Results on Residential 5

							f_1 : sigmoid; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.234	0.239	0.315	0.263	0.204	0.228	0.207	0.213	0.240	0.254	0.238	0.244
64	32	16	0.225	0.238	0.227	0.230	0.202	0.226	0.204	0.211	0.239	0.251	0.239	0.243
128	64	32	0.228	0.235	0.227	0.230	0.201	0.225	0.204	0.210	0.239	0.246	0.240	0.242
256	128	64	0.236	0.235	0.236	0.234	0.202	0.225	0.199	0.209	0.239	0.247	0.240	0.242
Avg.			0.231	0.237	0.251	0.239	0.202	0.226	0.204	0.211	0.239	0.250	0.239	0.243
							f_1 : tanh; f_2 :sigmoid ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.225	0.242	0.228	0.232	0.203	0.231	0.207	0.214	0.238	0.251	0.239	0.243
64	32	16	0.224	0.239	0.224	0.229	0.202	0.225	0.204	0.210	0.236	0.246	0.243	0.242
128	64	32	0.223	0.240	0.224	0.229	0.200	0.226	0.202	0.209	0.237	0.245	0.236	0.239
256	128	64	0.224	0.239	0.224	0.229	0.205	0.228	0.202	0.212	0.237	0.246	0.238	0.240
Avg.			0.224	0.240	0.225	0.230	0.203	0.228	0.204	0.211	0.237	0.247	0.239	0.241
							f_1 : tanh; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.228	0.238	0.224	0.230	0.207	0.225	0.204	0.212	0.240	0.242	0.237	0.240
64	32	16	0.226	0.236	0.224	0.229	0.200	0.220	0.205	0.208	0.242	0.239	0.238	0.240
128	64	32	0.228	0.229	0.225	0.227	0.208	0.212	0.201	0.207	0.236	0.238	0.236	0.237
256	128	64	0.227	0.227	0.223	0.226	0.208	0.207	0.199	0.205	0.240	0.238	0.238	0.239
Avg.			0.227	0.233	0.224	0.228	0.206	0.216	0.202	0.208	0.240	0.239	0.237	0.239
							f_1 : relu; f_2 :relu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	0.228	0.237	0.225	0.230	0.200	0.224	0.203	0.209	0.237	0.243	0.237	0.239
64	32	16	0.229	0.404	0.222	0.285	0.201	0.221	0.198	0.207	0.239	0.239	0.234	0.237
128	64	32	0.224	0.232	0.225	0.227	0.200	0.217	0.199	0.205	0.241	0.238	0.234	0.238
256	128	64	0.224	0.229	0.224	0.226	0.201	0.209	0.200	0.203	0.237	0.237	0.234	0.236
Avg.			0.226	0.276	0.224	0.242	0.201	0.218	0.200	0.206	0.239	0.239	0.235	0.238

For residential area 5, the results in Table 5.6 show that the last architecture, with f_1 :relu and f_2 :relu as activation functions, provides, in general, the best prediction on all three models.

For the time series model, the lowest RMSE (0.222) is obtained with $S_1 = 64$ neurons in the first layer, $S_2 = 32$ neurons in the second, and $S_3 = 16$ neurons in the third layer with Adamax optimizer.

For the auto-regressive model, the lowest RMSE (0.234) with $S_1 = 64$ neurons in the first layer, $S_2 = 32$ neurons in the second, and $S_3 = 16$ neurons in the third layer.

We have the same lowest value with $S_1 = 128$ neurons in the first layer, $S_2 = 64$ neurons in the second, $S_3 = 32$ neurons in the third layer and $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, and $S_3 = 64$ neurons in the third layer with Adamax optimizer.

For the hybrid model, the lowest RMSE (0.198) is obtained with $S_1 = 64$ neurons in the first layer, $S_2 = 32$ neurons in the second, and $S_3 = 16$ neurons in the third layer with Adamax optimizer.

On the four architectures and all three models, Adam and Adamax are more stable than Adagrad when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3 varies from 8 to 64 neurons. In general, the hybrid model provides the lowest value of RMSE among the three models.

For residential area 6, the results in Table 5.7 show that the last architecture, with f_1 :**relu** and f_2 :**relu** as activation functions, provides, in general, the best prediction and the lowest RMSE in average on all three models.

For the time series model, the lowest RMSE (0.992) is obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, and $S_3 = 64$ neurons in the third layer with Adamax optimizer.

For the auto-regressive model, the lowest RMSE (0.614) is obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, and $S_3 = 64$ neurons in the third layer with Adamax optimizer.

For the hybrid model, the lowest RMSE (0.553) is obtained with $S_1 = 256$ neurons in the first layer, $S_2 = 128$ neurons in the second, and $S_3 = 64$ neurons in the third layer with Adamax optimizer.

Table 5.7: Forecasting Results on Residential 6

							f_1 : sigmoid; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.046	1.700	1.230	1.325	0.576	0.994	0.651	0.740	0.636	0.915	0.678	0.743
64	32	16	1.023	1.597	1.055	1.225	0.573	0.802	0.583	0.653	0.633	0.809	0.631	0.691
128	64	32	1.006	1.682	1.04	1.243	0.570	0.769	0.604	0.648	0.644	0.757	0.653	0.685
256	128	64	1.009	1.682	1.014	1.235	0.600	0.702	0.595	0.632	0.633	0.713	0.654	0.667
		Avg.	1.021	1.665	1.085	1.257	0.580	0.817	0.608	0.668	0.637	0.799	0.654	0.697
							f_1 : tanh; f_2 :sigmoid ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.009	1.683	1.020	1.237	0.566	0.818	0.593	0.659	0.626	0.833	0.642	0.700
64	32	16	0.997	1.708	1.037	1.247	0.580	0.746	0.583	0.636	0.616	0.761	0.636	0.671
128	64	32	0.994	1.703	1.015	1.237	0.561	0.702	0.583	0.615	0.645	0.697	0.629	0.657
256	128	64	1.010	1.621	1.000	1.210	0.563	0.690	0.558	0.604	0.643	0.687	0.628	0.653
		Avg.	1.003	1.679	1.018	1.233	0.568	0.739	0.579	0.629	0.633	0.745	0.634	0.670
							f_1 : tanh; f_2 :selu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.024	1.603	1.045	1.224	0.565	0.691	0.597	0.618	0.627	0.700	0.634	0.654
64	32	16	1.027	1.487	1.019	1.178	0.564	0.667	0.567	0.599	0.635	0.680	0.630	0.648
128	64	32	1.006	1.342	1.042	1.130	0.572	0.658	0.556	0.595	0.657	0.670	0.630	0.652
256	128	64	0.995	1.245	1.000	1.080	0.566	0.630	0.561	0.586	0.637	0.664	0.626	0.642
		Avg.	1.013	1.419	1.027	1.153	0.567	0.662	0.570	0.600	0.639	0.679	0.630	0.649
							f_1 : relu; f_2 :relu ; f_3 : relu							
			Time series				Hybrid				Auto-regressive			
S_1	S_2	S_3	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.	Adam	Adagrad	Adamax	Avg.
32	16	8	1.000	1.576	1.020	1.199	0.562	0.711	0.580	0.618	0.635	0.755	0.634	0.671
64	32	16	1.030	1.409	1.000	1.146	0.564	0.673	0.561	0.599	0.652	0.672	0.629	0.651
128	64	32	0.995	1.252	0.995	1.081	0.567	0.629	0.556	0.584	0.645	0.663	0.621	0.643
256	128	64	0.999	1.110	0.992	1.034	0.571	0.612	0.553	0.579	0.630	0.643	0.614	0.629
		Avg.	1.006	1.337	1.002	1.115	0.566	0.656	0.563	0.595	0.641	0.683	0.625	0.649

On the four architectures and all three models, Adam and Adamax are more stable than Adagrad when S_1 varies from 32 to 256, S_2 varies from 16 to 128 neurons and S_3 varies from 8 to 64 neurons. In general, the hybrid model provides the lowest value of RMSE among the three models.

5.3 Summary of Results

Calculating the error between actual and predicted consumption is crucial to evaluate each model's performance, and make decisions on the specific model to be used to estimate future consumption.

For all results on data sets 1 and 2, the best accuracy for our three-layer ANN is obtained with the last architecture; $f_1 : \text{relu}$ and $f_2 : \text{relu}$ as activation functions. For data set 1, this is generally provided by Adam Optimizer. For data set 2, this is generally provided by Adamax Optimizer.

In particular for data set 2, with the time series model, the Adamax optimizer yields the best accuracy results for residential areas 1,2,3,5, and 6. For residential area 4, the Adam optimizer provides the best accuracy. With the auto-regressive model, the Adamax optimizer yields, the best accuracy results for residential areas 2,3,4,5, and 6. For residential area 1, the Adam optimizer provides the best accuracy. With the hybrid model, the Adamax optimizer yields the best accuracy results for all residential areas. The hybrid model provides the best accuracy (the lowest RMSE value) among the three models.

SGD fails with both data sets 1 and data set 2. SGD's main idea is based on taking random subsets of the gradient. Because we have sparse gradients, we have a lot of zeroes on every gradient subset.

The time series model is the fastest among the three models, but the lowest in accuracy. The hybrid model requires more computational time but provides the highest accuracy.

CHAPTER 6

CONCLUSION

Forecasting models contribute to a nation's long-term prosperity and development. They provide governments and organizations with the ability to prepare for future problems, exploit opportunities, and improve resource distribution for their best interests.

This thesis suggested a neural network forecasting approach for demand-side management in smart grids. The proposed neural network framework can predict energy consumption with accuracy and reliability. This is confirmed by the experimental results on two data sets, where data set 1 contains 1082 different household measurements and data set 2 contains over 30000 measurements for six residential areas. Hence, this approach can provide good predictions for spatial and temporal data sets.

Through the thesis, we investigated various three-layer neural network architectures. We found out that the most effective one uses $f : \text{relu}$ activation function in all of its three layers. The hybrid model can be used efficiently to forecast electricity consumption.

The chosen neural network would assist in energy saving, controlling demand and

supply, and effectively organizing financial support for individuals beginning their power production. Certainly, in the future, looking for better neural network architecture will lead to a smart grid with better energy demand forecasting which could help reduce the impact of issues related to the over-usage of electric appliances and power networking equipment.

REFERENCES

- [1] M. Q. Raza and A. Khosravi, “A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings,” *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, Oct. 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364032115003354>
- [2] H.S.Hippert, C.E.Pedreira, and C.R.Souza, “ Neural Networks for Short-Term Load Forecasting: A review and Evaluation ,” *IEEE Transactions on Power Systems*, vol. 16, no. 1, Feb. 2001. [Online]. Available: <https://ieeexplore.ieee.org/document/910780>
- [3] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana, “A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1460–1495, Apr. 2014. [Online]. Available: <https://ieeexplore.ieee.org/document/6803101>
- [4] J. S. M. Czapaj, R.; Kaminski, “A review of auto-regressive methods applications to short-term demand forecasting in power systems,” *Energies*, vol. 15, no. 15, p. 6729, 2022. [Online]. Available: <https://www.mdpi.com/1996-1073/15/18/6729>

- [5] A. Kheirkhah, A. Azadeh, M. Saberi, A. Azaron, and H. Shakouri, “Improved estimation of electricity demand function by using of artificial neural network, principal component analysis and data envelopment analysis,” *Computers Industrial Engineering*, vol. 64, no. 1, pp. 425–441, Jan. 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0360835212002331>
- [6] Y. Wang and J. M. Bielicki, “Acclimation and the response of hourly electricity loads to meteorological variables,” *Energy*, vol. 142, no. 23, pp. 473–485, Oct. 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0360544217317061>
- [7] M. Rana and I. Koprinska, “Forecasting electricity load with advanced wavelet neural networks,” *Neurocomputing*, vol. 182, pp. 118–132, Mar. 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231215019141>
- [8] G. Zhang, B. Eddy Patuwo, and M. Y. Hu, “Forecasting with artificial neural networks: The state of the art,” *International Journal of Forecasting*, vol. 14, no. 1, pp. 35–62, Mar. 1998. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0169207097000447>
- [9] M. Khashei, M. Bijari, and G. A. Raissi Ardali, “Improvement of autoregressive integrated moving average models using fuzzy logic and artificial neural networks (anns),” *Neurocomputing*, vol. 72, no. 4, pp. 956–967, Jan. 2009, brain Inspired Cognitive Systems (BICS 2006) / Interplay Between

- Natural and Artificial Computation (IWINAC 2007). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231208002440>
- [10] J. Anderson, J. Silverstein, S. Ritz, and R. Jones, “Distinctive features, categorical perception, and probability learning: Some applications of a neural model,” *Psychological Review*, vol. 84, no. 5, pp. 413–451, 1977. [Online]. Available: <https://psycnet.apa.org/record/1978-22353-001>
- [11] T. Kohonen, *Associative Memory: A system theoretical approach*, 1977. [Online]. Available: <https://link.springer.com/book/10.1007/978-3-642-96384-1>
- [12] K. Fukushima, “Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, Apr. 1980. [Online]. Available: <https://link.springer.com/article/10.1007/bf00344251>
- [13] J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, May 1982. [Online]. Available: <https://www.pnas.org/doi/pdf/10.1073/pnas.79.8.2554>
- [14] D. Park, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, “Electric load forecasting using an artificial neural network,” *IEEE Transactions on Power Systems*, vol. 6, no. 2, pp. 442–449, May 1991. [Online]. Available: <https://ieeexplore.ieee.org/document/76685>

- [15] N. Amjady and F. Keynia., “A new neural network approach to short term load forecasting of electrical power systems,” *Energies*, vol. 4, no. 3, pp. 488–503, Mar. 2011. [Online]. Available: <https://www.mdpi.com/1996-1073/4/3/488>
- [16] M. Macedo, J. Galo, L. Luz de Almeida, and A. Lima, “Demand side management using artificial neural networks in a smart grid environment,” *Renewable and Sustainable Energy Reviews*, vol. 41, pp. 128–133, Jan. 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S1364032114007114>
- [17] A. Baliyan, K. Gaurav, and S. K. Mishra, “A review of short term load forecasting using artificial neural network models,” *Procedia Computer Science*, vol. 48, pp. 121–125, 2015, international Conference on Computer, Communication and Convergence (ICCC 2015). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050915006699>
- [18] K. Muralitharan, R. Sakthivel, and R. Vishnuvarthan, “Neural network based optimization approach for energy demand prediction in smart grid,” *Neurocomputing*, vol. 273, pp. 199–208, Jan. 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231217313681>
- [19] C. Li, “Designing a short-term load forecasting model in the urban smart grid system,” *Applied Energy*, vol. 266, pp. 1–16, May 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0306261920303627>
- [20] E. Rahm and H. H. Do, “Data cleaning: Problems and current approaches,” *IEEE Data Eng. Bull.*, vol. 23, pp. 3–13, Jan. 2000. [Online]. Available:

<http://dbs.uni-leipzig.de>

- [21] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. Cambridge University Press, Feb. 2020. [Online]. Available: <https://www.cambridge.org/highereducation/books/mathematics-for-machine-learning/5EE57FD1CFB23E6EB11E130309C7EF98#overview>
- [22] P. Zhang and M. Qi, “Neural network forecasting for seasonal and trend time series,” *European Journal of Operational Research*, vol. 160, no. 2, pp. 501–514, Jan. 2005. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0377221703005484>
- [23] G. Dudek, “Std: A seasonal-trend-dispersion decomposition of time series,” vol. 14, no. 8, Aug. 2015. [Online]. Available: <https://arxiv.org/pdf/2204.10398.pdf>
- [24] Y. Li and H. Wu, “A clustering method based on k-means algorithm,” *Physics Procedia*, vol. 25, pp. 1104–1109, Apr. 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1875389212006220>
- [25] M. Hagan, H. Demuth, M. Beale, and O. Jesus, *Neural Network Design, 2nd Edition*, 2018. [Online]. Available: <https://hagan.okstate.edu/NNDesign.pdf>
- [26] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, Jan. 2017. [Online]. Available: <https://arxiv.org/pdf/1412.6980.pdf>

- [27] D. Soydaner, “A comparison of optimization algorithms for deep learning,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 13, Feb. 2020. [Online]. Available: <https://www.worldscientific.com/doi/abs/10.1142/S0218001420520138>

VITAE

- Name: Sara Abdalnasser Salah Eldin Al Abdallah.
- Nationality: Jordanian.
- Date of Birth: June 15, 1994.
- Email: *saraalabdallah94@gmail.com*
- Permenant Address: 8898 Ibn Jabr Street, Al Bandariyah Dist, Unit No.8, 34424 - 3278 AL Khubar, Kingdom of Saudi Arabia.

Academic qualifications:

- M.Sc. in Applied Mathematics from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, May 2023.
Thesis title: "A NEURAL NETWORK FORECASTING APPROACH FOR THE SMART GRID DEMAND RESPONSE MANAGEMENT PROBLEM".
- B.Sc. in Mathematics from Imam Abdulrahman Bin Faisal University (IAU), Dammam, Saudi Arabia, May 2016.