



YAŞAR UNIVERSITY  
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

**AIR QUALITY TIME SERIES FORECASTING  
WITH GENETIC PROGRAMMING**

SU TAŞBAŞ

THESIS ADVISOR: ASST.PROF. DR. KORHAN KARABULUT

DEPARTMENT OF COMPUTER ENGINEERING

PRESENTATION DATE: 17.05.2018

BORNOVA / İZMİR  
MAY 2018



We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

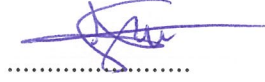
**Jury Members:**

Asst. Prof. Dr. Korhan KARABULUT  
Yaşar University

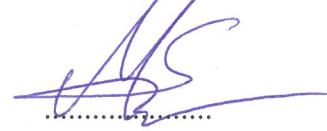
**Signature:**



Asst. Prof. Dr. Umut AVCI  
Yaşar University



Asst. Prof. Dr. Mete EMİNAĞAOĞLU  
Dokuz Eylül University



---

Prof. Dr. Cüneyt GÜZELİŞ  
Director of the Graduate School



## ABSTRACT

### AIR QUALITY TIME SERIES FORECASTING WITH GENETIC PROGRAMMING

Taşbaş, Su

MSc, Computer Engineering

Advisor: Asst. Prof. Dr. Korhan Karabulut

May 2018

In this study, air quality time series forecasting is performed by using genetic programming. Based on the reports of World Health Organization and the other environmental agencies, it has been shown how crucial air quality forecasting is to prevent deaths and health issues caused by air pollution. The primary aim of this study is to contribute to increasing the usage of genetic programming for air quality forecasting and to show its competitiveness with several machine learning algorithms and autoregressive integrated moving average (ARIMA) model. The hourly meteorological data for one-year length is utilized to forecast sulphur dioxide and particulate matter gas concentrations. The forecasting problem was identified as a symbolic regression problem and the Java-based Evolutionary Computation Research system (ECJ) was utilized to apply genetic programming to the problem. In order to demonstrate the performance of genetic programming, the forecast results were compared to the results that were collected from several decision tree algorithms and an ARIMA model. The comparisons showed that genetic programming performed better even than the ensemble learning method.

**Key Words:** genetic programming, air quality, time series forecasting



## ÖZ

### GENETİK PROGRAMLAMA İLE HAVA KALİTESİ ZAMANA BAĞLI SERİ TAHMİNLEME

Taşbaş, Su

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü

Danışman: Dr. Öğrt. Üyesi Korhan Karabulut

Mayıs 2018

Bu çalışmada, genetik programlama kullanılarak hava kalitesi zamana bağlı seri tahminleme gerçekleştirilmiştir. Dünya Sağlık Örgütü ve diğer çevre ajanslarının raporlarına dayanarak, hava kirliliğinin neden olduğu ölümleri ve sağlık sorunlarını önlemek için hava kalitesi tahminlemenin ne kadar önemli olduğu gösterilmiştir. Bu çalışmanın temel amacı, hava kalitesi tahmini için genetik programlamanın kullanımını arttırmaya ve makine öğrenmesi yöntemleri ve otoregresif bütünleşmiş hareketli ortalama (ARIMA) ile yarışabilirliğini göstermeye katkıda bulunmaktır. Çalışmada bir yıl süreyle saatlik olarak ölçülmüş meteorolojik veriler kükürt dioksit ve parçacık madde gaz yoğunlaşmalarını tahmin etmek için kullanılmıştır. Zamana bağlı seri tahminleme problemi sembolik regresyon problemi olarak tanımlanmış ve Java tabanlı Evrimsel Hesaplama Araştırma sistemi (ECJ) kullanılmıştır. Tahminleme sonuçları genetik programlamanın performansını göstermek için çeşitli karar ağacı algoritmalarından ve ARIMA modelinden elde edilen sonuçlarla karşılaştırılmıştır. Karşılaştırmalar genetik programlamanın topluluk öğrenme yönteminden bile daha iyi performans sergilediğini göstermiştir.

**Anahtar Kelimeler:** genetik programlama, hava kalitesi, zamana bağlı seri tahminleme





## **ACKNOWLEDGEMENTS**

First, I would like to thank my supervisor Korhan Karabulut for his guidance and patience during this study. I would also like to thank Prof. Dr. Tolga Elbir from Department of Environmental Engineering of Dokuz Eylül University for providing us data and helping us in understanding the details.

I would like to express my enduring love to my parents, who are always supportive, loving and caring to me in every possible way in my life.

Su TAŞBAŞ

İzmir, 2018



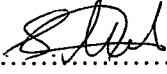


## TEXT OF OATH

I declare and honestly confirm that my study, titled “AIR QUALITY TIME SERIES FORECASTING WITH GENETIC PROGRAMMING” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Su TAŞBAŞ

Signature

  
.....

May 26, 2018



## TABLE OF CONTENTS

ABSTRACT .....	v
ÖZ .....	vii
ACKNOWLEDGEMENTS .....	ix
TEXT OF OATH .....	xi
TABLE OF CONTENTS .....	xiii
LIST OF FIGURES .....	xv
LIST OF TABLES .....	xvii
SYMBOLS AND ABBREVIATIONS .....	xix
CHAPTER 1 INTRODUCTION .....	1
CHAPTER 2 BACKGROUND .....	5
2.1. Air Quality .....	5
2.2. Time Series Forecasting .....	11
2.3. Genetic Programming .....	14
CHAPTER 3 PREVIOUS WORK .....	21
3.1. Genetic Programming .....	21
3.2. ARIMA, ANN and FTS .....	23
3.3. Data Mining Algorithms Used in This Study .....	25
3.4. Air Quality Modeling Systems .....	26
CHAPTER 4 IMPLEMENTATION .....	29
4.1. Data Preparation .....	29
4.2. Defining the Problem as a Symbolic Regression Problem .....	31
4.3. Methodology of Training and Evaluating Prediction Performance .....	32
4.4. Parameter Setting .....	35
4.5. Performance Evaluation of the Genetic Programming Model .....	39
CHAPTER 5 CONCLUSIONS AND FUTURE RESEARCH .....	45
REFERENCES .....	49
APPENDIX 1 – ECJ Function Set Parameter Settings .....	55
APPENDIX 2 – Experimental Design Plots .....	61
APPENDIX 3 – GP and Decision Tree Algorithm Results .....	67

APPENDIX 4 – R Code used for ARIMA forecasting ..... 75

APPENDIX 5 – Example GP Tree Graphs ..... 77



## LIST OF FIGURES

<b>Figure 2.1.</b> AQI Levels (USEPA, 2014) .....	6
<b>Figure 2.2.</b> AQI Health Advise (EPA USA, 2007) .....	6
<b>Figure 2.3.</b> Pollutant-Specific Sensitive Groups (USEPA, 2013).....	7
<b>Figure 2.4.</b> AQI Breakpoints (USEPA, 2013) .....	9
<b>Figure 2.5.</b> AirNow AQI Calculator (USEPA, n.d.) .....	10
<b>Figure 2.6.</b> Real-time AQI (WAQI, n.d.).....	11
<b>Figure 2.7.</b> Time Series Patterns (Krajewski, Ritzman, & Malhotra, 2007).....	12
<b>Figure 2.8.</b> Fundamental steps of genetic programming (Koza et al., 2003) .....	16
<b>Figure 2.9</b> GP syntax tree representation of $\max(x + x, x + 3*y)$ (Poli et al., 2008).....	18
<b>Figure 2.10</b> Examples of genetic programming operations (Lane, Sozou, Addis, & Gobet, 2016) .....	19
<b>Figure 4.1.</b> Train and test sets split points.....	31
<b>Figure 4.2.</b> If-then-else operation.....	32
<b>Figure 4.3.</b> The first experimental design generated by R DoE plugin.....	37
<b>Figure 4.4.</b> The second experimental design generated by R DoE plugin .....	39
<b>Figure A2.1.</b> The interval plots of the parameters used in the first experiment for the RMSE measurements of PM forecasting with GP when the maximum tree depth is seven. ....	61
<b>Figure A2.2.</b> The interval plots of the parameters used in the first experiment or the RMSE measurements of SO <sub>2</sub> forecasting with GP when the maximum tree depth is seven. ....	62
<b>Figure A2.3.</b> The interval plots of the parameters used in the used in the first experiment for correlation coefficient (R) measurements of PM forecasting with GP when the maximum tree depth is seven. ....	63
<b>Figure A2.4.</b> The interval plots of the parameters used in the used in the first experiment for correlation coefficient (R) measurements of SO <sub>2</sub> forecasting with GP when the maximum tree depth is seven. ....	64

**Figure A2.5.** The interval plots of the parameters used in the second experiment for the RMSE measurements of PM forecasting using GP .....65

**Figure A2.6.** The interval plots of the parameters used in the second experiment for the RMSE measurements of SO<sub>2</sub> forecasting using GP.....65

**Figure A2.7.** The interval plots of the parameters used in the second experiment for the correlation coefficient (R) measurements of PM forecasting using GP .....66

**Figure A2.8.** The interval plots of the parameters used in the second experiment for the correlation coefficient (R) measurements of SO<sub>2</sub> forecasting using GP.....66

**Figure A5.1.** SO<sub>2</sub> Tree Graph .....77

**Figure A5.2.** PM Tree Graph.....77





## LIST OF TABLES

<b>Table 4.1.</b> t-test comparisons over averages of 25 runs of GP and decision tree algorithms	41
<b>Table 4.2.</b> ARIMA forecast results .....	43
<b>Table A3.1.</b> GP's PM forecasting results .....	67
<b>Table A3.2.</b> Random Forest's PM forecasting results .....	68
<b>Table A3.3.</b> Random Tree's PM forecasting results .....	69
<b>Table A3.4.</b> Rep Tree's PM forecasting results .....	70
<b>Table A3.5.</b> GP's SO <sub>2</sub> forecasting results .....	71
<b>Table A3.6.</b> Random Forest's SO <sub>2</sub> forecasting results .....	72
<b>Table A3.7.</b> Random Tree's SO <sub>2</sub> forecasting results .....	73
<b>Table A3.8.</b> Rep Tree's SO <sub>2</sub> forecasting results .....	74



## SYMBOLS AND ABBREVIATIONS

### ABBREVIATIONS:

WHO	World Health Organization
EPA	Environmental Protection Agency
EEA	European Environment Agency
ANN	Artificial Neural Networks
ML	Machine Learning
GP	Genetic Programming
SO <sub>2</sub>	Sulphur Dioxide
PM	Particulate Matter
ECJ	Evolutionary Computation for Java
DOE	Design of Experiments
FTS	Fuzzy Time Series
ERC	Ephemeral Random Constant
ARIMA	Autoregressive Integrated Moving Average



# CHAPTER 1

## INTRODUCTION

Air quality is one of the most important health problems nowadays to people of all ages. Air pollution affects all countries and societies. Ambient air pollution kills about three million people per year (World Health Organization, 2016). Approximately 94% of worldwide deaths occur due to the non-communicable diseases in adults. Examples are cardiovascular diseases (heart attack and ischemic heart disease), chronic obstructive pulmonary disease and lung cancer. The remaining deaths are due to acute lower respiratory tract infections in children under five years of age (World Health Organization, 2016).

Based on the information gathered from the reports of World Health Organization (WHO) and the other environmental agencies, it has been revealed how critical air quality forecasting is in order to avoid deaths and health problems caused by air pollution. This study contributes to the solution of the problem identified in this direction.

The ambient air quality is decreasing for certain reasons such as warming in the cold seasons, traffic pollution and harmful gas emissions near the residential section of the industry. The air quality fluctuates with the lower or higher concentration of the pollutant gases in the atmosphere over a period. Therefore, air quality forecasting is an example of the time series forecasting. Time series forecasting is mainly performed by artificial neural networks (ANN) and the traditional forecasting methods such as ARIMA (Graff, Escalante, Ornelas-Tellez, & Tellez, 2016). In this research, genetic programming (GP) is applied to an air quality time series-forecasting problem.

The Environmental Engineering Department of Dokuz Eylül University delivered air quality data used in this research. It involves one-year of hourly measures of meteorological attributes, Sulphur dioxide (SO<sub>2</sub>) and particulate matter (PM) gases that were recorded in 2013. The meteorological attributes are temperature, air pressure, humidity, wind speed, and wind direction. Time stamping attributes, such as the year, month, day and hour are included as well.

As an initial step, data pre-processing was applied to the data. First, it was decided which attributes should be discarded. Year and month information was removed. Instead, weekday attribute was included and utilized for input. Second, non-numerical attributes were transformed to numerical values. For example, wind direction attribute instances were in the form of compass directions. Additionally, weekday and hour attributes were trivial to use as numerical. Therefore, these categorical attributes were transformed to the numerical values by disjunctive coding technique. New columns were included referring to the conversion. Sixteen columns for wind direction attribute, twenty-four columns for hour attribute and seven columns for weekday attribute were added. For example, wind direction attribute has sixteen divisions such as northeast(NE) and south-southwest (SSW). Therefore, sixteen columns took the place of the wind direction attribute. The same process was applied to the hour attribute and weekday attribute.

The second part of the pre-processing was normalization. After all the attributes were transformed to numerical values, normalization was applied to the whole data. To accomplish this, minimum, maximum and standard deviation values of all occurrences were found and utilized in normalization calculations.

The next part of the pre-processing was adding moving average values of both observed Sulphur dioxide and particulate matter gases. Moving average values of gases were accepted as input attributes. The final pre-processing step was breaking down the whole data into training and test sets. Training and test set partitioning was performed by trend evaluation. Sulphur dioxide and particulate matter gases have different trend tracks. Therefore, percentage splits show a variance.

After pre-processing was completed, the final version of data was ready to use in Evolutionary Computation for Java (ECJ) framework (Luke, 1998). With ECJ, the problem was set as a symbolic regression problem. All the attributes and operations, such as addition and multiplication, were defined as node object classes. Function set and other genetic programming settings of the problem were constructed in the parameter files. In the function set, attributes were applied as terminals and operations are accepted as non-terminals. Fitness evaluation was carried out on training sets. Error measures were calculated and written to both statistics and text files from test sets fitness evaluations. Recursive pre-process operations for the nodes were implemented

on the tree structures. Trees were written in dot style and visualized by Graphviz (Ellson, Gansner, Koutsofios, North, & Woodhull, 2002) application.

After initial experiments with the default ECJ Koza parameter setting, it turned out that the results could be better according to the data mining algorithms' results. Moreover, windowing approach was applied to the observed pollutant values as input but it made no progress. Therefore, the design of experiments (DoE) method was practiced. The experiments were handled in three stages. Different genetic programming parameters with varying values were attempted. Crossover, mutation, tournament selection, elitism and population size are example genetic programming parameters of experiment combinations.

At the final stage, the experiment results were compared to the Weka classifier tree algorithm runs. A batch file was formed and covered three classifier algorithms' contexts. Each algorithm ran for twenty-five times, same as the genetic programming experiments. Moreover, the genetic programming model was compared to an ARIMA model to observe the performance comparison of genetic programming in compliance with a traditional approach for the time series forecasting.

The contribution of this research is as follows:

- In the previous air quality forecasting studies, the Genetic Programming approach has been applied less than other methods (traditional approaches such as ARIMA, and artificial neural networks).
- Genetic programming produced better results than ensemble learning methods such as Random Forests and Random Tree.
- The distinctive advantage of genetic programming is producing a tree model as the problem solution.

There are several issues as the future work of this research. The initial point is the result models of the implemented genetic programming program. After final experiments, twenty-five distinct models are developed. The tree structures in the models might be expanded by changing the tree breeding algorithms in genetic programming. In this research, ECJ library was used as it is. Modifying the ECJ's breeding functions may further improve generated tree models in forthcoming studies.

The second point is the models themselves. Neural networks are black box systems. Therefore, genetic programming is a stronger choice for circumstances which may need model analyses. The generated tree models may be investigated for air quality study. For example, the meteorological attributes used in the result trees may provide a view to further researches on the same or comparable data.

The third point is the design of experiments. In this research, three experiments were performed and each of them involves various combinations of genetic programming parameters. The tested parameter combinations in this research might be utilized to determine the parameter combinations of future experiments.

Another improvable argument of the research is the moving average computation technique. In this research, the simple moving average calculation was utilized as input for forecasting. For further explorations, other moving average calculations, such as cumulative moving average or weighted moving average, may yield better results.

The final objective is that new air quality time series forecasting might be implemented with genetic programming by practicing this research. As it was pointed out before, air quality estimating is not commonly studied by genetic programming. Therefore, such as traffic-related air quality time series data might be inspected with the same procedures in this research.

The thesis outline was planned as follows. Chapter 1 is the introduction of the research. Chapter 2 contains background information such as air quality, time series forecasting and genetic programming. Chapter 3 involves the previous research. In chapter 4, the implementation and the thesis study was explained in detail. Chapter 5 includes the conclusion and future research.



## **CHAPTER 2**

### **BACKGROUND**

#### **2.1. Air Quality**

Clean air is one of the most vital needs for a healthy society. Therefore, in recent years, air quality forecasting has awakened considerable social awareness. Developing online and offline systems of pollutant gas concentrations to monitor and forecast are crucial in terms of preventing serious health problems. Major attempts have been performed by academics in air quality forecasting which mostly focus on the concentrations forecasting of PM<sub>10</sub>, PM<sub>2.5</sub>, SO<sub>2</sub>, NO<sub>x</sub>, CO and O<sub>3</sub> (Wang, Wei, Luo, Yue, & Grunder, 2017).

For several years, modelling methods based on analytical and numerical approaches have been applied further and further repeatedly to analyze the relations between air pollution and disorders or deaths, causing serious disputes on the importance of the data gathered and on the demand for proper education of professionals in these fields (Oliveri Conti, Heibati, Kloog, Fiore, & Ferrante, 2017).

Air Quality Index (AQI) is a number to state the quality of air. It reports the pollution degree of the air, and the possible health effects which may occur. The AQI is mainly utilized to how people may be affected within a few hours or days after being exposed to pollutants in the air. Air quality index may vary from country to country. Countries have diversified levels with different ranges of the air quality indices according to their national air quality criterions.

Air quality indexes are sorted by categories and each category have their own color, description, and health-related advises or warning. Figure 2.1 and Figure 2.2 show the related AQI values, corresponding summary and details of concerns and related color that are used in the USA.

Air Quality Index (AQI) Values	Levels of Health Concern	Colors
<i>When the AQI is in this range:</i>	<i>..air quality conditions are:</i>	<i>...as symbolized by this color:</i>
0 to 50	Good	Green
51 to 100	Moderate	Yellow
101 to 150	Unhealthy for Sensitive Groups	Orange
151 to 200	Unhealthy	Red
201 to 300	Very Unhealthy	Purple
301 to 500	Hazardous	Maroon

**Figure 2.1.** AQI Levels (USEPA, 2014)

Air Quality Index Levels of Health Concern	Numerical Value	Meaning
Good	0 to 50	Air quality is considered satisfactory, and air pollution poses little or no risk.
Moderate	51 to 100	Air quality is acceptable; however, for some pollutants there may be a moderate health concern for a very small number of people who are unusually sensitive to air pollution.
Unhealthy for Sensitive Groups	101 to 150	Members of sensitive groups may experience health effects. The general public is not likely to be affected.
Unhealthy	151 to 200	Everyone may begin to experience health effects; members of sensitive groups may experience more serious health effects.
Very Unhealthy	201 to 300	Health alert: everyone may experience more serious health effects.
Hazardous	301 to 500	Health warnings of emergency conditions. The entire population is more likely to be affected.

**Figure 2.2.** AQI Health Advise (EPA USA, 2007)

The higher values of the AQI indicate the more air pollution and the more critical health problems might be seen. Being exposed to air pollution for both short-period of time, such as over a few hours or days, and long-period of time, which is over months or years, is relevant to the health effects. These health effects are acute health conditions, which occur with short-term exposure, and persistent health impacts, which occur with long-term exposure (EEA, 2017).

In addition, according to USEPA’s technical assistance document published in 2013 (USEPA, 2013), pollutant sensitive groups are categorized on the basis of which

groups can be affected after the air quality exceeds 100, that is when it starts to become unhealthy for sensitive groups. The groups are as shown Figure 2.3.

When this pollutant has an AQI above 100...	Report these Sensitive Groups
Ozone	People with lung disease, children, older adults, people who are active outdoors (including outdoor workers), people with certain genetic variants, and people with diets limited in certain nutrients are the groups most at risk
PM2.5	People with heart or lung disease, older adults, children, and people of lower socioeconomic status are the groups most at risk
PM10	People with heart or lung disease, older adults, children, and people of lower socioeconomic status are the groups most at risk
CO	People with heart disease is the group most at risk
NO2	People with asthma, children, and older adults are the groups most at risk
SO2	People with asthma, children, and older adults are the groups most at risk

**Figure 2.3.** Pollutant-Specific Sensitive Groups (USEPA, 2013)

According to the European Environment Agency’s report in 2017, the major sectors contributing most to air pollutant emissions in Europe are transportation, commercial, institutional and household sectors, energy production and distribution, industrial energy use, industrial processes and product use, agriculture and waste. Moreover, traffic and household emissions, which are described as emission sectors with low emission heights, usually have more impact to the health effects and surface densities in the urban sections than high emission heights (EEA, 2017).

In Turkey, air pollution is a severe environmental problem due to rapid urbanization and economic growth. In some major urban areas such and industrial centers, air pollution threatens health because SO<sub>2</sub> and particulate concentrations in these areas are above the national air quality standards (Büke & Köne, 2016; OECD, 2008). The national air quality index was created by adapting the EPA air quality index to Turkey's national legislation and boundary values. The national boundary values of nearly all pollutants utilized for air quality index calculation exceed EPA’s boundary values. In Turkey, the Ministry of Environment and Urbanization provides real time air quality index data by cities through the national air quality monitoring network (MoEU, n.d.).

Air pollutants can be classified as primary or secondary. Primary pollutants are straightly transmitted to the atmosphere. Primary pollutants are primary PM, BC, Sulphur oxides (SO<sub>x</sub>), NO<sub>x</sub> (which consists of both NO and NO<sub>2</sub>), NH<sub>3</sub>, CO, methane (CH<sub>4</sub>), NMVOCs, C<sub>6</sub>H<sub>6</sub>, particular metals, and polycyclic aromatic hydrocarbons (PAH, including BaP). Secondary air pollutants are built in the atmosphere from

precursor pollutants. Secondary pollutants have secondary PM, O<sub>3</sub> and NO<sub>2</sub>. Air pollutants might have a natural, anthropogenic or varied basis, resting on their causes or the origins of their harbingers (EEA, 2017).

To date, the biggest environmental risk for health, which is responsible for about one in every nine deaths annually, is air pollution as both ambient (outdoor) and household (indoor). Ambient (outdoor) air pollution, primarily from non-communicable diseases, kills about 3 million people each year. Only one in ten people live in a city that complies with the WHO Air Quality Act (World Health Organization, 2016). Air pollution affects economies and societies' quality of life and carries on to rise in a worrisome manner. Therefore, air pollution is a public health emergency.

Since air pollution sources also produce pollutants that cause climate change (e.g. CO<sub>2</sub> or black carbon), air pollution is used as a sign of sustainable development. Policies for air pollution provide numerous benefits to human health. In addition, these benefits apply not only to air quality improvements but also to other health benefits such as the prevention of injury or the activation of physical activity.

The first and essential step to be taken by public authorities at both national and municipality levels is to monitor air quality to overwhelm the multi-sectoral challenge of air pollution. Besides, there are still large monitoring and reporting differences between high-wage countries and low- and middle-wage counterparts. The vast majority of air quality data still comes from Europe and the United States; Africa, South East Asia and the Eastern Mediterranean remain behind.

Non-communicable diseases in adults, such as cardiovascular disorders (stroke and ischemic heart disease), chronic obstructive pulmonary disease and lung cancer, cause about 94% of deaths worldwide. Acute lower respiratory tract infections observed in children under the age of five are the cause of remaining deaths. Ambient air pollution affects all regions of the world, even though, West Pacific and Southeast Asia are the most effected regions. Diseases caused by ambient air pollution kill about 3 million people per year (World Health Organization, 2016).

Air quality index is calculated by using a piecewise linear function of the polluted concentration. The equation is:

$$I_p = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}} (C_p - BP_{Lo}) + I_{Lo} \quad (1)$$

where,

$I_p$  = the index for pollutant  $p$ ,

$C_p$  = the truncated concentration of pollutant  $p$ ,

$BP_{Hi}$  = the concentration breakpoint that is greater than or equal to  $C_p$ ,

$BP_{Lo}$  = the concentration breakpoint that is less than or equal to  $C_p$ ,

$I_{Hi}$  = the AQI value corresponding to  $BP_{Hi}$ ,

$I_{Lo}$  = the AQI value corresponding to  $BP_{Lo}$ .

Table of breakpoints shown in Figure 2.4 for the AQI level is used for finding the concentration and equation result index breakpoints. If there are more than one pollutant measurements, then the highest AQI value is accepted.

These Breakpoints...							...equal this AQI	...and this category
O <sub>3</sub> (ppm) 8-hour	O <sub>3</sub> (ppm) 1-hour <sup>1</sup>	PM <sub>2.5</sub> (µg/m <sup>3</sup> ) 24-hour	PM <sub>10</sub> (µg/m <sup>3</sup> ) 24-hour	CO (ppm) 8-hour	SO <sub>2</sub> (ppb) 1-hour	NO <sub>2</sub> (ppb) 1-hour	AQI	
0.000 - 0.054	-	0.0 - 12.0	0 - 54	0.0 - 4.4	0 - 35	0 - 53	0 - 50	Good
0.055 - 0.070	-	12.1 - 35.4	55 - 154	4.5 - 9.4	36 - 75	54 - 100	51 - 100	Moderate
0.071 - 0.085	0.125 - 0.164	35.5 - 55.4	155 - 254	9.5 - 12.4	76 - 185	101 - 360	101 - 150	Unhealthy for Sensitive Groups
0.086 - 0.105	0.165 - 0.204	(55.5 - 150.4) <sup>3</sup>	255 - 354	12.5 - 15.4	(186 - 304) <sup>4</sup>	361 - 649	151 - 200	Unhealthy
0.106 - 0.200	0.205 - 0.404	(150.5 - 250.4) <sup>3</sup>	355 - 424	15.5 - 30.4	(305 - 604) <sup>4</sup>	650 - 1249	201 - 300	Very unhealthy
( <sup>2</sup> )	0.405 - 0.504	(250.5 - 350.4) <sup>3</sup>	425 - 504	30.5 - 40.4	(605 - 804) <sup>4</sup>	1250 - 1649	301 - 400	Hazardous
( <sup>2</sup> )	0.505 - 0.604	(350.5 - 500.4) <sup>3</sup>	505 - 604	40.5 - 50.4	(805 - 1004) <sup>4</sup>	1650 - 2049	401 - 500	Hazardous

**Figure 2.4.** AQI Breakpoints (USEPA, 2013)

<sup>1</sup> In some areas, AQI based on 1-hour ozone values would be more protective measure than 8-hour vales. In these cases, 1-hour ozone value might be considered when calculating the AQI.

<sup>2</sup> Higher AQI values ( $\geq 301$ ) are not determined by 8-hour O<sub>3</sub> values. 1-hour O<sub>3</sub> concentrations are used for calculating the AQI values of 301 or higher.

<sup>3</sup> These numbers might differ according to the different promulgated SHL for PM<sub>2.5</sub>.

<sup>4</sup> Higher AQI values ( $\geq 200$ ) are not identified by 1-hour SO<sub>2</sub> values. 24-hour SO<sub>2</sub> concentrations are used for calculating the AQI values of 200 or greater (USEPA, 2013).

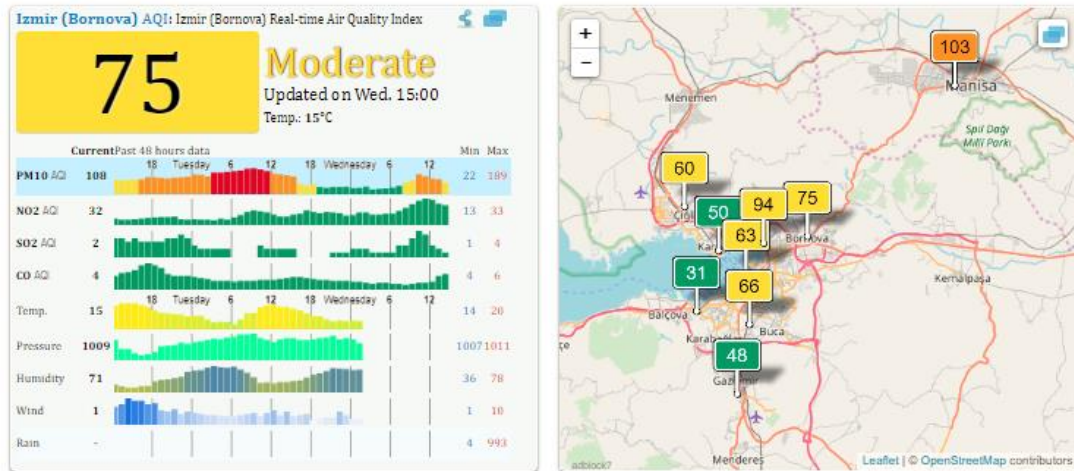
There are also online AQI calculators available. For example, US Environmental Protection Agency (USEPA) has an AQI calculator at the AirNow website, which is shown in (Figure 2.5).

The screenshot displays the AirNow AQI Calculator interface. At the top, there are two tabs: 'AQI to Concentration' and 'Concentration to AQI', with the latter being active. Below the tabs, a dropdown menu labeled 'Select a Pollutant' is set to 'SO2 - Sulfur Dioxide (1hr avg)'. Underneath, the 'Units' are set to 'ppb'. The 'Enter the Concentration' field contains the value '80', with 'Calculate' and 'Reset' buttons to its right. The results section shows an 'AQI' of '103' and an 'AQI Category' of 'Unhealthy for Sensitive Groups'. Below this, there are three columns: 'Sensitive Groups', 'Health Effects Statements', and 'Cautionary Statements', each containing specific text related to the pollutant and concentration.

**Figure 2.5.** AirNow AQI Calculator (USEPA, n.d.)

Online air quality forecasting systems might give warnings for possible health complications that may be associated with air quality. For example, World Air Quality

Index project's website (Retrieved March 28, 2018, from <http://aqicn.org>) is available to monitor instant or forecasted AQI values by countries and cities (Figure 2.6).



**Figure 2.6.** Real-time AQI (WAQI, n.d.)

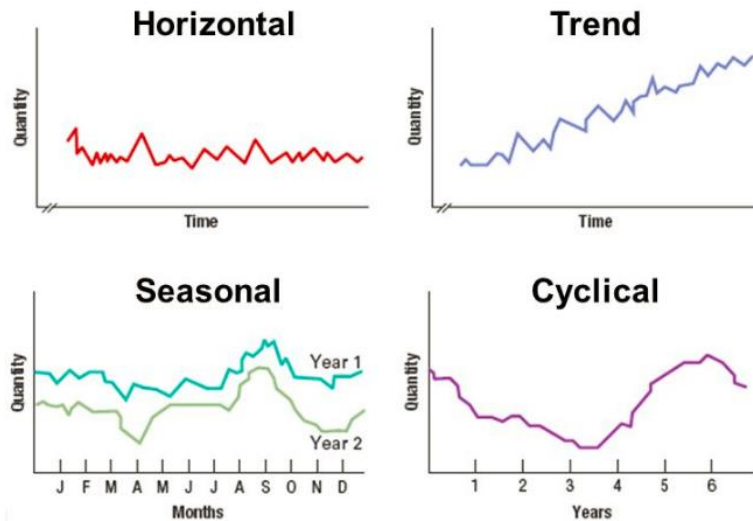
## 2.2. Time Series Forecasting

A time series is a collection of observations, each recorded at a particular time. If observations are made at constant time periods, the time series is a discrete-time time series. When observations are acquired perpetually over some time period, the time series is a continuous-time time series (Brockwell & Davis, 2002). A time series model is used for forecasting future values by using observed time series values. Time series data might have typical characteristics such as trend, seasonality and cyclical pattern.

A trend is a tendency or continuous alteration to comparatively greater or smaller measures over a long duration. Each trend in time series is called as a trend cycle and a trend might not be linear (R. J Hyndman & Athanasopoulos, 2013). When a trend pattern demonstrates a general increasing aspect, it is called as an uptrend, and when a trend pattern displays a general descending aspect, it is called as a downtrend. Time series may have varying aspect move from an uptrend to a downtrend. Horizontal trends may occur when there is no trend.

A seasonal pattern or seasonality occurs when a time series, which shows a repeating pattern at stationary periods, is affected by seasonal elements such as seasons of the year, holidays, etc. If the variations in a time series do not belong to a fixed time period, the cyclical pattern appears (R. J Hyndman & Athanasopoulos, 2013). Illustrations of different patterns are given in Figure 2.7.





**Figure 2.7.** Time Series Patterns (Krajewski, Ritzman, & Malhotra, 2007)

Time series are used in statistics, management, health, tourism, energy, pollution, manufacturing, etc. (Aladag & Egrioglu, 2012). Time series forecasting is widely implemented as financial forecasting (Sapankevych & Sankar, 2009), gene expression (Bar-Joseph, 2004), meteorological forecasting such as ozone forecasting (Paci, Gelfand, & Holland, 2013), sea water level forecasting (Ali Ghorbani, Khatibi, Aytekin, Makarynskiy, & Shiri, 2010) and wind speed forecasting (Graff, Pena, & Medina, 2013), air quality forecasting (Rahman, Lee, Suhartono, & Latif, 2015).

The most common methods used for time series forecasting are traditional methods such as simple moving average, exponential smoothing and autoregressive integrated moving average (ARIMA) (Poncela, 2004). ARIMA model was proposed by Box–Jenkins in 1976 and it is one of the first time series models. ARIMA is extensible for autoregressive (AR), moving average (MA), autoregressive moving average (ARMA), and autoregressive integrated moving average (ARIMA). The crucial restriction of the ARIMA model is that the model can only perform efficiently with the stationary and linear time series (Rahman et al., 2015).

Esling & Agon (2012) and Fu (2011) survey data mining methods applied to time-series data. The other popular time series forecasting methods are neural networks (Qiu, Zhang, Ren, Suganthan, & Amaratunga, 2014), support vector machines (Sapankevych & Sankar, 2009), and evolutionary algorithms such as genetic programming (Santini & Tettamanzi, 2001) (Ahalpara, 2010). Additionally, combinations of these methods are getting more and more prevailing to achieve better



results in time series forecasting (Cortez, Rocha, & Neves, 2001) (Aladag & Egrioglu, 2012) (Ferreira, Vasconcelos, & Adeodato, 2007) (Lee & Tong, 2011).

Moreover, there are several software tools utilized for time series forecasting. For example, R Project (R Core Team, 2017) has a software package called as forecast, which can apply automatic ARIMA model applications on time series data (Rob J. Hyndman & Khandakar, 2008). Furthermore, Weka (Hall et al., 2009) has a software package called time series forecasting to model time series and provides sample data to experiment with time series modelling and forecasting (Bouckaert et al., 2013).

To assess the forecast performance of a time series model, the measure of differences between the observed values and the values predicted by the model is utilized. There are several methods used for calculating the forecast error.

Pearson coefficient correlation (R) is a statistical measure to show the statistical relevance between two values in the range of -1 to 1. While values near to 1 indicate a strong correlation, values approaching -1 indicate an opposite correlation, and values around 0 show a weak relationship. The formula is:

$$R = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum(X - \bar{X})^2} \sqrt{\sum(Y - \bar{Y})^2}} \quad (2)$$

$X$  indicates observed values,  $\bar{X}$  indicates mean of  $X$ ,  $Y$  indicates predicted values and  $\bar{Y}$  indicates mean of  $Y$ .

Let  $\hat{\theta}_i$  indicate predicted values,  $\theta_i$  indicates observed values and  $\bar{\theta}$  indicates a mean value of  $\theta_i$  for the rest of the error measure formulas.

The root-mean-square error (RMSE) is a commonly used measure of the differences between forecasted and observed values. The formula is:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{\theta}_i - \theta_i)^2} \quad (3)$$

In time series forecasting, mean absolute error (MAE), which is also used in statistics as well, is used as a measurement of differences between observed and predicted values. The formula is:

$$MAE = \frac{1}{N} \sum_{i=1}^N |\hat{\theta}_i - \theta_i| \quad (4)$$

The relative absolute error (RAE) is the significance of the difference between the observed and the predicted values in terms of percentage. The higher relative absolute error indicates the bigger difference between the actual and predicted values. The formula is:

$$RAE = \frac{\sum_{i=1}^N |\hat{\theta}_i - \theta_i|}{\sum_{i=1}^N |\bar{\theta} - \theta_i|} \quad (5)$$

Similarly, root relative squared error (RRSE) is another used forecasting accuracy measurement. The formula is:

$$RRSE = \sqrt{\frac{\sum_{i=1}^N (\hat{\theta}_i - \theta_i)^2}{\sum_{i=1}^N (\bar{\theta} - \theta_i)^2}} \quad (6)$$

The certainty of estimates conducted by a time series model might particularly be established by examining how closely the model operates on new data which are not utilized when fitting the model (Hyndman & Athanasopoulos, 2013). For model fitting and model testing, the time series data are split into a training set and a test set. The time series model is constructed based on the training time series data and forecasting performance of the model is evaluated on the test time series data.

### 2.3. Genetic Programming

Genetic programming (GP) is an assortment of evolutionary computation approaches, which provide computers to solve problems systematically without demanding the user to determine the formation of the solution beforehand (Poli, Langdon, & McPhee, 2008). Genetic programming is an endeavor to answer one of the essential questions in computer science as follows:

“How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what needs to be done, without being told exactly how to do it?” (Koza, 1994).

The question was answered as that computers can be set up by natural selection. Especially, it was explained that the field-independent genetic programming paradigm can develop computer programs which solve or nearly solve diverse problems in distinct fields (Koza, 1992).

Genetic programming is an improvement over the traditional genetic algorithm. The genetic algorithm makes a population of individuals into a new generation of the population. Each individual in the population has a related fitness value. The genetic algorithm adopts the Darwinian doctrine of reproduction and survival of the fittest and naturally resulting genetic activities such as crossover (recombination) and mutation. In genetic programming, each individual of a population refers a computer program (Koza, 1992).

Genetic programming has a wide range of application areas. For example, human-competitive studies are antenna design (Lohn et al., 2003) and search algorithm evaluation in Chess (Hauptman & Sipper, 2007). Moreover, image and signal processing with genetic programming is commonly applied. The examples are satellite image processing for environmental studies (Chami & Robilliard, 2002) and signal processing algorithm evaluations (Holladay & Robbins, 2007). Furthermore, financial trading (Austin, Bates, Dempster, Leemans, & Williams, 2004; Potvin, Soriano, & Maxime, 2004) and economic modelling (Duyvesteyn & Kaymak, 2005) are one of the most popular practiced subjects in genetic programming.

In addition, genetic programming is widely used in medicine, such as cancer research (Worzel, Yu, Almal, & Chinnaiyan, 2009), and bioinformatics, such as biological modelling (Jacob & Burleigh, 2005). The other example areas are industrial process control (Castillo, Kordon, & Smits, 2006), time series prediction such as financial time series prediction (Santini & Tettamanzi, 2001), real-time crash prediction (Chengcheng Xu, Wang, & Liu, 2013) and meteorology prediction (Rodriguez-Vazquez, 2001).

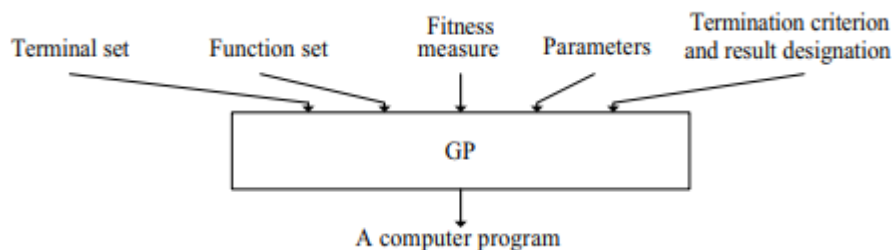
There are several widely used genetic programming tools and frameworks. In this study, the ECJ Evolutionary Computation Toolkit (Luke, 1998) which is written in

Java and is popular particularly with its genetic programming facilities is utilized. Moreover, EpochX (Otero, Castle, & Johnson, 2012) and JGAP (Chen, Chuang, & Tsai, 2001) are other open source genetic programming frameworks written in Java. Furthermore, TinyGP (Poli et al., 2008) is a symbolic regression centered genetic programming system which has implementations both in C and Java programming languages. Additionally, GPLAB is a genetic programming system for MATLAB (Silva, 2007). In addition, HeuristicLab framework, written in C# language, is utilized for genetic programming applications (Wagner & Affenzeller, 2002).

In this research, time series prediction is handled as a symbolic regression problem with genetic programming. Symbolic regression discovers the best suitable model of the space of mathematical expressions for a given dataset without establishing any inferences related to the structure of the model function (Poli, Langdon & McPhee, 2008). Symbolic regression is one of the most broadly examined applications of genetic programming (Bautu, Bautu, & Luchian, 2005; Hoai, McKay, Essam, & Chau, 2002; Shengwu & Weiwu, 2003).

The essential preparation steps for a genetic programming run are as noted below and in Figure 2.8:

1. Specifying the terminal set,
2. Specifying the function set,
3. Determining the fitness measure,
4. Determining the parameters to control the run,
5. Specifying the termination criterion of the run.



**Figure 2.8.** Fundamental steps of genetic programming (Koza et al., 2003)

The first two preliminary steps are utilized to determine the search space. The terminal set might contain variables and numerical constants. The function set might simply

contain the arithmetic functions such as multiplication, division, addition and subtraction, and conditional statements. At the third step, the main target of the search is defined by the fitness measure.

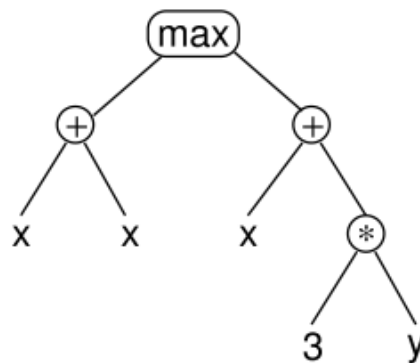
The fourth and fifth steps specify how the run is conducted. There are a number of control parameters in genetic programming. The example basic control parameters are the population size, the maximum number of generations, crossover, mutation and reproduction probabilities, the maximum depth of the program tree, the selection method of parents and the elitism (Koza, 1992). The termination criterion indicates when the run should be completed. For example, the number of generations can be specified as a control parameter and the run is terminated by reaching the maximum number of generations (Banzhaf, Koza, Ryan, Spector, & Jacob, 2000).

After completing the preliminary steps, the execution process is started by launching the genetic programming run. The execution steps are as indicated below:

1. Randomly generate an initial population of computer programs which consist of functions and terminals.
2. Iteratively produce a generation from the population until the termination criterion is fulfilled by following the sub-steps:
  - a. Implement each program in the population and assign its fitness utilizing the fitness measure of the problem.
  - b. Choose by allowing reselection of one or two individual program(s) from the population with a probability according to fitness.
  - c. Produce the new individual program(s) for the population by performing the next genetic operations with indicated probabilities on the selected individuals in step (2.b):
    - i. Reproduction: Replicate the chosen individual program to the new population.
    - ii. Crossover: Generate new offspring program(s) for the new population by recombining arbitrarily selected fragments from two chosen programs.

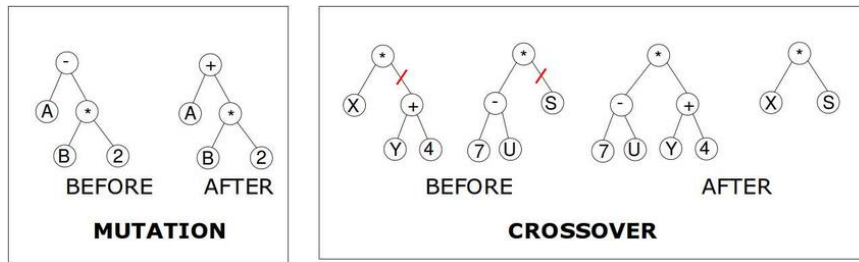
- iii. Mutation: Produce one new offspring program for the new population by randomly modifying an arbitrarily selected fragment of one chosen program.
  - iv. Architecture-altering operations: Choose an architecture-altering operation from the present genetic operations and produce one new offspring program for the new population by implementing the adopted architecture-altering operation to one chosen program.
3. After completing the termination criterion, pick the best program (the best-so-far individual) in the population generated during the run and assign as the result of the run.

The result might be a solution or an approximate solution to the problem if the run is successful (Banzhaf et al., 2000). In genetic programming, computer programs and the problem solution are generally declared as syntax trees (S-expressions) rather than code lines (Poli et al., 2008). Figure 2.9 shows the representation of the program  $\max(x + x, x + 3 * y)$ .



**Figure 2.9** GP syntax tree representation of  $\max(x + x, x + 3 * y)$  (Poli et al., 2008)

Examples of mutation and crossover operations made on the individuals are shown in Figure 2.10.



**Figure 2.10** Examples of genetic programming operations (Lane, Sozou, Addis, & Gobet, 2016)







## **CHAPTER 3**

### **PREVIOUS WORK**

#### **3.1. Genetic Programming**

Recently, Graff et al. (Graff et al., 2016) published a study that uses a similar methodology to this research. The study presents a comparison between genetic programming and traditional approaches for wind speed forecasting. Various types of time series from M1 and M3 competitions at different intervals such as a monthly, quarterly, and yearly were used for forecasting. Two different genetic programming systems were utilized. One is a steady-state system with tournament selection, and the other is enhanced with the resilient back-propagation (RPROP) method for the tree constants optimization. Various time series with different periods such as yearly, quarterly and hourly were used. Three function sets with different operators were utilized to the impact on the prediction quality.

The research focuses on the effect of using different function sets rather than tuning the traditional GP parameters. Tree simplification was conducted by RPROP method. This procedure was applied to forecast the wind speed using genetic programming. Symmetric mean absolute percentage error (sMAPE) was used for measuring the performance. The study reveals that particular configurations of genetic programming are competitive against the traditional forecasters and genetic programming performs better scientifically in a number of cases.

There is another study which focuses on a traffic flow data time series prediction by Xu, Li and Wang (C. Xu, Li, & Wang, 2016). This study presents a hybrid model using ARIMA and GP for short-term traffic volume forecasting. Several periods of traffic flow ranging between five, ten, and fifteen minutes were predicted in this study. The aim of combining models is improving the chance of obtaining the linear and nonlinear patterns, and better prediction performance.

The study reported two advantages of genetic programming. The first advantage is producing mathematical equations as solutions which can be used practically in engineering applications. The other advantage is the solution of genetic programming

takes precedence over the black-box solutions in artificial intelligence models. The study claimed that machine running time is the crucial disadvantage of genetic programming. However, the study also indicated that the calibrated models of genetic programming could reduce the required time.

Genetic programming was utilized to forecast the non-linear elements of the traffic flow time series in this study. Basic arithmetic operators, trigonometric functions such as sin and cos, and square operations were included in the function set. Reproduction operations were eliminated and crossover with the mutation is applied. The mean absolute error (MAE) was utilized for the fitness function. However, for a better comparison, three other error measurements were used as well, such as the mean relative error (MRE), the mean square error (MSE), and the mean square relative error (MSRE).

The results of the study showed that developing a hybrid model obtains better forecast performance than ARIMA models. Moreover, the results indicated that predictions of the hybrid model have a higher correlation with the observed values. Therefore, the hybrid approach can obtain the attributes of the time series data better. The study concluded that hybrid models might be utilized for online traffic management and control.

Vazquez and Escolar (Vázquez & Escolar, 2001) conducted another time series forecasting with genetic programming. The aim of the study is meteorological time series prediction by using genetic programming. The non-linear Auto-Regressive Moving Average with exogenous inputs (NARMAX) model was used as a base while implementing the genetic programming.

The residual variance metric and the long-term prediction error (LTPE) were utilized to measure the model performance. The time interval in data is fifteen minutes with thirty-seven days. The data columns are date, time, temperature, relative humidity, wind velocity, wind direction and ground radiation. The study focuses on forecasting the temperature. Without using reproduction, crossover and mutation operations were utilized. Six days were used to carry on the experiment. The first four days were accepted as a training set, and the remaining two days are utilized for forecasting.

The experiment results showed that the model representation is robust. Genetic programming was easily applicable to the different models such as the single-input-

single-output (SISO), the multiple-input single output (MISO) and its expansion to the multiple-input-multiple-output (MIMO). Moreover, the study revealed that the developed modelling problem approach with genetic programming is a viable settlement for the meteorological time series forecasting.

Furthermore, Graff, Pena and Medina (Graff et al., 2013) presented a study on wind speed forecasting with genetic programming. In this study, two different genetic programming systems were compared to the ARIMA for time series prediction. One genetic programming system used traditional fitness function and the other genetic programming system used modified fitness function. The input time series data contains one-year hourly wind speed measures. The data was split in two. In the first split, 291 days were evaluated as training and the next day was evaluated as a test. In the second split, 327 days were evaluated as training and the 328th day was evaluated as a test.

Two ARIMA models were constructed for different time horizons prediction. One-step to six-step forecasting were performed and repeated for covering all the data. Ten separate runs were conducted for each system. From the ten best individuals of runs, the one whose prediction was better in both one-step and multiple step forecasting was chosen. For the performance comparison, the symmetric mean absolute error was utilized.

The study results showed that genetic programming results were efficient for short-term time series forecasting against the traditional forecasting methods such as ARIMA. Moreover, it was reported that genetic programming with modified fitness function produced better results than the traditional fitness function usage. Therefore, modifications for the fitness function could be beneficial for more accurate forecasting.

### **3.2. ARIMA, ANN and FTS**

Rahman, Lee, Suhartono and Latif (Rahman et al., 2015) conducted the first relevant study about air quality forecasting. The study involves a time series forecasting model of monthly air quality index values by three different statistical models such as the Box–Jenkins approach of seasonal autoregressive integrated moving average (ARIMA), artificial neural network (ANN) and fuzzy time series (FTS). Data were collected from three different areas in Johor, Malaysia between 2000 and 2009. A performance comparison was made by utilizing the mean absolute percentage error

(MAPE), mean absolute error (MAE), mean square error (MSE), and root mean square error (RMSE).

The ultimate goal of the study was developing the accurate statistical forecasting models and using the models to observe the air quality status. The data is split into training and test sets. The period from 2000 to 2008 was accepted as the training set and the data from 2009 were accepted as the test set. Twelve examinations were conducted in total. While using the Box-Jenkins modelling approach, writers included seasonal components in the ARIMA model, and it is called as SARIMA model. Moreover, the artificial neural network was utilized as a multi-layer perceptron (MLP).

To find the best model statistically, root mean squared error is used as the determinant indicator. The study results revealed that the traditional ARIMA model surpassed the FTS model in two stations. These stations are located in urban areas. FTS model performance was better in the suburban area. In all three stations, artificial neural network method generated the most accurate predictions. In addition, writers reported that the basic form of ANN might be utilized for further prediction of pollutant gases because ANN is convenient for estimating the changing series with seasonal and trend characteristics, especially air quality time series.

There is another study conducted by Peng et al. (Peng et al., 2017), which was based on the air pollutant concentrations forecasting which includes machine learning and an artificial neural network together. The study utilized a nonlinear machine learning algorithm by the randomized neural network, which is the extreme learning machine (ELM). The forecast data contains hourly spot concentrations up to forty-eight hours for six stations across Canada. The main concern of the study is the updatable forecast models in terms of real-time forecasting. To achieve this, the authors reported the need for developing nonlinear updatable models for real-time prediction. The study focuses on a proficient non-linear machine learning method for air quality forecasting. The method was easy to be updated when new data is available.

The data was split as two years for training and three years for testing the model. Four air quality prediction models were examined and the climatology is utilized for performance comparisons. The models were the multiple linear regression (MLR), Online Sequential Multiple Linear Regression (OSMLR), Multi-layer Perceptron Neural Network (MLPNN), and Online Sequential Extreme Learning Machine

(OSELM). Mean absolute error (MAE), coefficient correlation and mean error measure metrics were used as forecast statistics. The results showed the capabilities of the updatable non-linear machine learning methods. The study revealed that these models might improve air quality predictions significantly. Moreover, it was reported that using the non-linear modelling and reasonable model updating together would be advantageous.

### **3.3. Machine Learning Algorithms Used in This Study**

Random Forest, Random Tree and Rep Tree algorithms are utilized to compare genetic programming performance in this study.

Random Forest is an ensemble machine learning algorithm. It is a combination of the Bagging algorithm and the random subspace method, which utilizes multiple randomized decision trees as the base classifier (Sammut & Webb, 2017). It performs classification and regression tasks (Breiman, 2001). Random Forests is a supervised learning approach and operates in accordance with the easy yet practical “divide and conquer” technique which samples portions of the data, builds a randomized tree predictor on each small sample and gathers the predictors together (Biau & Scornet, 2015).

Random Tree is a supervised learning method. It is an ensemble learning algorithm which builds diverse separate learners. To construct a random set of data for building a decision tree, it uses a bagging idea. By utilizing the best split of all variables, each tree node is split. On the other hand, each tree node is split by utilizing the best among the subgroup of estimators randomly selected at that node in a random forest (Kalmegh, 2015). Random Tree performs no pruning and selects a test according to a certain number of arbitrary features at each node. To build random forests, Random Forest uses bagging technique on random trees ensembles (Witten, Frank, & Hall, 2011).

Rep (Reduced Error Pruning) Tree creates a decision or regression tree utilizing information gain or variance reduction and prunes it by reduced-error pruning (Witten et al., 2011). Rep Tree is a fast decision tree learner and sorts values for numeric attributes one time only. Missing values are handled by splitting the corresponding instances into fragments (Dhakate, Patil, Rajeswari, & Abin, 2014). Rep Tree builds multiple trees in various iterations with the regression tree logic and chooses the best

one from all constructed trees. In pruning phase, the mean square error measurement is utilized on the estimations performed by the tree (Jayanthi & Sasikala, 2013).

### **3.4. Air Quality Modeling Systems**

Next, there was another study conducted by Zhou et al. (Zhou et al., 2017) related to weather and forecast/chemistry model. In this study, comprehensive assessment of The Regional Atmospheric Environmental Modeling System for eastern China (RAEMS) was conducted. The fully online coupled weather research and forecasting/chemistry (WRF-Chem) model was established. The system generated daily predictions upto seventy-two-hours for a two years' period (2014-2015). The authors reported that online systems might be advantageous in terms of analyzing the interactive relations between meteorology and chemistry. RAEMS is an online system and it perpetually supplies meteorological forecasts via SMS. Moreover, it was indicated that RAEMS is proficient at non-permanent fluctuation and spatial dispersion of vital pollutant gases over eastern China.

The study produced forecasts with different lengths such as twenty-four, forty-eight and seventy-two hours. Prediction evaluations were done for a three-day output approach to distinguish performances between different time lengths. Average pollutant concentrations of 131 cities are forecasted during the study. Various error measure metrics were utilized for performance comparison, such as root mean squared error and correlation coefficient. It is shown that the overall results of the study on meteorological values, such as temperature, relative humidity and precipitation, were forecasted accurately. On the other hand, writers indicated that the model overestimates wind speed. However, the results were conventionally satisfactory for air quality modelling.

Elbir et al. (Elbir, Kara, Bayram, Altioik, & Dumanoglu, 2011) presented a study of traffic-related air pollution forecasting with an operational street pollution model (OSPM). The particulate matter (PM<sub>10</sub>) concentrations in five different street canyons in Izmir, Turkey was predicted. Some of the streets showed symmetric canyon characteristics and others showed asymmetric canyon characteristics.

Writers reported that traffic was one of the crucial air pollution cause in the city. In this study, different street canyon models constructed to observe the effects of traffic emissions in street canyons. Therefore, contributions from dissimilar sources were

accepted as the background pollution. The traffic volume information was categorized as motorcycle, passenger cars, vans, and trucks and buses. In addition to the traffic-related parameters, meteorological parameters were collected as well. One week long hourly measurements were utilized as data. Statistical investigations conducted by utilizing the correlation coefficient of determination ( $R^2$ ), the fractional bias (FB), and the index of agreement (IA) metrics.

Results showed that at the symmetric canyons, the model performed statistically good forecasts. The study showed that for the symmetric canyons, the predictions of non-permanent changes of the model were accurate. On the other hand, results concluded that the model did not perform well for the asymmetric street canyons. Thus, the writers reported that the OSPM model might be utilized as a forecasting tool in terms of the streets' traffic-related air pollution detection.





## **CHAPTER 4**

### **IMPLEMENTATION**

#### **4.1. Data Preparation**

To begin with, Environmental Engineering Department of Dokuz Eylül University delivered the air quality data. It contains hourly meteorological and pollutant gas measures for Çanakkale in 2013. Every data row has the month, day and hour columns. The date starts from January 1 and ends in 31th of December. The data pre-processing was required because there were empty cells in the hour and observed gas columns. First, the missing hour values were filled in the initial data. Then, the pollutant gas attribute rows with empty cells were discarded separately.

As the next step, data were examined for the redundant columns. Since the hourly measures are more meaningful in terms of the air quality prediction, year and month attributes were removed. Instead of the excluded attributes, weekday column is included. In addition, instead of using the initial values of SO<sub>2</sub> and PM gas measures, using the temporal fluctuations is more meaningful to forecast. Thus, differences between rows were calculated by subtracting previous cell values from the initial cell values and utilized as observed gas measures. Eventually, there are nine attributes in the data. These attributes are the hour, weekday, wind speed, humidity, air pressure, temperature, wind direction, SO<sub>2</sub> and PM measures.

After the data cleansing process, normalization was required to be implemented. Since the scale of the attributes varies, it is difficult to correlate the relationship between the attributes. To achieve this, a common scale for all the attributes was provided by normalization. Before normalization, non-numeric attribute values were converted to numeric values. For a better application, categorical and date attributes needed to be revised. For example, wind direction attribute has the compass direction values such as 'N', 'NNW', and 'SE'. Wind direction and date attributes were impractical to be utilized as numerical values. Therefore, they were transformed from nominal values to numerical values by disjunctive coding technique.

To apply disjunctive coding method, new columns were added up to the number of categories in each attribute. For example, wind direction has sixteen categories. Therefore, instead of one wind direction attribute, sixteen categories were utilized to generate wind direction columns. For instance, if a row's wind direction value is 'S', S column's value was set to one and other wind direction columns' values were set to zero. The same procedure was applied to date attributes as well. Although date attribute values were numerical, they were trivial to be utilized as numerical. Therefore, hour and weekday attributes were discarded and new columns were included up to twenty-four for hour attribute and seven for weekday attribute.

After all the attributes were transformed to numerical values, the standard deviation and mean values were calculated. T-statistic normalization was applied to all attributes. The general t-statistic formula is:

$$g(x, X) = \frac{x - \bar{X}}{s} \quad (7)$$

where  $x$  indicates observed values,  $s$  indicates the standard deviation of  $x$ , and  $\bar{X}$  indicates the mean of  $x$ .

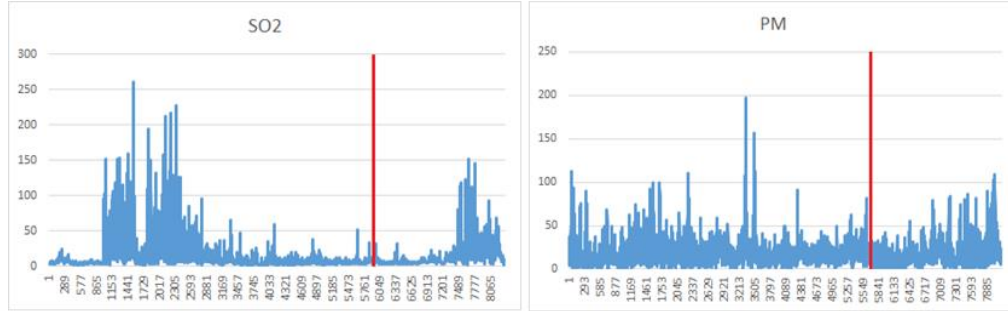
The pre-processing operation steps were applied separately for both SO<sub>2</sub> and PM values. The two new datasets were utilized as input for forecasting individually. Before completing the data pre-processing, the simple moving average values of both observed pollutants were calculated to improve prediction by highlighting the hourly patterns in data. To achieve this, the average of the summation of each value of SO<sub>2</sub> and PM columns with the previous values was calculated. The computed values were included in the dataset as the moving average attribute. The simple moving average formula is:

$$x_{SMA} = \frac{1}{n} \sum_{i=0}^{n-1} x_{p-i} \quad (8)$$

Where  $n$  is the number of previous values and  $x_p$  indicates observed values.

Next, train and test set split percentages were identified by inspecting both data sets. According to the trend analysis, 70% of data was defined as a training set and 30% of data was defined as a test set. Since the data is a time series data, the splitting was

applied sequentially instead of doing it randomly. Training and test set split points for SO<sub>2</sub> and PM values are shown in Figure 4.1. Training and test sets were converted to the attribute-relation file format (ARFF) by using Weka. Thus, the data pre-processing was completed.



**Figure 4.1.** Train and test sets split points

## 4.2. Defining the Problem as a Symbolic Regression Problem

After preparing the datasets, the air quality forecasting was defined as a Symbolic Regression problem in ECJ framework (version 25). Firstly, except for the observed gas attributes, all attributes in the dataset were defined as Java classes individually as required by the framework. These attributes were used as problem inputs. Therefore, each of them had an index value, which was matched to the dataset order. By this way, when the genetic programming tree was bred, the program understands which input attribute corresponds to which Java class.

Moreover, each attribute class was utilized as a terminal tree node. Each attribute class includes a common function, which returns the number of expected children by the node attribute. Since these attributes are terminal nodes, the number of expected children for each of them is zero. In addition, an Ephemeral Random Constant (ERC) class was used a terminal node. In this research, ECJ's RegERC Java class was utilized to generate random constants. It has an interval between -1.0 and 1.0.

On the other hand, there are other node classes, which require expected children. These are the non-terminal tree nodes. Some arithmetic and the logical operations were defined as non-terminal nodes. Arithmetic operations are addition, subtraction, multiplication, division, square root and square functions. Logical operations are if-then-else, equal, greater than, greater than equal, less than and less than equal functions.

Square root and square functions require one excepted children. The if-then-else function needs three excepted children. The first child is utilized as the condition, the second child is used as the true consequent, and the third child is used as the false consequent.

Logical operations have a special case. As it was indicated before, there are categorical attributes values in the dataset. These attributes were useless to be utilized in arithmetical operations. Therefore, logical operations became a necessity. Since the categorical attributes were normalized by using one and zeros with the disjunctive coding technique, logical operations implementation differed from other operations. Except for if-then-else operation, other logical operations compare two children. Since the first child of if-then-else operation is a condition, its value must be true or false. To achieve this, the other logical operations own a common Boolean variable. This was provided by an interface, which obligates overriding a function that returns a Boolean variable. The comparison result of two children nodes was assigned to the Boolean variable. If-then-else operation calls the function and utilizes its return value as a condition. The structure of the if-then-else operation is shown in Figure 4.2.



**Figure 4.2.** If-then-else operation

In addition, all the node Java classes are obligated to own a name attribute. The name variable was assigned and returned with a function. For instance, the addition operation node name was defined as '+', or greater than equal operation node name was defined as '>=', etc.

### **4.3. Methodology of Training and Evaluating Prediction Performance**

The problem was defined in a GPPProblem Java class. Training and test sets were read from files in this class. First, the training set was read line by line. For every single line, input attributes were read and put into an array. At the end of every line, the array was added to an array list, which was stored as training input set. After reading input

attributes, output attribute was read and the same process was repeated for the training set. The same steps were applied to test set as well. Therefore, there were four array lists, which were training input set, training output set, test input set and test output set. Training sets were utilized in problem evaluation and test sets were utilized in post-evaluation step. Populating array lists was executed in the setup function.

The problem's individual evaluation was executed in evaluate function. First, pre-processing was applied to tree nodes recursively. It started from the deepest level children and ended at the topmost level children.

One of the main problems in genetic programming is bloat that is the trees grow too large. A preprocessing step was added, in order to eliminate useless nodes. For the non-terminal operations addition and subtraction, children whose values were zero were checked and the sub-trees were eliminated from the tree. To achieve this, the sub-tree child node was assigned to the other non-zero value child node. For instance, if there was a summation parent node, and children nodes' values were zero and an arbitrary double value, then summation node and its children nodes were eliminated from the tree. Instead, the child node of the parent node of summation node was assigned to the non-zero node, which was the other child node of the summation node.

Moreover, child nodes with zero and one values were checked for division and multiplication operations as well. If a child node was multiplied or divided by one, the same process was applied by the previous control operation. However, if zero, or child with value zero node was the dividend in a division sub-tree multiplies a child node, then the same process was applied but the new child node was assigned to zero value node.

For the if-then-else node, the consequent children nodes were checked in the tree pre-processing. If both of the nodes' values were the same, then the same operation was conducted with the arithmetical operations and the first consequent child node was assigned as the new child node to the parent node of the if-then-else node.

After the tree pre-processing operation, training input set instances were utilized one by one in a loop. In every step, the instance of input attributes was assigned to an array. As it was indicated before, ECJ matches the attributes with the corresponding Java classes, which has the same index information. For example, if the wind speed attribute was the first attribute of the input instances, then the wind speed attribute was the first

element in the instance array and it was matched with the wind direction Java class, whose index was zero as well.

Next, in the same loop, training output set's value at the same index was assigned to a variable, which was an observed pollutant value. The individual's tree was evaluated and produced a value. The value was used as the forecasted value. The square of the differences of the observed and predicted values was summed in the loop. After the loop ends, the summation was divided by the training set size. Eventually, the square root function was applied to the final value and fitness of the individual was calculated. The training set fitness was computed by using the root mean square error (RMSE) measure. The KozaFitness Java class was utilized as the individuals' fitness setup.

Moreover, if-then-else nodes were checked recursively at the tree post-processing as well. To achieve this, two integer variables were added to if-then-else node class as true and false counts. The count values were increased according to the condition node's true or false value. At the post-processing, these counts were compared to the data size for eliminating tautology and fallacy. If the true or false count was matched with the data size, the parent node's child node was changed by the if node's true or false child node.

As the final step of individual evaluation, the individual was checked as evaluated. Therefore, if the individual does not change, the next time individual evaluation can be skipped. Eventually, individual evaluation step was completed.

After evaluation process on the training data, post-processing was applied to the test data by using the best individual from the training evaluation. This process was conducted in the describe function. The same steps with the evaluation process were applied. Since there was only one individual (the best performing individual), post evaluation was applied only once. The test set was utilized in a loop likewise. For the fitness evaluation, five error metrics were calculated. These were the root mean square error, mean absolute error, correlation coefficient, relative absolute error and root relative squared error measures. After computations, results were written to both stat files and separate text files. Thus, the symbolic regression problem evaluation was completed.

#### 4.4. Parameter Setting

Next step was configuring the genetic programming settings. In ECJ, settings are kept in parameter files. As it was mentioned before, ECJ links the attribute Java classes to the input file instance attributes in the tree evaluation process. To achieve this, function set was defined in a parameter file. The function set must have a size and each function needs a constraint. All the terminal and non-terminal node functions were added to a function set. Each had varied constraints. For instance, categorical attributes like wind direction attribute or weekday attribute had a Boolean constraint. On the other hand, an if-then-else operation had a different constraint, who had three children and its children had different types. The parameter file used for setting the terminal and non-terminal nodes is given in Appendix 1.

After defining the function set and function constraints, the population size and the number of generations were configured. The symbolic regression problem Java class was also assigned to the related genetic programming problem parameter. Moreover, since the double data type was used in tree evaluations and node constraint types, a Java class was created and utilized as the problem data. All the non-terminal and terminal node attributes and functions used the same data type.

Next, the common genetic programming parameters were defined based on the default Koza parameters in ECJ. Desired parameters were modified. For example, in Koza settings, mutation is not used in breeding operations. While conducting experiments, mutation pipeline was utilized and its probability was configured as well. The crossover probability, elitism, tournament selection size, the tree builder, the minimum and the maximum tree depth, and the number of jobs parameter settings were modified with respect to the experiment configurations.

In addition, stat files were modified as well. At the end of each run, one stat file was created. Instead of printing all the generations' individual details, the best individual details such as the training data fitness, test data error measurements and the tree model were written to the stat files. The tree model output style was defined as the graph description language format (DOT). The definition was made by the print style parameter. The output tree models were examined by using the Graphviz (Ellson et al., 2002) program.



Moreover, the terminal output details were modified as well. Since the necessary details were written to the stat files, the used parameters were written to the console for parameter usage controls. These modifications were configured in the parameter files by setting the related parameters.

After the program setup was completed, experiments were conducted. First, default Koza parameter configuration was utilized, and the program was run based on the Koza parameters. The results were compared to the Weka classifier tree algorithms such as Random Forest, Random Tree and Rep Tree algorithms. In Weka, the training data was defined as a training set and test data was assigned as a test set. The training and the test data were the same in both ECJ and Weka. It was observed that tree algorithms' results were better. Therefore, parameters were modified and the comparison was conducted again. At first, every run for each change was conducted by the same seed values. When the improvement was detected, multiple runs were conducted by using the random seeds and comparisons were made according to the runs' results. The results again showed that ECJ results were still worse. To bring a different approach, windowing method was tried.

The previous instance or more than one previous instances' observed output values were utilized as input for the instance in a manner similar to the moving average method. Instead of average, the exact values were used. The window size was broadened up to five. After observing each of the windowing results, it turned out that the error measures were needed to improve. To achieve this, the full factorial design approach was used. After investigating each of the changes of the results based on the parameter modifications, certain parameter effects were observed and included in the experiment combinations. Three experiments were conducted. The full-factorial experimental design was generated in the R Project (R Core Team, 2017). Since the correlation coefficient and the root mean squared error are the commonly used error metrics, these errors metrics of experiments results were investigated and utilized to determine the best parameter combinations.

In the first experiment, population size, elite count, tournament size, crossover rate and the maximum depth of tree parameters were changed. It was not known that which parameters were effective, so it was not known how effective these were. Therefore, the first experiment was designed in this way. The number of generations and the tree builder parameters remained the same. 1215 runs with 750 generations for each run



were conducted. The tree builder was set to Full Builder. The crossover and reproduction rates changed according to the experiment result. For example, if the crossover rate was set to 0.70, the reproduction rate was set to 0.30 respectively. Elite count experiment was set as the percentage of the given population. For instance, if the population size was two hundred and fifty and the elite count rate is 0.1, then the elite count was calculated as twenty-five. Mutation operation was not used in the first experiment. The seed parameter varied by the experiment design. The number of breed threads and the evaluation threads were set to four. Each experiment was repeated for five times. The parameter tuning was applied as:

- Generations = 750 (constant)
- Tree Builder = Full Builder (constant)
- Population Size: {250, 500, 750}
- Tournament Size: {2, 5, 7}
- Elite Count Percentage: {0.01, 0.05, 0.1}
- Crossover Rate: {0.7, 0.8, 0.9}
- Tree Size: {5, 6, 7}
- The number of jobs for each experiment = 5
- $3^5 \times 5 = 1215$  runs

The execution order determined by the Design of Experiment (DoE) plugin for R is shown in Figure 4.3.

name	run.no.in.std.order	run.no	run.no.std.rp	popSize	tournamentSize	eliteCount	crossoverProbability	Blocks
1	64	1	64.1	250	2	0.05	0.9	0.1
2	22	2	22.1	250	5	0.1	0.7	0.1
3	55	3	55.1	250	2	0.01	0.9	0.1
4	56	4	56.1	500	2	0.01	0.9	0.1
5	51	5	51.1	750	5	0.1	0.8	0.1
6	78	6	78.1	750	5	0.1	0.9	0.1
7	30	7	30.1	750	2	0.01	0.8	0.1
8	45	8	45.1	750	7	0.05	0.8	0.1
9	58	9	58.1	250	5	0.01	0.9	0.1
10	42	10	42.1	750	5	0.05	0.8	0.1

**Figure 4.3.** The first experimental design generated by R DoE plugin

The first experimental results were analyzed in R Project by considering both correlation coefficient and root mean squared error measures metrics. The interval

plots of significance for effects are included in Appendix 2. The bigger values of the population size, the tree size and the elite count percentage gave better results. The population size was set to one thousand. Since bigger values of the tournament size and crossover rates did not improve the results, smaller values were selected in order to save computation time. The tournament size was set to two. The maximum tree size was selected as seven based on the reasonable tree models. It was observed that when the tree size was bigger, the generated tree models were more comprehensible. It was not planned to increase the maximum tree size more than seven because this time the tree models became too complex to obtain a reasonable prediction. Therefore, the second experimental design was made with parameters, which created an alteration respectively.

In the second experiment, the mutation was included as well. The probability rate was assigned as the same as the first experiment. If the experiment indicated the mutation usage, the second pipeline was set to mutation pipeline instead of reproduction pipeline. In addition, the number of generations was increased to two-thousand and the population size was set to one-thousand. Since the population size was not changed, the percentage used for the elite count was eliminated. The maximum depth of the tree parameter was assigned as seven and the tournament size was set to two. Each experiment was conducted for five times. The parameter tuning was applied as:

- Generations = 2000 (constant)
- Tree Builder = Full Builder (constant)
- Population Size = 1000 (constant)
- Tree Size = 7 (constant)
- Tournament Size = 2 (constant)
- Elite Count: {100, 150, 200}
- Crossover Rate: {0.25, 0.5, 0.75}
- Mutation Usage: {T, F} (indicates the usage of mutation or reproduction pipeline)
- The number of jobs for each experiment = 5
- $3^2 \times 2 \times 5 = 90$  runs

The execution order determined by the Design of Experiment (DoE) plugin for R is shown in Figure 4.3.

name	run.no.in.std.order	run.no	run.no.std.rp	EliteCount	CrossoverRate	useMutation	Blocks
1	15	1	15.1	200	0.5	FALSE	0.1
2	13	2	13.1	100	0.5	FALSE	0.1
3	18	3	18.1	200	0.75	FALSE	0.1
4	10	4	10.1	100	0.25	FALSE	0.1
5	16	5	16.1	100	0.75	FALSE	0.1
6	8	6	8.1	150	0.75	TRUE	0.1
7	7	7	7.1	100	0.75	TRUE	0.1
8	5	8	5.1	150	0.5	TRUE	0.1
9	4	9	4.1	100	0.5	TRUE	0.1
10	17	10	17.1	150	0.75	FALSE	0.1

**Figure 4.4.** The second experimental design generated by R DoE plugin

In the second experiment, a parameter file was overridden for mutation usage to override the pipeline source at run-time. Hence, the parameter file was modified based on the experiment parameters. If the experiment indicated mutation usage, the parameter file was overridden to set the second pipeline source to mutation pipeline. The experiment results again analyzed with R. The better results' parameter variables were selected as the final experiment. It was observed that final experiment parameters for SO<sub>2</sub> and PM datasets are varied. Therefore, two different parameter tunings were applied to data sets.

#### 4.5. Performance Evaluation of the Genetic Programming Model

In the final experiment, seed values were set randomly by using time variable in ECJ. The number of jobs parameter was assigned to twenty-five. The population size, tournament size, tree builder and the number of generations remained the same. Elitism and pipeline probabilities were changed. The parameter values selected for SO<sub>2</sub> forecasting were:

- Generations = 2000 (constant)
- Tree Builder = Full Builder (constant)
- Population Size = 1000 (constant)
- Tree Size = 7 (constant)
- Tournament Size = 2 (constant)
- Elite Count: 100

- Crossover Rate: 0.25
- Mutation Usage: false
- The number of jobs = 25
- Seed = time

The parameter values selected for PM forecasting were:

- Generations = 2000 (constant)
- Tree Builder = Full Builder (constant)
- Population Size = 1000 (constant)
- Tree Size = 7 (constant)
- Tournament Size = 2 (constant)
- Elite Count: 200
- Crossover Rate: 0.25
- Mutation Usage: false
- The number of jobs = 25
- Seed = time

The detailed results obtained by the developed GP and the Weka algorithms are given in Appendix 3. The resulting error measures obtained in the final experiment were compared to the Weka tree algorithms' results. It was noticed that the average values of the five error measures were better than the Weka algorithms' average error measures. Moreover, the statistical significance testing (t-test) approach was applied to the final results as well. Each of the Weka algorithms' averages of 25 runs was compared to the genetic programming regarding the t-test. The significance level for the t-test was set to 0.05. The results of the t-test are shown in Table 4.1. The test was conducted for all the error measure metrics such as the correlation coefficient, mean absolute error, root mean squared error, relative absolute error and root relative squared error. The significance test scores showed that the genetic programming was better than Weka classifier tree algorithms at forecasting for both SO<sub>2</sub> and PM gas measures in every metric.

**Table 4.1.** t-test comparisons over averages of 25 runs of GP and decision tree algorithms

SO <sub>2</sub>					
	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
GP vs Random Tree	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>
GP vs Random Forest	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>
GP vs Rep Tree	0.3644x10 <sup>-8</sup>	0.0x10 <sup>-12</sup>	0.1704x10 <sup>-8</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>
PM					
	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
GP vs Random Tree	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>
GP vs Random Forest	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>
GP vs Rep Tree	0.18022x10 <sup>-7</sup>	0.29x10 <sup>-10</sup>	0.7249x10 <sup>-8</sup>	0.0x10 <sup>-12</sup>	0.0x10 <sup>-12</sup>

The Random Tree algorithm produced the worst results for both of the gas measures. While the correlation coefficient result for PM was better than SO<sub>2</sub>, the remaining error results were worse than SO<sub>2</sub>. The Random Forest tree algorithm produced better results in every error metrics than the Random Tree algorithm for both of the gas measures. While the correlation coefficient, relative absolute error and root relative squared error results were better for PM, mean absolute error and root mean square error measures were worse than SO<sub>2</sub> error measures. The Rep Tree algorithm was the best of all tree algorithms for each error metrics. It was observed that the coefficient correlation,

relative absolute error and root relative squared error results were better for SO<sub>2</sub> pollutant forecasting. The other error measures were better for PM. In addition to these results, the genetic programming produced significantly better results than the other algorithms. The results showed that for each error metrics, SO<sub>2</sub> pollutant prediction was slightly better than PM pollutant prediction.

As the final comparison, genetic programming's performance was compared to an ARIMA model using R Project. R Project (R Core Team, 2017) is an open source framework for statistical computing. The forecast package in R Project supports automatically generated optimal ARIMA models (Dalinina, 2017). The sample implementation code was included in Appendix 4. While constructing the ARIMA model, the time series data was utilized as it is. Only the pollutant gas values were converted to their normalized forms. In addition, moving average values were added to the model. The seasonal component of the time series was calculated by the stl function. The frequency parameter utilized for constructing a time series object was set daily since the data consists of the hourly measures. Then, seasonal adjustment function was used to eliminate the seasonality from the time series.

Next, since the time series were required to be stationary to use the ARIMA model, the augmented Dickey-Fuller (ADF) test was applied to the data. Therefore, it was confirmed that the air quality time series was stationary. Next, the data was split into a training set and a test set. The data was divided into a train set and test set similar to the genetic programming and Weka tree algorithm experiments. The train set was utilized for constructing the ARIMA model. The test set was used for forecasting. The model was constructed based on the training set and the forecasting was made based on the test set.

The different point of the experiment was that with ARIMA, the data was handled as a time series. On the other hand, with genetic programming, the numerical prediction as a regression on the time series data was conducted. The experiment results differed for SO<sub>2</sub> and PM gases. The mean absolute error and the root mean squared error measure metrics were analyzed for performance comparison. It was observed that ARIMA model performed better at the SO<sub>2</sub> gas prediction for both of the error measure metrics. On the other hand, the genetic programming model produced better result of SO<sub>2</sub> forecasting than ARIMA model for mean absolute error measure metric.

**Table 4.2.** ARIMA forecast results

PM	RMSE	MAE		SO <sub>2</sub>	RMSE	MAE
ARIMA						
Training set	0.2386032	0.1591638		Training set	0.4971606	0.2060416
Test set	0.6214668	0.410306		Test set	0.3912609	0.1772004
GP						
Test set (average)	0.764172898	0.486630375		Test set (average)	0.44337426	0.167358973
Test set (best)	0.760064331	0.481893545		Test set (best)	0.435097259	0.163717903





## **CHAPTER 5**

### **CONCLUSIONS AND FUTURE RESEARCH**

In this study, air quality time series prediction is conducted by genetic programming approach. The forecasting problem is defined as a symbolic regression problem using the ECJ framework. One year-long air quality metric measurements per hour in Çanakkale region are utilized as input data, and SO<sub>2</sub> and PM gas measures are used as the observed data. The aim of this research is constructing a numerical prediction model by using genetic programming on the given dataset. The dataset is evaluated as a time series. Moreover, normalization, windowing and moving average calculations are utilized in order to obtain better forecasting results.

Different genetic programming parameter combinations are tested to determine the best combination of parameters. The default Koza parameter settings in ECJ is used as an initial step. To understand the success of the problem fitness, Weka classifier tree algorithms are run on the same dataset to make a comparison. These algorithms are Rep Tree, Random Forest and Random tree algorithms. Train set and test set division is applied to the data. Test set error measure metrics are utilized for comparison such as the correlation coefficient and root mean squared error. It is observed that Koza parameter settings produce worse results than tree algorithms. Thus, a full factorial design approach is utilized.

In the first design of experiment test, different values of genetic programming parameters such as crossover probability, tournament selection size and population size are considered. The first results show us that some parameters with certain values can provide better results. Therefore, the second experiment is carried out by focusing on the prominent parameters. Constructed tree models are taken into consideration as well during the decision process for the next experiments' parameter combinations. The success of the solution does not only depend on the problem fitness but also on generated genetic programming tree models. The second experiment results' analysis revealed that SO<sub>2</sub> and PM gas prediction success varies with different combinations.

Hence, the concluding experiment has two different parameter settings for SO<sub>2</sub> and PM gas forecasting separately.

The outcome of the experiments revealed that with certain parameter combinations, the genetic programming is more successful than other classifier tree algorithms for both SO<sub>2</sub> and PM gas prediction. The comparative analysis with the other algorithms is conducted by using statistical tests. The pair-wise t-test comparisons for each of five error measure metrics prove that genetic programming achieves better results. The significance of the comparison outcomes is that the genetic programming performed even better than the Random Forest, which is one of the most frequently practiced ensemble learning methods. Random Forest uses more than one learning algorithm to achieve better predictive performance and generally produces a very good result.

Additionally, to observe the level of performance of the genetic programming in comparison to the traditional time series prediction approaches, an ARIMA model is developed and tested as well. The prediction results of the ARIMA model showed that ARIMA model performed better than genetic programming.

As a conclusion, it can be suggested that there are several contributions of this study. First, it is proven that genetic programming holds out a successful approach to the time series forecasting. The capabilities of genetic programming might be enhanced further with additional research. Moreover, the previous research is limitedly carried out on air quality time series forecasting with genetic programming. Most of the previous studies are based on machine learning, the traditional methodologies such as ARIMA models, or artificial neural network techniques. Genetic programming might get ahead not only with better results but provide a solution model as well. In this research, the tree models produced by genetic programming might be advantageous in many ways. For example, the models may serve as an inference regarding air quality time series analysis. Mostly used attributes in the generated models may be investigated further in future studies based on the same dataset. It may reveal hidden characteristics of the time series data, which may be unprecedented.

There are a number of aspects, which might be accepted as future work. Firstly, the tree models produced by genetic programming may be enhanced. In this study, tree breeding algorithms in ECJ are used without any modifications. The modification is applied after the tree breeding process is completed. The changes mainly focus on

making logical and arithmetic operations more sensible. For example, division by one or multiplication by one or zero, etc. In future studies, breeding operations might be considered to increase the quality of the generated tree models.

Secondly, the runtime of genetic programming might be reduced in prospecting studies. The biggest disadvantage of forecasting with genetic programming is the length of runtime. The runtime is longer than other algorithms' runtime length. However, the disadvantage might be resolved by using a more powerful CPU. If the genetic programming is run with a more powerful CPU, the runtime may be shortened.

Next, the factorial design approach may be enhanced by making inferences from the tested experiments in this research. A considerable amount of parameter combinations with variant values are experimented with and analyzed by R Project. Without trying the same combinations, new experiments might be designed by including unattempted combinations. Moreover, according to the outcome of this study, tested combinations may be extended further. Therefore, the time spent on experiments would be utilized for new trials. Moreover, the chance of obtaining better tree model solutions with better fitness might be increased.

Another improvable aspect of future work is the moving average method. In this research, the moving average computation is utilized by including only the previous value before the observed value. The moving average context might be extended to two or more previous values. This approach may yield better results in further studies. Moreover, there are other moving average calculation methods. For example, the simple moving average calculation is utilized in this study. In prospective studies, the other calculations such as the cumulative moving average or the weighted moving average techniques might be utilized as well.

The final aspect is implementing the genetic programming on the other problems of air quality time series analysis. As it was indicated before, time series forecasting with genetic programming is not widely used. Moreover, previous studies related to the air quality forecasting with genetic programming are even fewer. Thus, this research serves as a beneficial model of air quality forecasting by genetic programming. By investigating the methods used in this research, different applications of air quality time series utilizing the same or different genetic programming frameworks might be

used. For example, a traffic-related air quality dataset might be modelled as time series and predicted by following the similar steps and techniques.

As a conclusion, the number of studies based on air quality time series forecasting with genetic programming should be increased. Most of the previous studies focused on the artificial neural network or traditional methodologies such as ARIMA models for time series forecasting. Moreover, the studies, which used genetic programming for time series prediction are seldom, none of them investigated air quality time series. As a result of this study, genetic programming has proven to be statistically successful in air quality time series forecasting. The result obtained from this study can be used successfully to predict the air quality using the generated tree models. Air quality, which threatens human health in a dangerous way, is one of the most important problems that preventive precautions must be taken against and this study has contributed to the solution of this problem with the achieved results.

## REFERENCES

- Ahalpara, D. P. (2010). Improved forecasting of time series data of real system using genetic programming. In *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation* (pp. 977–978). <https://doi.org/doi:10.1145/1830483.1830658>
- Aladag, C. H., & Egrioglu, E. (2012). Advanced Time Series Forecasting Methods. In *Advances in Time Series Forecasting* (Vol. 2100, pp. 3–10). <https://doi.org/10.1007/978-3-319-31450-1>
- Ali Ghorbani, M., Khatibi, R., Aytak, A., Makarynsky, O., & Shiri, J. (2010). Sea water level forecasting using genetic programming and comparing the performance with Artificial Neural Networks. *Computers and Geosciences*, 36(5), 620–627. <https://doi.org/10.1016/j.cageo.2009.09.014>
- Austin, M. P., Bates, G., Dempster, M. A. H., Leemans, V., & Williams, S. N. (2004). Adaptive systems for foreign exchange trading. *Quantitative Finance*, 4(4). <https://doi.org/10.1080/14697680400008593>
- Banzhaf, W., Koza, J. R., Ryan, C., Spector, L., & Jacob, C. (2000). Genetic programming. *IEEE Intelligent Systems*, 15(3), 74–84. <https://doi.org/10.1109/5254.846288>
- Bar-Joseph, Z. (2004). Analyzing time series gene expression data. *Bioinformatics (Oxford, England)*, 20(16), 2493–503. <https://doi.org/10.1093/bioinformatics/bth283>
- Bautu, E., Bautu, A., & Luchian, H. (2005). Symbolic Regression on Noisy Data with Genetic and Gene Expression Programming. In *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)* (pp. 321–324). <https://doi.org/doi:10.1109/SYNASC.2005.70>
- Biau, G., & Scornet, E. (2015). A Random Forest Guided Tour. *ArXiv, submitted*, 173–184. <https://doi.org/10.1007/s11749-016-0481-7>
- Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., & Scuse, D. (2013). WEKA Manual for Version 3-7-8, 1–327. Retrieved from <papers3://publication/uuid/24E005A2-AA1B-4614-BAF5-4D92C4F37413>
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- Brockwell, P. J., & Davis, R. a. (2002). *Introduction to Time Series and Forecasting , Second Edition*. <https://doi.org/10.1007/b97391>
- Büke, T., & Köne, A. (2016). Assessing Air Quality in Turkey: A Proposed, Air Quality Index. *Sustainability*, 8(1), 73. <https://doi.org/10.3390/su8010073>
- Castillo, F., Kordon, A., & Smits, G. (2006). Robust Pareto Front Genetic Programming Parameter Selection Based on Design of Experiments and Industrial Data. *Genetic Programming Theory and Practice {IV}*, 5, 149–166. [https://doi.org/doi:10.1007/978-0-387-49650-4\\_10](https://doi.org/doi:10.1007/978-0-387-49650-4_10)

- Chami, M., & Robilliard, D. (2002). Inversion of oceanic constituents in case I and II waters with genetic programming algorithms. *Appl. Opt.*, *41*(30), 6260–6275. <https://doi.org/10.1364/AO.41.006260>
- Chen, D. Y., Chuang, T. R., & Tsai, S. C. (2001). JGAP: A Java-based graph algorithms platform. *Software - Practice and Experience*, *31*(7), 615–635. <https://doi.org/10.1002/spe.379>
- Cortez, P., Rocha, M., & Neves, J. (2001). Evolving time series forecasting neural network models, 1–25. <https://doi.org/10.1023/B:HEUR.0000034714.09838.1e>
- Dalinina, R. (2017). Introduction to Forecasting with ARIMA in R. Retrieved March 28, 2018, from <https://www.datascience.com/blog/introduction-to-forecasting-with-arima-in-r-learn-data-science-tutorials>
- Dhakate, P. P., Patil, S., Rajeswari, K., & Abin, D. (2014). Preprocessing and Classification in WEKA Using Different Classifiers, *4*(8), 91–93.
- Duyvesteyn, K., & Kaymak, U. (2005). Genetic Programming in Economic Modelling. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation* (Vol. 2, pp. 1025–1031). <https://doi.org/doi:10.1109/CEC.2005.1554803>
- EEA. (2017). *Air quality in Europe — 2017 report*. <https://doi.org/10.2800/850018>
- Elbir, T., Kara, M., Bayram, A., Altioek, H., & Dumanoglu, Y. (2011). Comparison of predicted and observed PM10 concentrations in several urban street canyons. *Air Quality, Atmosphere and Health*, *4*(2), 121–131. <https://doi.org/10.1007/s11869-010-0080-9>
- Ellson, J., Gansner, E., Koutsofios, L., North, S. C., & Woodhull, G. (2002). *Graphviz—open source graph drawing tools*. *Graph Drawing*. [https://doi.org/10.1007/3-540-68339-9\\_34](https://doi.org/10.1007/3-540-68339-9_34)
- EPA USA. (2007). AQI Forecasts: Your Advance Notification About Unhealthy Air, (February 2007), 1–2.
- Esling, P., & Agon, C. (2012). Time-series data mining. *ACM Computing Surveys*, *45*(1), 1–34. <https://doi.org/10.1145/2379776.2379788>
- Ferreira, T. A. E., Vasconcelos, G. C., & Adeodato, P. J. L. (2007). A New Evolutionary Approach for Time Series Forecasting. *2007 IEEE Symposium on Computational Intelligence and Data Mining*. <https://doi.org/10.1109/CIDM.2007.368933>
- Fu, T. C. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, *24*(1), 164–181. <https://doi.org/10.1016/j.engappai.2010.09.007>
- Graff, M., Escalante, H. J., Ornelas-Tellez, F., & Tellez, E. S. (2016). Time series forecasting with genetic programming. *Natural Computing*, *16*(1), 165–174. <https://doi.org/10.1007/s11047-015-9536-z>
- Graff, M., Pena, R., & Medina, A. (2013). Wind Speed Forecasting using Genetic Programming, 408–415.



- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software. *ACM SIGKDD Explorations*, 11(1), 10–18. <https://doi.org/10.1145/1656274.1656278>
- Hauptman, A., & Sipper, M. (2007). Evolution of an Efficient Search Algorithm for the Mate-In- $\{N\}$  Problem in Chess. *Proceedings of the 10th European Conference on Genetic Programming*, 4445, 78–89. [https://doi.org/doi:10.1007/978-3-540-71605-1\\_8](https://doi.org/doi:10.1007/978-3-540-71605-1_8)
- Hoai, N. X., McKay, R. I., Essam, D., & Chau, R. (2002). Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: The comparative results. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002* (Vol. 2, pp. 1326–1331). <https://doi.org/10.1109/CEC.2002.1004435>
- Holladay, K. L., & Robbins, K. A. (2007). Evolution of Signal Processing Algorithms using Vector Based Genetic Programming. In *15th International Conference on Digital Signal Processing* (pp. 503–506). <https://doi.org/doi:10.1109/ICDSP.2007.4288629>
- Hyndman, R. J., & Athanasopoulos, G. (2013). *Forecasting: principles and practice*. Retrieved March 28, 2018, from <http://otexts.org/fpp/2/3>
- Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal Of Statistical Software*, 27(3), C3–C3. <https://doi.org/10.18637/jss.v027.i03>
- Jacob, C., & Burleigh, I. (2005). Genetic Programming inside a Cell. In *Genetic Programming Theory and Practice {III}* (Vol. 9, pp. 191–206). [https://doi.org/doi:10.1007/0-387-28111-8\\_13](https://doi.org/doi:10.1007/0-387-28111-8_13)
- Jayanthi, S. K., & Sasikala, S. (2013). Reptree Classifier for Identifying Link Spam in Web Search Engines. *ICTACT Journal on Soft Computing*, 3(2), 498–505. <https://doi.org/10.21917/ijsc.2013.0075>
- Kalmegh, S. (2015). Analysis of WEKA Data Mining Algorithm REPTree , Simple Cart and RandomTree for Classification of Indian News. *International Journal of Innovative Science, Engineering & Technology*, 2(2), 438–446.
- Koza, J. R. (1992). *Genetic Programming On the Programming of Computers by Means of Natural Selection*. Massachusetts Institute of Technology. [https://doi.org/10.1016/0303-2647\(94\)90062-0](https://doi.org/10.1016/0303-2647(94)90062-0)
- Koza, J. R. (1994). Introduction to genetic programming. *Advances in Genetic Programming*, 22(1983), 21–42. <https://doi.org/doi:10.1145/1388969.1389057>
- Koza, J. R., Keane, M. A., Streeter, M. J., William, M., Yu, J., Lanza, G., ... Lanza, G. (2003). Genetic Programming IV: Routine Human-Competitive Machine Intelligence. *Genetic Programming and Evolvable Machines*, 6(1), 231–233. <https://doi.org/10.1007/s10710-005-7579-0>
- Krajewski, L. J., Ritzman, L. P., & Malhotra, M. K. (2007). Operations management – processes and value chain. *International Journal of Production Economics*. <https://doi.org/10.1016/j.ijpe.2008.07.023>
- Lee, Y.-S., & Tong, L.-I. (2011). Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming.

- Knowledge-Based Systems*, 24(1), 66–72.  
<https://doi.org/10.1016/j.knosys.2010.07.006>
- Lohn, J. D., Linden, D. S., Hornby, G. S., Kraus, W. F., Rodríguez-Arroyo, A., & Seufert, S. E. (2003). Evolutionary design of an X-band antenna for NASA's Space Technology 5 mission. In *Proceedings - NASA/DoD Conference on Evolvable Hardware, EH* (Vol. 2003–Janua, pp. 155–163).  
<https://doi.org/10.1109/EH.2003.1217660>
- Luke, S. (1998). {ECJ} Evolutionary Computation Library.
- MoEU. (n.d.). Air Quality Monitoring Stations Website. Retrieved May 15, 2018, from <http://www.havaizleme.gov.tr/Default.ltr.aspx>
- OECD. (2008). *OECD Environmental Performance Reviews: Turkey 2008*. OECD Publishing. <https://doi.org/10.1787/9789264049161-en>
- Oliveri Conti, G., Heibati, B., Kloog, I., Fiore, M., & Ferrante, M. (2017). A review of AirQ Models and their applications for forecasting the air pollution health outcomes. *Environmental Science and Pollution Research*, 24(7), 6426–6445.  
<https://doi.org/10.1007/s11356-016-8180-1>
- Otero, F., Castle, T., & Johnson, C. (2012). EpochX: genetic programming in java with statistics and event monitoring. *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion*, 93–100.  
<https://doi.org/doi:10.1145/2330784.2330800>
- Paci, L., Gelfand, A. E., & Holland, D. M. (2013). Spatio-temporal modeling for real-time ozone forecasting. *Spatial Statistics*, 4, 79–93.  
<https://doi.org/10.1016/j.spasta.2013.04.003>
- Peng, H., Lima, A. R., Teakles, A., Jin, J., Cannon, A. J., & Hsieh, W. W. (2017). Evaluating hourly air quality forecasting in Canada with nonlinear updatable machine learning methods. *Air Quality, Atmosphere and Health*, 10(2), 195–211. <https://doi.org/10.1007/s11869-016-0414-3>
- Poli, R., Langdon, W., & McPhee, N. (2008). *A field guide to genetic programming (With contributions by JR Koza)*(2008). Published vi a <http://lulu.com>. Retrieved from <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+Field+Guide+to+Genetic+Programming+with+contributions+by#1>
- Poncela, P. (2004). Time series analysis by state space methods. *International Journal of Forecasting*, 20(1), 139–141.  
<https://doi.org/10.1016/j.ijforecast.2003.11.005>
- Potvin, J. Y., Soriano, P., & Maxime, V. (2004). Generating trading rules on the stock markets with genetic programming. *Computers and Operations Research*, 31(7), 1033–1047. [https://doi.org/10.1016/S0305-0548\(03\)00063-7](https://doi.org/10.1016/S0305-0548(03)00063-7)
- Qiu, X., Zhang, L., Ren, Y., Suganthan, P., & Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. In *2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL)* (pp. 1–6). IEEE. <https://doi.org/10.1109/CIEL.2014.7015739>



- R Core Team. (2017). R: A Language and Environment for Statistical Computing. Vienna, Austria. Retrieved from <https://www.r-project.org>
- Rahman, N. H. A., Lee, M. H., Suhartono, & Latif, M. T. (2015). Artificial neural networks and fuzzy time series forecasting: an application to air quality. *Quality & Quantity*, 49(6), 2633–2647. <https://doi.org/10.1007/s11135-014-0132-6>
- Rodriguez-Vazquez, K. (2001). Genetic Programming in Time Series Modelling: An Application to Meteorological Data. *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, 1(May), 261–266. <https://doi.org/doi:10.1109/CEC.2001.934399>
- Sammut, C., & Webb, G. I. (Eds.). (2017). Random Forests. In *Encyclopedia of Machine Learning and Data Mining* (p. 1054). Boston, MA: Springer US. [https://doi.org/10.1007/978-1-4899-7687-1\\_695](https://doi.org/10.1007/978-1-4899-7687-1_695)
- Santini, M., & Tettamanzi, A. (2001). Genetic Programming for Financial Time Series Prediction. *Genetic Programming, Proceedings of EuroGP'2001*, 2038, 361–370. [https://doi.org/10.1007/3-540-45355-5\\_29](https://doi.org/10.1007/3-540-45355-5_29)
- Sapankevych, N., & Sankar, R. (2009). Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2), 24–38. <https://doi.org/10.1109/MCI.2009.932254>
- Shengwu, X., & Weiwu, W. (2003). A new hybrid structure genetic programming in symbolic regression. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003* (pp. 1500–1506). <https://doi.org/doi:10.1109/CEC.2003.1299850>
- Silva, S. (2007). GPLAB: A Genetic Programming Toolbox for MATLAB. *ECOS - Evolutionary and Complex Systems Group, Version 3*(April), 1–73.
- USEPA. (n.d.). AirNow AQI Calculator. Retrieved March 28, 2018, from [www.airnow.gov/index.cfm?action=airnow.calculator](http://www.airnow.gov/index.cfm?action=airnow.calculator)
- USEPA. (2013). Technical Assistance Document for the Reporting of Daily Air Quality – the Air Quality Index (AQI). *Environmental Protection*, (May), 1–28.
- USEPA. (2014). Air Quality Index (AQI). *A Guide to Air Quality and Your Health*, (February), 12. Retrieved from <http://airnow.gov/index.cfm?action=aqibasics.aqi>
- Vázquez, K. R., & Escolar, C. (2001). Genetic Programming in Time Series Modelling : an Application To Meteorological Data. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 1(1989), 261–266. <https://doi.org/10.1109/CEC.2001.934399>
- Wagner, S., & Affenzeller, M. (2002). The HeuristicLab Optimization Environment. *Evolutionary Computation*, 1–15.
- Wang, D., Wei, S., Luo, H., Yue, C., & Grunder, O. (2017). A novel hybrid model for air quality index forecasting based on two-phase decomposition technique and modified extreme learning machine. *Science of the Total Environment*, 580, 719–733. <https://doi.org/10.1016/j.scitotenv.2016.12.018>

- WAQI. (n.d.). Real-time Air Quality Index (AQI). Retrieved March 28, 2018, from <http://aqicn.org>
- Witten, I. H., Frank, E., & Hall, M. a. (2011). *Data Mining: Practical Machine Learning Tools and Techniques (Google eBook). Complementary literature None*. <https://doi.org/0120884070>, 9780120884070
- World Health Organization. (2016). Ambient Air Pollution: A global assessment of exposure and burden of disease. *World Health Organization*, 1–131. <https://doi.org/9789241511353>
- Worzel, W. P., Yu, J., Almal, A. a, & Chinnaiyan, A. M. (2009). Applications of genetic programming in cancer research. *The International Journal of Biochemistry & Cell Biology*, 41(2), 405–13. <https://doi.org/10.1016/j.biocel.2008.09.025>
- Xu, C., Li, Z., & Wang, W. (2016). Short-term traffic flow prediction using a methodology based on autoregressive integrated moving average and genetic programming. *Transport*, 31(3). <https://doi.org/10.3846/16484142.2016.1212734>
- Xu, C., Wang, W., & Liu, P. (2013). A genetic programming model for real-time crash prediction on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 14(2), 574–586. <https://doi.org/10.1109/TITS.2012.2226240>
- Zhou, G., Xu, J., Xie, Y., Chang, L., Gao, W., Gu, Y., & Zhou, J. (2017). Numerical air quality forecasting over eastern China: An operational application of WRF-Chem. *Atmospheric Environment*, 153, 94–108. <https://doi.org/10.1016/j.atmosenv.2017.01.020>

## APPENDIX 1 – ECJ Function Set Parameter Settings

```
parent.0 = koza.params

gp.fs.size = 1

gp.fs.0 = ec.gp.GPFunctionSet

# We'll call the function set "f0".
gp.fs.0.name = f0

# weekday attribute terminal nodes
gp.fs.0.size = 65
gp.fs.0.func.0 = common.category.weekday.WD1
gp.fs.0.func.0.nc = booleanterminal
gp.fs.0.func.1 = common.category.weekday.WD2
gp.fs.0.func.1.nc = booleanterminal
gp.fs.0.func.2 = common.category.weekday.WD3
gp.fs.0.func.2.nc = booleanterminal
gp.fs.0.func.3 = common.category.weekday.WD4
gp.fs.0.func.3.nc = booleanterminal
gp.fs.0.func.4 = common.category.weekday.WD5
gp.fs.0.func.4.nc = booleanterminal
gp.fs.0.func.5 = common.category.weekday.WD6
gp.fs.0.func.5.nc = booleanterminal
gp.fs.0.func.6 = common.category.weekday.WD7
gp.fs.0.func.6.nc = booleanterminal

# hour attribute terminal nodes
gp.fs.0.func.7 = common.category.hour.TD1
gp.fs.0.func.7.nc = booleanterminal
gp.fs.0.func.8 = common.category.hour.TD2
gp.fs.0.func.8.nc = booleanterminal
gp.fs.0.func.9 = common.category.hour.TD3
gp.fs.0.func.9.nc = booleanterminal
gp.fs.0.func.10 = common.category.hour.TD4
gp.fs.0.func.10.nc = booleanterminal
gp.fs.0.func.11 = common.category.hour.TD5
gp.fs.0.func.11.nc = booleanterminal
gp.fs.0.func.12 = common.category.hour.TD6
```

```
gp.fs.0.func.12.nc = booleanterminal
gp.fs.0.func.13 = common.category.hour.TD7
gp.fs.0.func.13.nc = booleanterminal
gp.fs.0.func.14 = common.category.hour.TD8
gp.fs.0.func.14.nc = booleanterminal
gp.fs.0.func.15 = common.category.hour.TD9
gp.fs.0.func.15.nc = booleanterminal
gp.fs.0.func.16 = common.category.hour.TD10
gp.fs.0.func.16.nc = booleanterminal
gp.fs.0.func.17 = common.category.hour.TD11
gp.fs.0.func.17.nc = booleanterminal
gp.fs.0.func.18 = common.category.hour.TD12
gp.fs.0.func.18.nc = booleanterminal
gp.fs.0.func.19 = common.category.hour.TD13
gp.fs.0.func.19.nc = booleanterminal
gp.fs.0.func.20 = common.category.hour.TD14
gp.fs.0.func.20.nc = booleanterminal
gp.fs.0.func.21 = common.category.hour.TD15
gp.fs.0.func.21.nc = booleanterminal
gp.fs.0.func.22 = common.category.hour.TD16
gp.fs.0.func.22.nc = booleanterminal
gp.fs.0.func.23 = common.category.hour.TD17
gp.fs.0.func.23.nc = booleanterminal
gp.fs.0.func.24 = common.category.hour.TD18
gp.fs.0.func.24.nc = booleanterminal
gp.fs.0.func.25 = common.category.hour.TD19
gp.fs.0.func.25.nc = booleanterminal
gp.fs.0.func.26 = common.category.hour.TD20
gp.fs.0.func.26.nc = booleanterminal
gp.fs.0.func.27 = common.category.hour.TD21
gp.fs.0.func.27.nc = booleanterminal
gp.fs.0.func.28 = common.category.hour.TD22
gp.fs.0.func.28.nc = booleanterminal
gp.fs.0.func.29 = common.category.hour.TD23
gp.fs.0.func.29.nc = booleanterminal
gp.fs.0.func.30 = common.category.hour.TD24
gp.fs.0.func.30.nc = booleanterminal

gp.fs.0.func.31 = canakkale.attributes.AirPressure
gp.fs.0.func.31.nc = nc0
gp.fs.0.func.32 = canakkale.attributes.WindSpeed
```

```
gp.fs.0.func.32.nc = nc0

# wind direction attribute terminal nodes
gp.fs.0.func.33 =
canakkale.category.wind_direction.WindN
gp.fs.0.func.33.nc = booleanterminal
gp.fs.0.func.34 =
canakkale.category.wind_direction.WindNNE
gp.fs.0.func.34.nc = booleanterminal
gp.fs.0.func.35 =
canakkale.category.wind_direction.WindNE
gp.fs.0.func.35.nc = booleanterminal
gp.fs.0.func.36 =
canakkale.category.wind_direction.WindENE
gp.fs.0.func.36.nc = booleanterminal
gp.fs.0.func.37 =
canakkale.category.wind_direction.WindE
gp.fs.0.func.37.nc = booleanterminal
gp.fs.0.func.38 =
canakkale.category.wind_direction.WindESE
gp.fs.0.func.38.nc = booleanterminal
gp.fs.0.func.39 =
canakkale.category.wind_direction.WindSE
gp.fs.0.func.39.nc = booleanterminal
gp.fs.0.func.40 =
canakkale.category.wind_direction.WindSSE
gp.fs.0.func.40.nc = booleanterminal
gp.fs.0.func.41 =
canakkale.category.wind_direction.WindS
gp.fs.0.func.41.nc = booleanterminal
gp.fs.0.func.42 =
canakkale.category.wind_direction.WindSSW
gp.fs.0.func.42.nc = booleanterminal
gp.fs.0.func.43 =
canakkale.category.wind_direction.WindSW
gp.fs.0.func.43.nc = booleanterminal
gp.fs.0.func.44 =
canakkale.category.wind_direction.WindWSW
gp.fs.0.func.44.nc = booleanterminal
gp.fs.0.func.45 =
canakkale.category.wind_direction.WindW
gp.fs.0.func.45.nc = booleanterminal
gp.fs.0.func.46 =
canakkale.category.wind_direction.WindWNW
gp.fs.0.func.46.nc = booleanterminal
```

```

gp.fs.0.func.47 =
canakkale.category.wind_direction.WindNW
gp.fs.0.func.47.nc = booleanterminal
gp.fs.0.func.48 =
canakkale.category.wind_direction.WindNNW
gp.fs.0.func.48.nc = booleanterminal

gp.fs.0.func.49 = canakkale.attributes.AirTemperature
gp.fs.0.func.49.nc = nc0
gp.fs.0.func.50 = canakkale.attributes.Humidity
gp.fs.0.func.50.nc = nc0

# logical and arithmetic function nodes
gp.fs.0.func.51 = common.functions.Mul
gp.fs.0.func.51.nc = nc2
gp.fs.0.func.52 = common.functions.Add
gp.fs.0.func.52.nc = nc2
gp.fs.0.func.53 = common.functions.Sub
gp.fs.0.func.53.nc = nc2
gp.fs.0.func.54 = common.functions.Div
gp.fs.0.func.54.nc = nc2
gp.fs.0.func.55 = common.functions.RegERC
gp.fs.0.func.55.nc = nc0
gp.fs.0.func.56 = common.functions.Square
gp.fs.0.func.56.nc = nc1
gp.fs.0.func.57 = common.functions.Sqrt
gp.fs.0.func.57.nc = nc1
gp.fs.0.func.58 = common.functions.Equal
gp.fs.0.func.58.nc = booleannode
gp.fs.0.func.59 = common.functions.GreaterThan
gp.fs.0.func.59.nc = booleannode
gp.fs.0.func.60 = common.functions.GreaterThanEqual
gp.fs.0.func.60.nc = booleannode
gp.fs.0.func.61 = common.functions.LessThan
gp.fs.0.func.61.nc = booleannode
gp.fs.0.func.62 = common.functions.LessThanEqual
gp.fs.0.func.62.nc = booleannode
gp.fs.0.func.63 = common.functions.Iff
gp.fs.0.func.63.nc = ifstatement

# moving average attribute terminal node

```

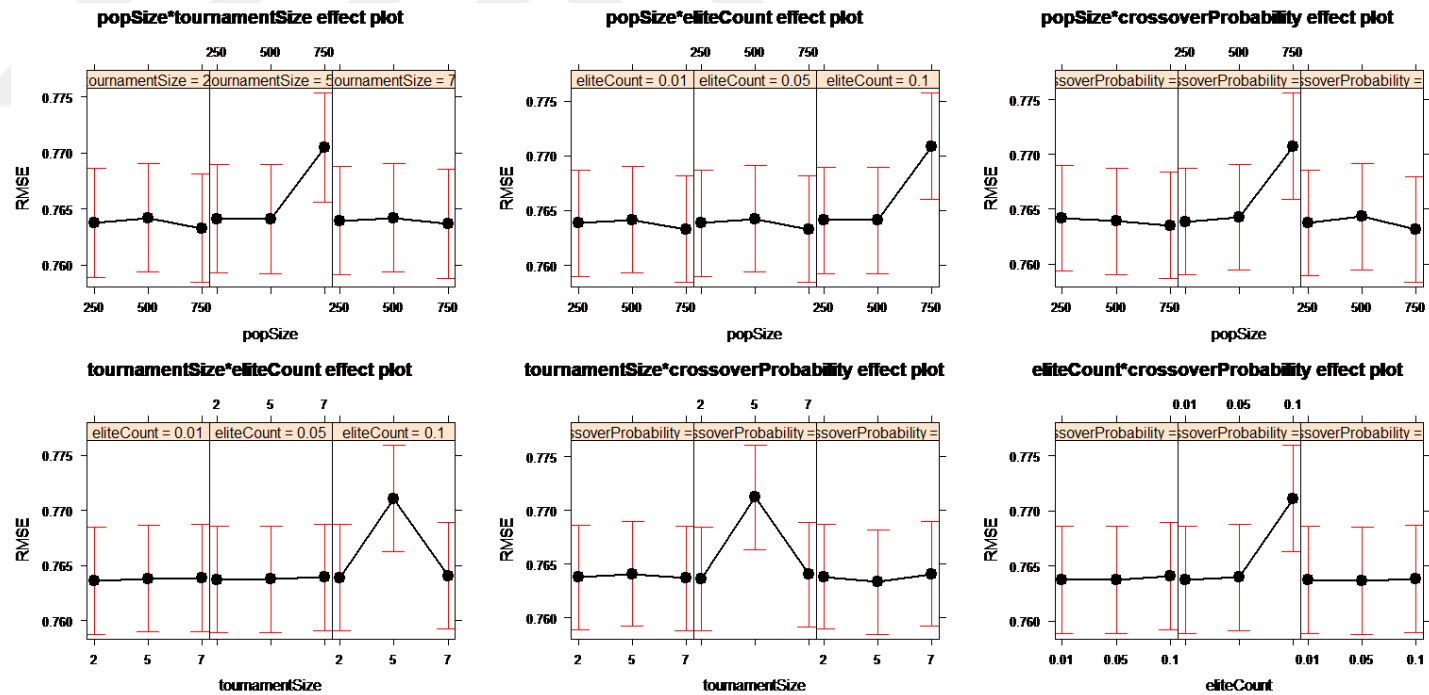
```
gp.fs.0.func.64 = canakkale.attributes.MovingAverage  
gp.fs.0.func.64.nc = nc0
```







## APPENDIX 2 – Experimental Design Plots



**Figure A2.5.1.** The interval plots of the parameters used in the first experiment for the RMSE measurements of PM forecasting with GP when the maximum tree depth is seven.

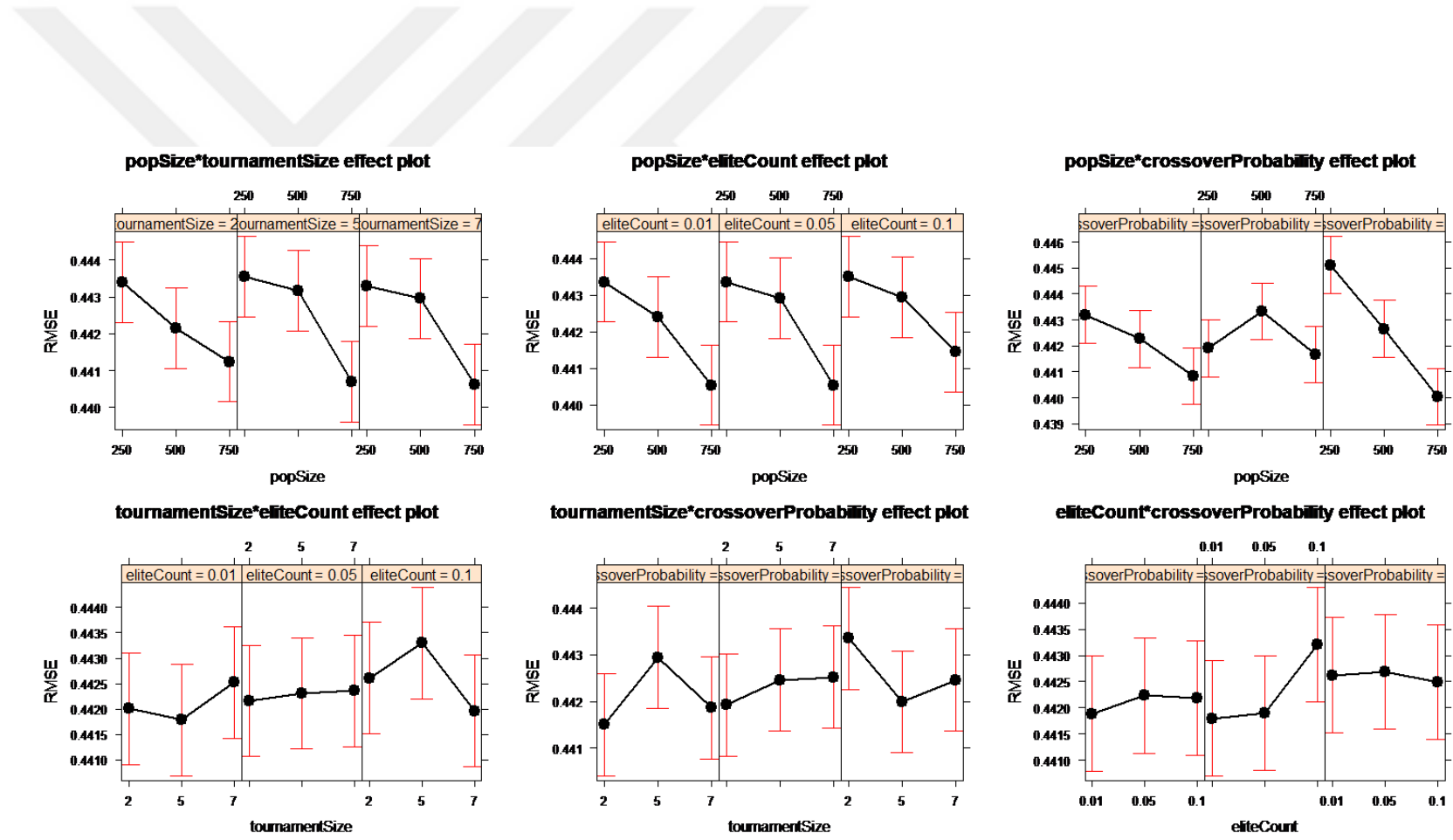


Figure A2.5.2. The interval plots of the parameters used in the first experiment or the RMSE measurements of SO<sub>2</sub> forecasting with GP when the maximum tree depth is seven.

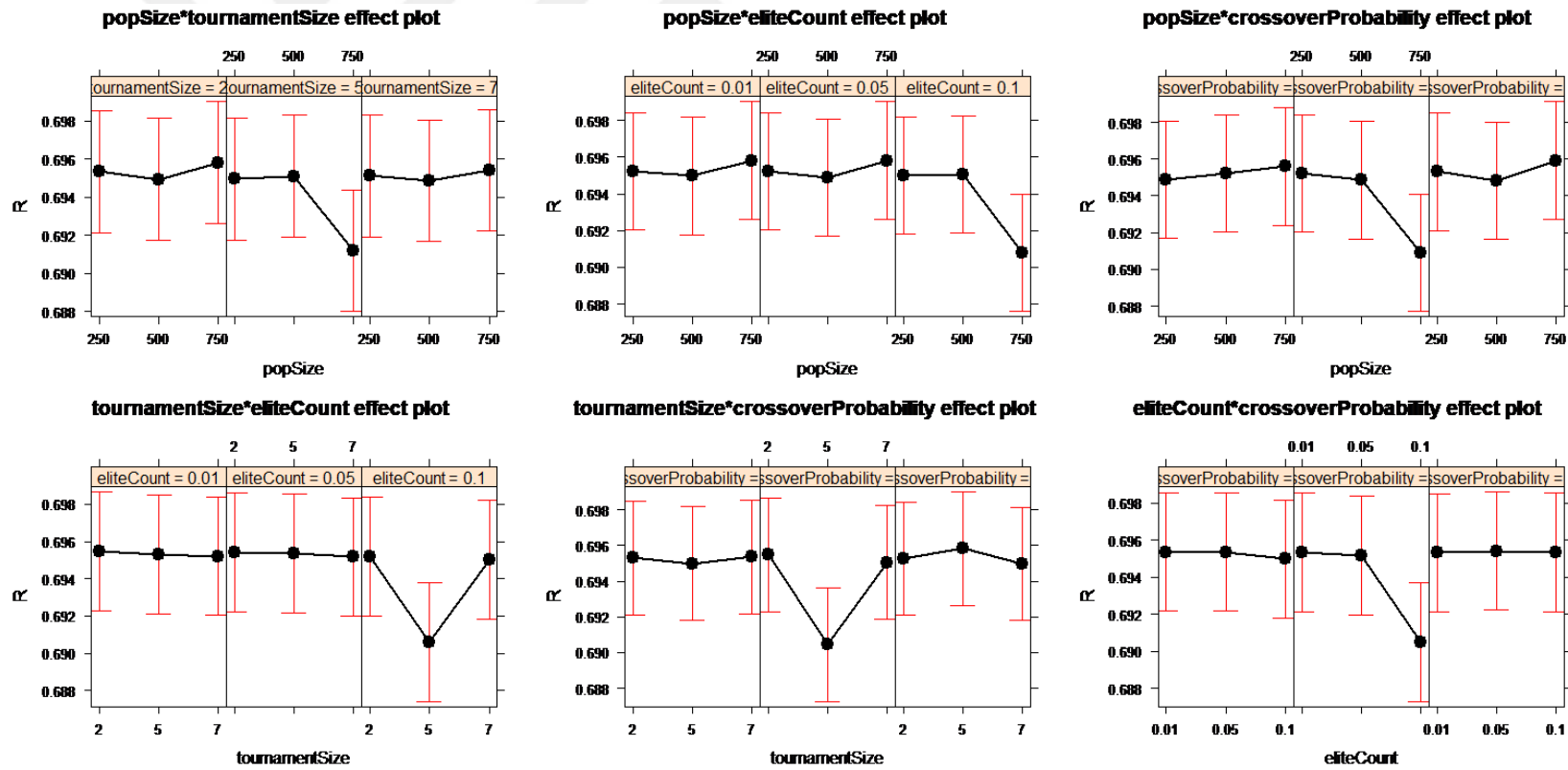


Figure A2.5.3. The interval plots of the parameters used in the used in the first experiment for correlation coefficient (R) measurements of PM forecasting with GP when the maximum tree depth is seven.

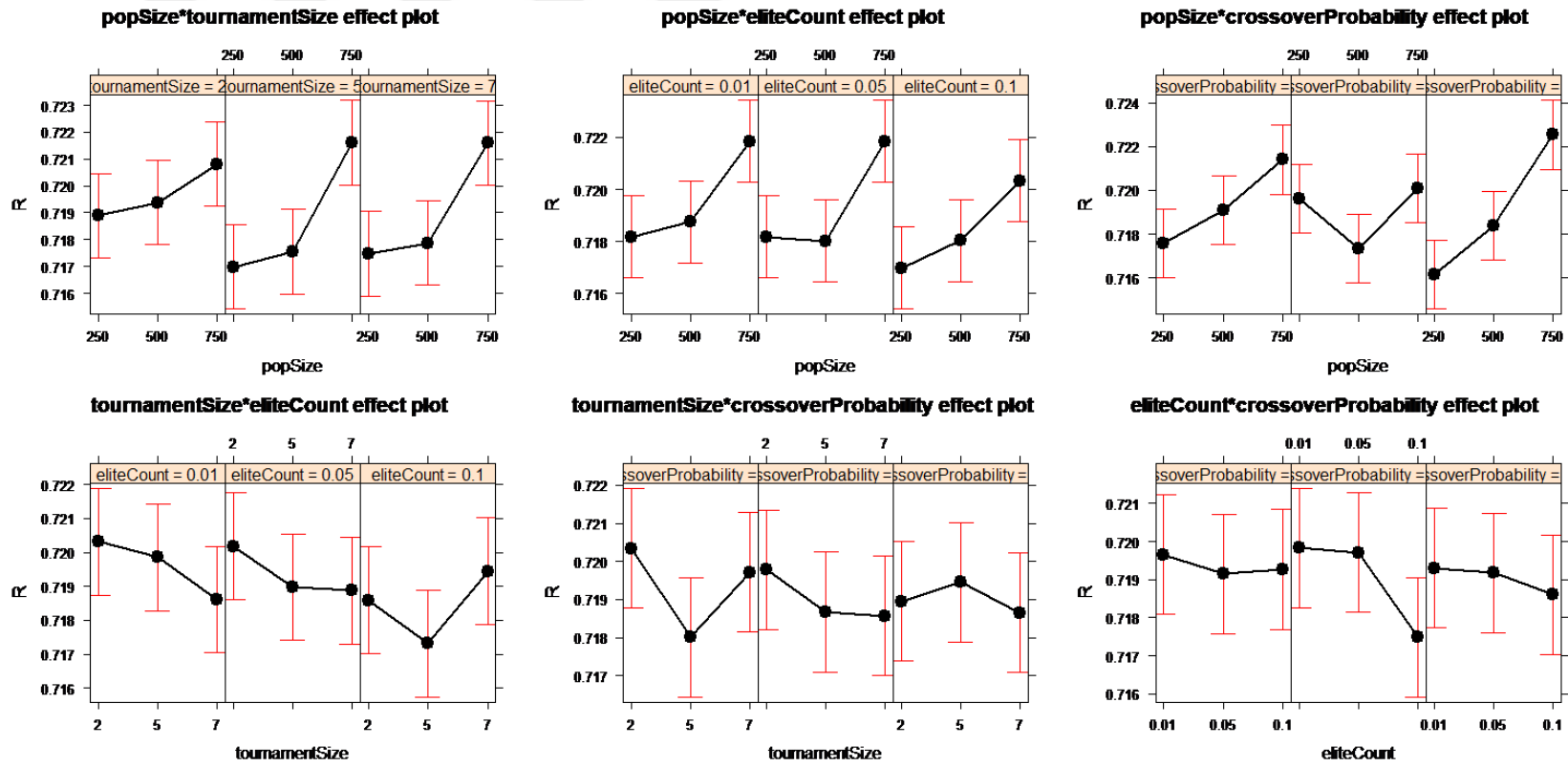


Figure A2.5.4. The interval plots of the parameters used in the used in the first experiment for correlation coefficient (R) measurements of SO<sub>2</sub> forecasting with GP when the maximum tree depth is seven.

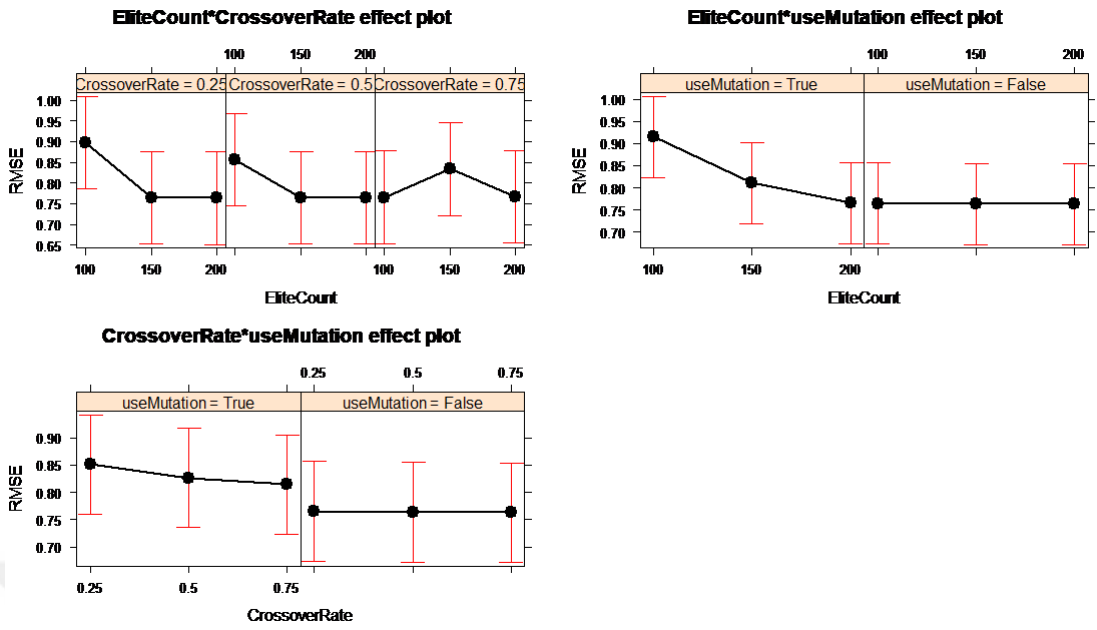


Figure A2.5. The interval plots of the parameters used in the second experiment for the RMSE measurements of PM forecasting using GP

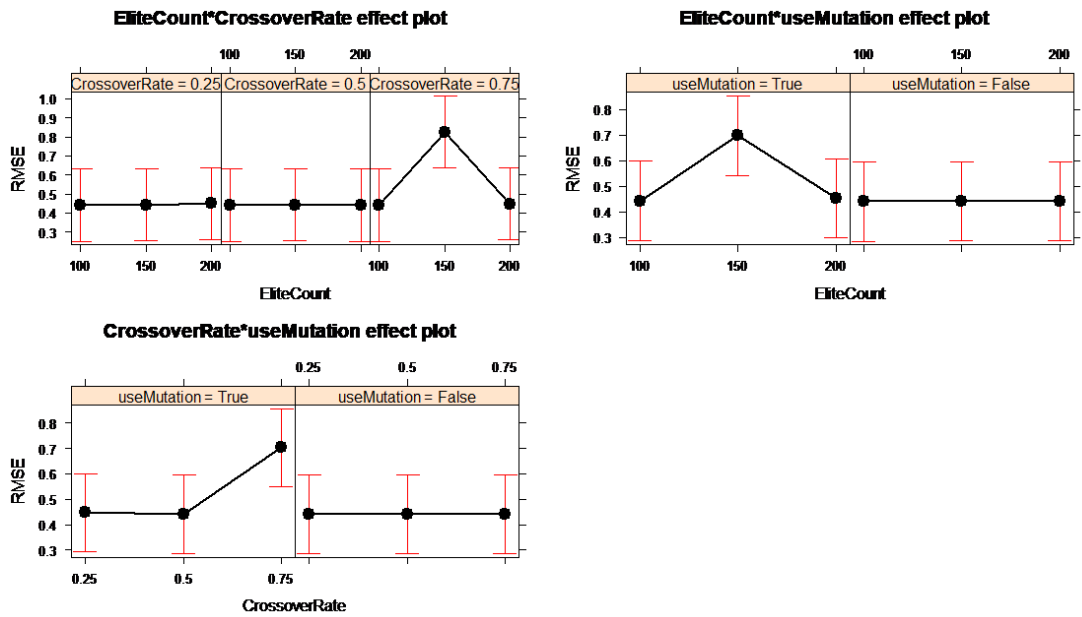
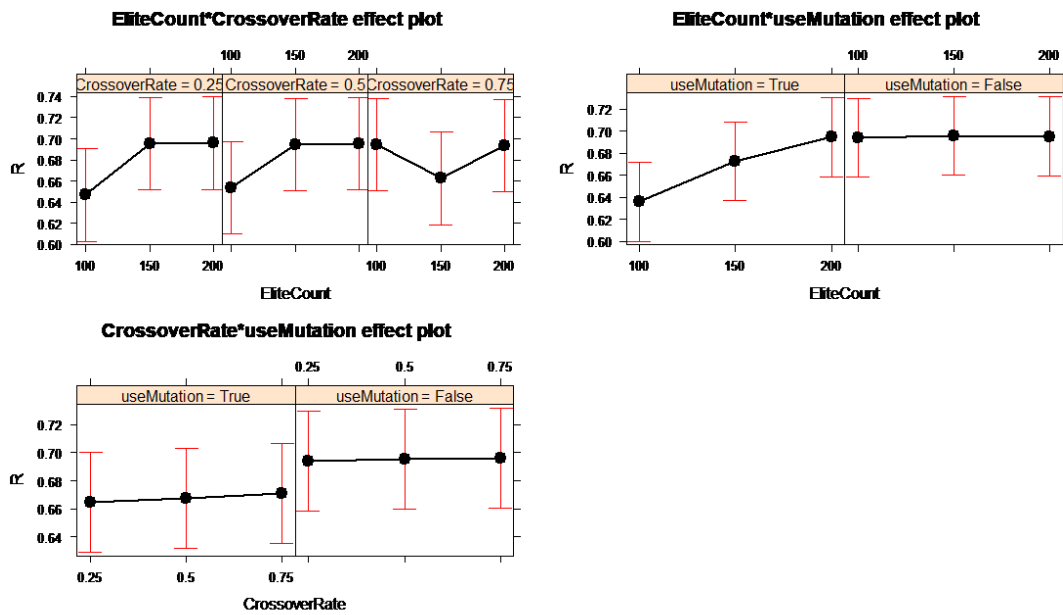
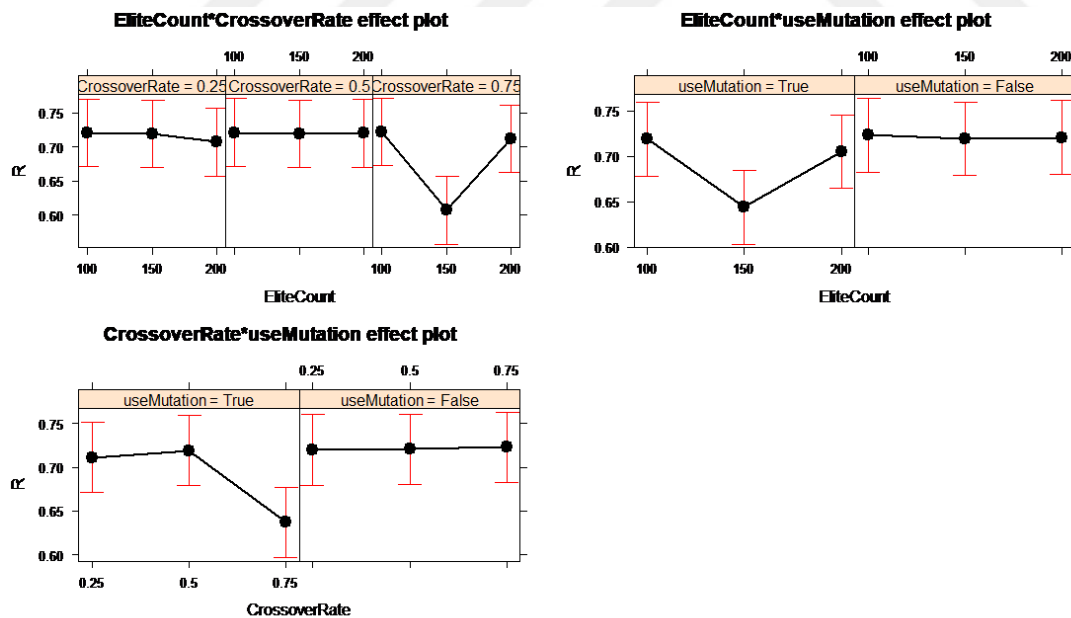


Figure A2.5.5. The interval plots of the parameters used in the second experiment for the RMSE measurements of SO<sub>2</sub> forecasting using GP



**Figure A2.5.6.** The interval plots of the parameters used in the second experiment for the correlation coefficient (R) measurements of PM forecasting using GP



**Figure A2.5.7.** The interval plots of the parameters used in the second experiment for the correlation coefficient (R) measurements of SO<sub>2</sub> forecasting using GP

### APPENDIX 3 – GP and Decision Tree Algorithm Results

**Table A3.1.** GP's PM forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
2	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
3	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
4	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
5	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
6	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
7	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
8	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
9	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
10	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
11	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
12	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
13	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
14	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
15	0.694721902	0.486832173	0.764344083	0.719945096	0.71927846
16	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
17	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
18	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
19	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
20	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
21	0.699593816	0.481893545	0.760064331	0.712641673	0.715251041
22	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
23	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
24	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
25	0.694721902	0.48682755	0.764344088	0.719938259	0.719278465
Average	0.694916779	0.486630375	0.764172898	0.719646669	0.719117368

**Table A3.2.** Random Forest's PM forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.6499	0.5185	0.8177	0.766785	0.769469
2	0.6561	0.517	0.8153	0.764601	0.767247
3	0.6549	0.5157	0.8151	0.762733	0.767043
4	0.6536	0.5163	0.8163	0.763595	0.768124
5	0.6658	0.5108	0.8033	0.755422	0.755963
6	0.6536	0.5189	0.8149	0.767367	0.766891
7	0.6547	0.5161	0.8142	0.763269	0.766198
8	0.6555	0.5149	0.8135	0.761569	0.765527
9	0.6567	0.5167	0.8146	0.764183	0.766611
10	0.6524	0.5138	0.8162	0.759812	0.768066
11	0.6556	0.5177	0.8141	0.765566	0.766062
12	0.6508	0.5192	0.8173	0.767783	0.769151
13	0.6536	0.5167	0.8154	0.764122	0.767284
14	0.652	0.5167	0.8162	0.764158	0.768097
15	0.645	0.5217	0.8237	0.771556	0.775144
16	0.6512	0.5199	0.8196	0.76892	0.771301
17	0.6532	0.5158	0.8153	0.762832	0.767216
18	0.6558	0.5166	0.8149	0.763993	0.766833
19	0.6496	0.5187	0.8184	0.767185	0.770152
20	0.6574	0.5148	0.8127	0.761417	0.764793
21	0.6551	0.5161	0.8143	0.763259	0.766263
22	0.6586	0.5147	0.8097	0.761247	0.761956
23	0.6517	0.5176	0.8167	0.765433	0.768502
24	0.6492	0.5199	0.8219	0.768817	0.773411
25	0.6525	0.5175	0.8154	0.765401	0.767316
Average	0.65378	0.516892	0.815468	0.764441	0.7673848



**Table A3.3.** Random Tree's PM forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.3892	0.7738	1.1218	1.144428	1.055668
2	0.3642	0.7832	1.1673	1.158355	1.098517
3	0.3171	0.8163	1.205	1.207217	1.133947
4	0.4606	0.7325	1.0541	1.083371	0.991913
5	0.387	0.7654	1.1375	1.131997	1.07045
6	0.4254	0.7328	1.1093	1.083765	1.0439
7	0.4256	0.738	1.0889	1.091397	1.024655
8	0.4666	0.7104	1.0515	1.05064	0.989503
9	0.3901	0.7448	1.1109	1.101468	1.04536
10	0.4852	0.7399	1.0748	1.094229	1.011426
11	0.4124	0.7561	1.1283	1.118227	1.061747
12	0.415	0.7558	1.1173	1.117834	1.051444
13	0.4241	0.773	1.112	1.143169	1.046434
14	0.4527	0.749	1.0771	1.107763	1.01359
15	0.4101	0.7579	1.1401	1.120824	1.072839
16	0.3527	0.7758	1.136	1.147418	1.069056
17	0.3228	0.8696	1.2859	1.286134	1.210103
18	0.4279	0.7821	1.1486	1.156624	1.080908
19	0.3705	0.7723	1.1267	1.142225	1.060292
20	0.4004	0.7562	1.0971	1.118385	1.032441
21	0.3568	0.7695	1.1371	1.137976	1.070066
22	0.3274	0.7755	1.1913	1.146868	1.121106
23	0.3759	0.7771	1.1365	1.149228	1.069506
24	0.3905	0.7553	1.1546	1.116968	1.086561
25	0.4245	0.7522	1.1093	1.112405	1.043876
Average	0.398988	0.76458	1.12876	1.1307566	1.06221232

**Table A3.4.** Rep Tree's PM forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.6746	0.4967	0.7845	0.734507	0.738266
2	0.684	0.4948	0.7753	0.731821	0.729597
3	0.6601	0.5084	0.7983	0.751927	0.751197
4	0.6634	0.5029	0.7953	0.743745	0.7484
5	0.6483	0.5115	0.8092	0.756415	0.761465
6	0.6786	0.4986	0.7807	0.737373	0.734668
7	0.6658	0.5033	0.7929	0.744275	0.746181
8	0.6708	0.5031	0.791	0.744025	0.744359
9	0.681	0.4969	0.7789	0.734909	0.732967
10	0.6918	0.4934	0.7679	0.72969	0.722631
11	0.6933	0.489	0.7659	0.723184	0.720772
12	0.6785	0.492	0.7807	0.727629	0.734681
13	0.6815	0.4957	0.7776	0.733072	0.731783
14	0.6754	0.5003	0.7838	0.739952	0.737601
15	0.6255	0.5099	0.8297	0.754068	0.780802
16	0.6644	0.5028	0.7943	0.743616	0.747425
17	0.6589	0.5082	0.7995	0.751647	0.752338
18	0.681	0.5009	0.7783	0.740822	0.732388
19	0.6655	0.5022	0.7931	0.742751	0.746383
20	0.6707	0.5057	0.7887	0.747888	0.742197
21	0.6768	0.4976	0.7824	0.735942	0.736253
22	0.6733	0.5011	0.7859	0.741085	0.739565
23	0.6667	0.503	0.792	0.743951	0.745319
24	0.6535	0.5102	0.8044	0.754531	0.756985
25	0.6863	0.4916	0.7729	0.727025	0.727308
Average	0.670788	0.500792	0.788128	0.740634	0.74166124

**Table A3.5.** GP's SO<sub>2</sub> forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
2	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
3	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
4	0.715878949	0.167701168	0.444203025	0.679979423	0.698224594
5	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
6	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
7	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
8	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
9	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
10	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
11	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
12	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
13	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
14	0.726360377	0.169077141	0.437581005	0.68555859	0.687815711
15	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
16	0.71606172	0.167455394	0.444083664	0.67898288	0.698036975
17	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
18	0.730038057	0.163717903	0.435097259	0.663828443	0.683911612
19	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
20	0.722191037	0.166979284	0.441718281	0.677052392	0.694318926
21	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
22	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
23	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
24	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
25	0.71606172	0.167452172	0.444083663	0.678969814	0.698036973
Average	0.717270582	0.167358973	0.44337426	0.67859192	0.696921891

**Table A3.6.** Random Forest's SO<sub>2</sub> forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.6206	0.2584	0.5048	1.047741	0.793441
2	0.622	0.2562	0.5024	1.038972	0.789697
3	0.6258	0.2559	0.5008	1.037807	0.787212
4	0.6068	0.2609	0.5113	1.057754	0.803751
5	0.6442	0.251	0.4912	1.017619	0.772045
6	0.6233	0.2516	0.5015	1.020101	0.788248
7	0.6197	0.257	0.5033	1.042136	0.791155
8	0.6144	0.2605	0.507	1.05625	0.7969
9	0.6122	0.2561	0.5071	1.038337	0.797043
10	0.6444	0.2424	0.4901	0.983103	0.770297
11	0.6397	0.2456	0.4932	0.995759	0.775202
12	0.6164	0.2597	0.5083	1.053246	0.798978
13	0.6187	0.2534	0.5039	1.02736	0.792137
14	0.6046	0.2605	0.5121	1.05618	0.804941
15	0.6191	0.2489	0.503	0.100923	0.790706
16	0.6176	0.2561	0.5049	1.038677	0.793553
17	0.6172	0.2539	0.5055	1.029722	0.794556
18	0.6509	0.2433	0.4865	0.986438	0.764755
19	0.6145	0.2574	0.5081	1.043874	0.798647
20	0.6263	0.2526	0.4992	1.024312	0.784655
21	0.6379	0.2465	0.4928	0.999387	0.774631
22	0.6196	0.2582	0.5046	1.046937	0.793138
23	0.6363	0.2527	0.4958	1.024798	0.779303
24	0.6125	0.2605	0.5076	1.056174	0.797891
25	0.6222	0.2559	0.5025	1.037647	0.789837
Average	0.623476	0.254208	0.5019	1.03078244	0.78890876

**Table A3.7.** Random Tree's SO<sub>2</sub> forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.3065	0.4618	1.0365	1.87263	1.629202
2	0.1438	0.5983	1.4033	2.42629	2.205711
3	0.2691	0.5468	1.0726	2.217332	1.686041
4	0.2798	0.4612	0.9395	1.870081	1.476811
5	0.171	0.4903	1.0816	1.98836	1.700107
6	0.268	0.4719	0.9447	1.913465	1.484943
7	0.1694	0.5798	1.346	2.350906	2.115749
8	0.2872	0.5396	1.1604	2.188219	1.823907
9	0.2625	0.4913	1.0137	1.992249	1.59333
10	0.2871	0.5452	1.2991	2.210945	2.042013
11	0.3256	0.4711	1.0661	1.910193	1.675713
12	0.1554	0.5798	1.3217	2.351171	2.077588
13	0.2503	0.5572	1.147	2.259314	1.802992
14	0.3391	0.4563	0.9577	1.850191	1.505337
15	0.2614	0.4899	1.2138	1.986457	1.90789
16	0.2312	0.4802	1.0393	1.947138	1.633697
17	0.2976	0.5522	1.2594	2.238984	1.979555
18	0.2553	0.5192	1.1234	2.105189	1.765775
19	0.3426	0.4444	1.0296	1.802052	1.618366
20	0.1514	0.6613	1.4368	2.681736	2.258489
21	0.1583	0.4979	1.107	2.019074	1.740071
22	0.21	0.55	1.4384	2.23038	2.260917
23	0.1932	0.6031	1.2406	2.445536	1.950102
24	0.3058	0.5044	1.0606	2.04555	1.667085
25	0.1898	0.5794	1.2133	2.349532	1.907153
Average	0.244456	0.525304	1.158084	2.13011896	1.82034176

**Table A3.8.** Rep Tree's SO<sub>2</sub> forecasting results

	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
1	0.6971	0.1867	0.4569	0.757043	0.718113
2	0.6703	0.1923	0.4724	0.779593	0.742535
3	0.6705	0.1862	0.4739	0.755033	0.74493
4	0.6622	0.1889	0.484	0.766163	0.760829
5	0.7051	0.1788	0.4524	0.724943	0.711066
6	0.6526	0.1974	0.4831	0.800406	0.759416
7	0.7104	0.1849	0.4485	0.749669	0.705042
8	0.6498	0.1984	0.4842	0.804413	0.761111
9	0.6527	0.1912	0.4901	0.775477	0.770331
10	0.6199	0.2056	0.5045	0.83365	0.793039
11	0.6552	0.1894	0.4851	0.768126	0.762433
12	0.6653	0.1969	0.4762	0.798524	0.748567
13	0.6937	0.1834	0.4621	0.743891	0.726388
14	0.6364	0.1976	0.4994	0.801303	0.784949
15	0.7015	0.1878	0.4543	0.761648	0.714112
16	0.6684	0.1929	0.4806	0.782192	0.755371
17	0.6647	0.1875	0.4861	0.760346	0.764149
18	0.7041	0.1889	0.4521	0.765933	0.710675
19	0.665	0.1884	0.4775	0.764126	0.750567
20	0.6412	0.1943	0.4913	0.787863	0.7722
21	0.6637	0.1894	0.4788	0.768136	0.752575
22	0.679	0.1965	0.4701	0.796971	0.738934
23	0.704	0.1828	0.4524	0.74132	0.711173
24	0.6577	0.1946	0.4802	0.789094	0.754757
25	0.7056	0.1924	0.4514	0.780315	0.709494
Average	0.671844	0.190928	0.473904	0.77424712	0.74491024

## APPENDIX 4 – R Code used for ARIMA forecasting

```
# required libraries
library(forecast)
library('tseries')
library("readxl")
daily_data = read_excel('arima_PM.xlsx', sheet="Sheet1")

# moving average
daily_data$ma = ma(daily_data$PM, order=2)

# seasonal component calculation
count_ma = ts(na.omit(daily_data$ma), frequency=24)
decomp = stl(count_ma, s.window="periodic")
deseasonal_cnt <- seasadj(decomp)

# the augmented Dickey-Fuller (ADF) test
adf.test(count_ma, alternative = "stationary")

# train set and test set splitting
training<-deseasonal_cnt[1:5716]
testing<-deseasonal_cnt[5717:8164]

# model fitting and forecasting
fit<-auto.arima(training)
fcast<-forecast(fit,h=2449)
accuracy(fcast,testing)
```





## APPENDIX 5 – Example GP Tree Graphs

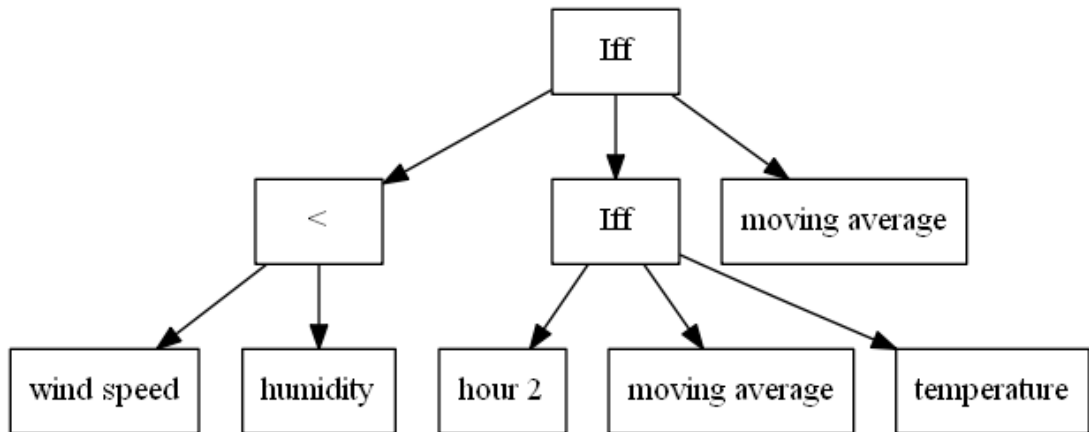


Figure A5.1. SO<sub>2</sub> Tree Graph

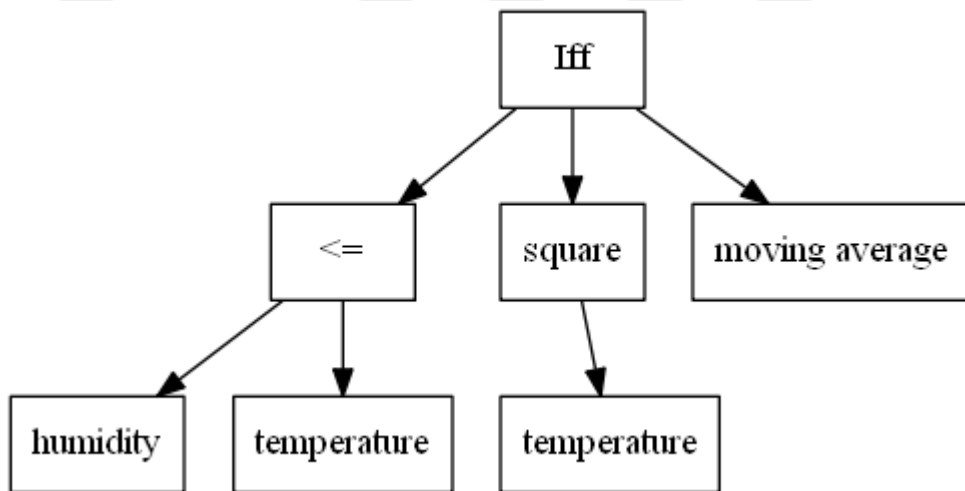


Figure A5.2. PM Tree Graph