SAVITRIBAI PHULE PUNE UNIVERSITY

A PROJECT REPORT ON

Solar Energy Prediction using Machine Learning Algorithms

SUBMITTED TOWARDS THE
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF

BACHELOR OF ENGINEERING
(Information Technology)

BY

Niraj Rathod	Seat No: B120398562
Vishal Patil	Seat No: B120398552
Gajanan Hakare	Seat No:B120398526
Jyotiba Bhosale	Seat No: B120398512

Under The Guidance of

Prof. Suruchi G Dedgaonkar

DEPARTMENT OF INFORMATION TECHNOLOGY
Vishwakarma Institute of Information Technology
Kondhwa(BK),Pune

Vishwakarma Institute of Information Technology
DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Project Entitled

Solar Energy Prediction using Machine Learning Algorithms

Submitted by

Niraj Rathod	Seat No: B120398562
Vishal Patil	Seat No: B120398552
Gajanan Hakare	Seat No:B120398526
Jyotiba Bhosale	Seat No: B120398512

have successfully completed this project report entitled "Solar Energy Prediction using Machine Learning Algorithms", under my guidance in fulfillment of the requirements for the degree of Bachelor of Engineering in Department of Information Technology of University of Pune during the academic year 2015-16.

Prof. Suruchi G Dedgaonkar
Internal Guide
Department of IT

Prof. M S Karyakarte
Head of the Department
Department of IT

External Examiner

Abstract

A challenge with renewable energy prediction is that their power generation is intermittent and uncontrollable. But, prediction of renewable energy is important, because of variation in weather parameters and demand of energy at each location. Nowadays energy consumption is increased as technology is increasing leading to new applications or appliances. But sources are limited, so we need another energy source to fulfil need of energy. Solar energy is infinitely available as sun is source of energy this can use to create energy. The amount of solar radiation varies at every location depending on the weather factors like temperature, rainfall, humidity, wind speed, etc. While manually developing sophisticated prediction models may be feasible for large-scale solar farms, developing tem for distributed generation at millions of homes is a challenging problem. To address the problem, in this project user will select date and select model to create energy prediction or how much amount of energy will create after converting the solar energy in suitable form. We explore automatically creating site-specific prediction models for solar power generation from National Data Centre (NDC) weather forecasts using machine learning techniques.

We have implemented various machine learning algorithms for prediction as linear regression, support vector machine, artificial neural network and ARIMA model. Using authenticated solar data from NDC we are predicting solar radiation for next year. This prediction helps to give data about places where solar plant will be suitable to establish and can be used to its maximum potential.

Acknowledgments

It gives us great pleasure in presenting the preliminary project report on **'Solar Energy Prediction using Machine Learning Algorithms'**.

I would like to take this opportunity to thank my internal guide ***Prof. S G Dedgaonkar*** *for giving me all the help and guidance I needed. I am really grateful to them for their kind support. Their valuable suggestions were very helpful.*

I am also grateful to ***Prof. M S Karyakarte****, Head of Information Technology Department, Vishwakarma Institute of Information Technology for his indispensable support, suggestions.*

In the end our special thanks to all the staff members of the Department of Information Technology of Vishwakarma Institute of Information Technology, Kondhwa (BK), Pune for their valuable time, support, comments, suggestions and persuasion. We would also like to thank the institute for providing the required facilities, Internet access and important books.

Niraj Rathod
Vishal Patil
Gajanan Hakare
Jyotiba Bhosale
(B.E. Information Tehchnology)

June 1, 2016

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

These days energy consumption is increased as technology is increasing leading to new applications or appliances. But sources are limited, so we need another energy source to fulfil need of energy. Solar energy is infinitely available as sun is source of energy.

Solar radiation varies with variation in weather factors like temperature, dew points, humidity, wind speed, etc. Thus it is important to find out the exact factors affecting radiation at a particular location. The correlation between different parameters is found by applying regression techniques. Various machine learning algorithms are applied on the weather dataset. The performance of the algorithms are compared. Finally the efficient algorithm is selected to make the decision about location. The input parameters used in the datasets of the system are solar radiation, temperature, wind speed, cloud, dew, humidity and month.

1.2 Motivation

Renewable energy sources, such as solar and wind, offer many environmental advantages over fossil fuels for electricity generation, but the energy produced by them fluctuates with changing weather conditions. Electric utility companies need accurate forecasts of energy production in order to have the right balance of renewable and fossil fuels available. Errors in the forecast could lead to large expenses for the utility from excess fuel consumption or emergency purchases of electricity from neighboring utilities. Power forecasts typically are derived from numerical weather prediction models, but

statistical and machine learning techniques are increasingly being used in conjunction with the numerical models to produce more accurate forecasts.

1.3 Background

Renewable energy decision makers are required to make critical judgments on a daily basis with regard to energy generation, distribution, demand, storage, and integration. Accurate knowledge of the present and future state of the atmosphere is vital in making these decisions.

Wind and solar energy are among the most difficult weather variables to forecast. Topography, surface roughness, ground cover, temperature inversions, foliage, gravity waves, lowlevel jets, clouds, and aerosols, all affect wind and solar energy prediction skill.

As wind and solar energy portfolios expand, this forecast problem is taking on new urgency because wind and solar energy forecast inaccuracies frequently lead to substantial economic losses and constrain the national expansion of renewable energy. Improved weather prediction and precise spatial analysis of smallscale weather events are crucial for energy management, as is the need to further develop and implement advanced technologies.

The National Center for Atmospheric Research (NCAR), a leader in atmospheric research, development and technology transfer for the past 50 years, is uniquely qualified to support the renewable energy industry in these endeavors. NCAR scientists are already actively engaged with industry decision makers on how best to foresee and respond to short and longterm changes in atmospheric conditions to mitigate risks associated with weather, particularly wind and solar energy prediction.

1.4 Need

To characterize and predict solar radiation to be used as energetic source

Chapter 2

Literature Survey

2.1 R- Language

2.1.1 Why the R-Language

- R is not just a statistics package, its a language.
- R is designed to operate the way that problems are thought about.
- R is both flexible and powerful.

2.1.2 Importance

In this context package has the specific meaning of software that gives you a set number of choices of what to do. It is not at all the same as R package which is discussed later.

Though the distinction between a package and a language is subtle, that subtle difference has a massive impact. With a package you can perform some set number of tasks often with some options that can be varied. A language allows you to specify the performance of new tasks.

Your retort may be, But I wont want to create a new form of regression. Yes, R does allow you to create new forms of regression (and many people have), but R also allows you to easily perform the same sort of standard regression on your 5 datasets (or maybe it is 500 datasets).

The key is abstraction. You easily see that your 5 regressions are really the same there is merely different data involved with each. In your mind you have abstracted the specific tasks so that they all look similar. Once you see the abstraction, it is simple to teach R the abstraction. Languages are all about abstraction.

2.1.3 Why R for the Data Analysis

R is not the only language that can be used for data analysis. Why R rather than another? Here is a list:

- Interactive language
- data structures
- graphics
- missing values
- functions as first class objects
- packages
- community

Data analysis is inherently an interactive process what you see at one stage determines what you want to do next. Interactivity is important. Language is important. The two together an interactive language is even more than their sum. But there is a down-side: compromises between interactive use and programming use are the cause of some user trauma.R has a fantastic mechanism for creating data structures. Obviously if you are doing data analysis, you want to be able to put your data into a natural form. You dont have to warp your data into a particular structure because that is all that is available.

Graphics should be central to data analysis. Humans are predominantly visual, we dont intuitively grasp numbers like we do pictures. It is easy to produce graphs for exploring data. The default graphs can be tweaked to get publication-quality graphs.

Real data have missing values. Missing values are an integral part of the R language. Many functions have arguments that control how missing values

are to be handled.

Functions, like mean and median, are objects that you can use like data. You can easily change your analysis to use the median (or some strange estimate you make up on the spot) rather than the mean.

R has a package system that makes it extremely easy for people to add their own functionality so it is indistinguishable from the central part of R. And people have. There are thousands of packages that do all sorts of extraordinary things.

2.2 Machine Learning Algorithms

2.2.1 Introduction

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions expressed as outputs,:2 rather than following strictly static program instructions. Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

These are various Machine Learning algorithms:

- Artificial neural network
- Bayesian statistics
- Case-based reasoning
- Logistic Model Tree
- Nearest Neighbor Algorithm

- Support vector machines
- Random Forests
- Linear regression
- k-nearest neighbor
- Naive Bayes classifier

2.3 Prediction Model using Machine Learning Algorithms

2.3.1 Data Analysis

We analyze extensive traces of historical data from a weather station to correlate the weather metrics present in the forecast with the solar intensity, in mega joule per square meter, recorded by the weather station. Our analysis quantifies how each forecast parameter affects each other and the solar intensity.

2.3.2 Model Generation

We apply multiple machine learning techniques to derive prediction models for solar intensity using multiple forecast metrics, and then analyze the prediction accuracy of each model. We use machine learning on a training data set of historical solar intensity observations and forecasts to derive a function that computes future solar intensity for a given time horizon from a set of forecasted weather metrics. We formulate models based on linear least squares regression, as well as support vector machines (SVM). We find that SVM with radial basis function kernels built using historical data from seven weather metrics is more accurate than existing forecast-based models that use only sky condition for predictions and is better than simple approaches that only use the past to predict the future.

Fig. 2.2 shows how the day of the year affects solar intensity by charting the average solar intensity reading at noon per day over our 10 month monitoring period where day zero is January 1st, 2010. As expected, the graph shows that the solar intensity is lowest in January near the winter solstice and increases into the summer before decreasing after the vernal equinox.

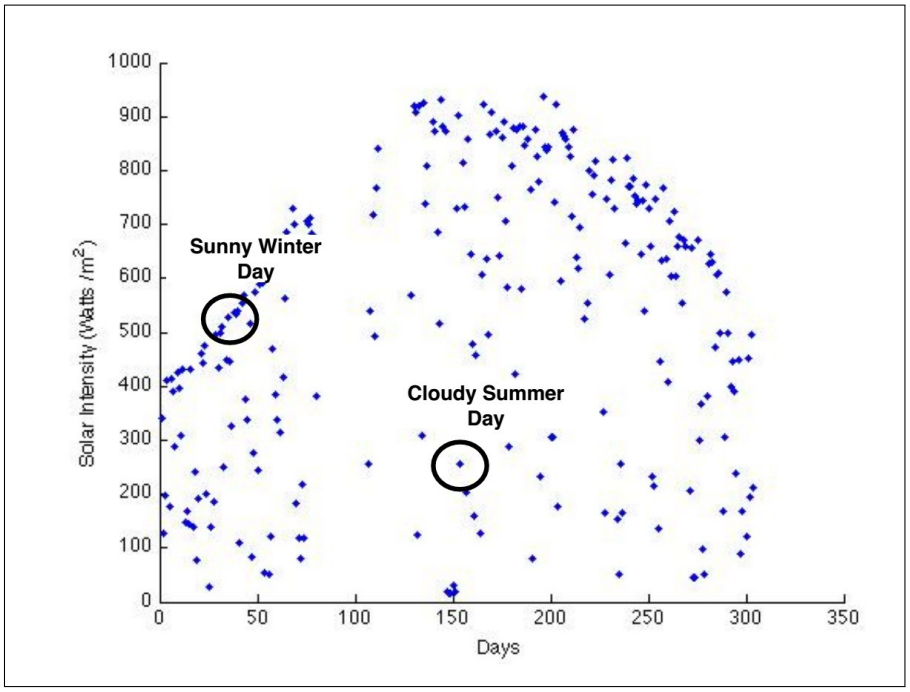


Figure 2.1: Solar intensity shows seasonal variation with days of a year, although daily weather conditions also have a significant impact

Additionally, the graph also implies that other conditions also have a significant impact on solar intensity, since many days throughout the spring and summer have low solar intensity readings. The graph shows that solar intensity and the day of the year are roughly correlated: most of the time, but not always, a summer day will have a higher solar intensity than a winter day. However, other factors must contribute to the solar intensity, since there are clearly some sunny winter days that record higher solar intensity readings than some cloudy summer days. To better understand correlations with other weather metrics, we model similar relationships for the other forecast metrics.

For example, Fig. 2.2,2.3,2.4 show that wind speed, dew point, and temperature are not highly correlated with solar intensity. Solar intensity varies almost uniformly from lower to higher values at any value of wind speed (a).

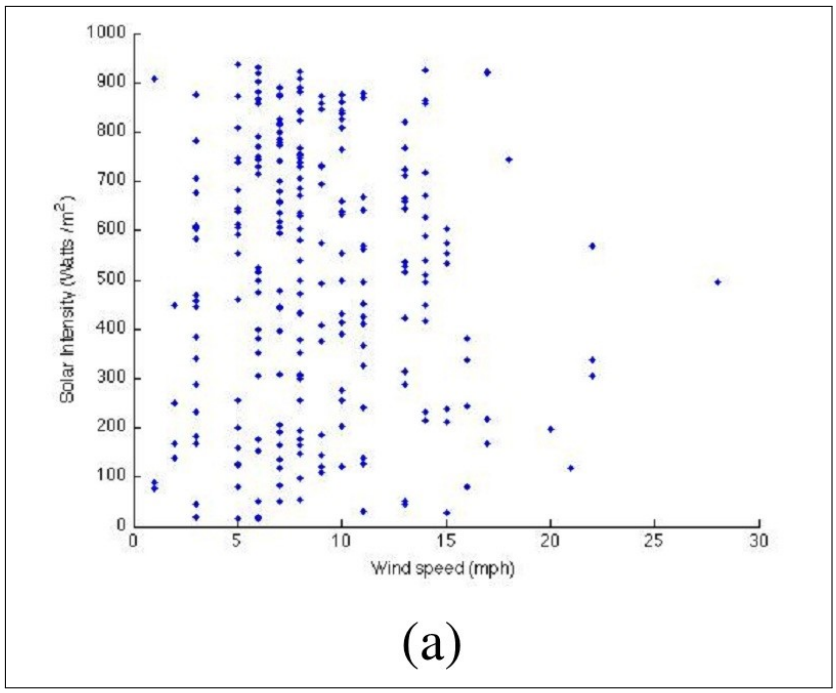


Figure 2.2: Solar intensity and wind speed show little correlation

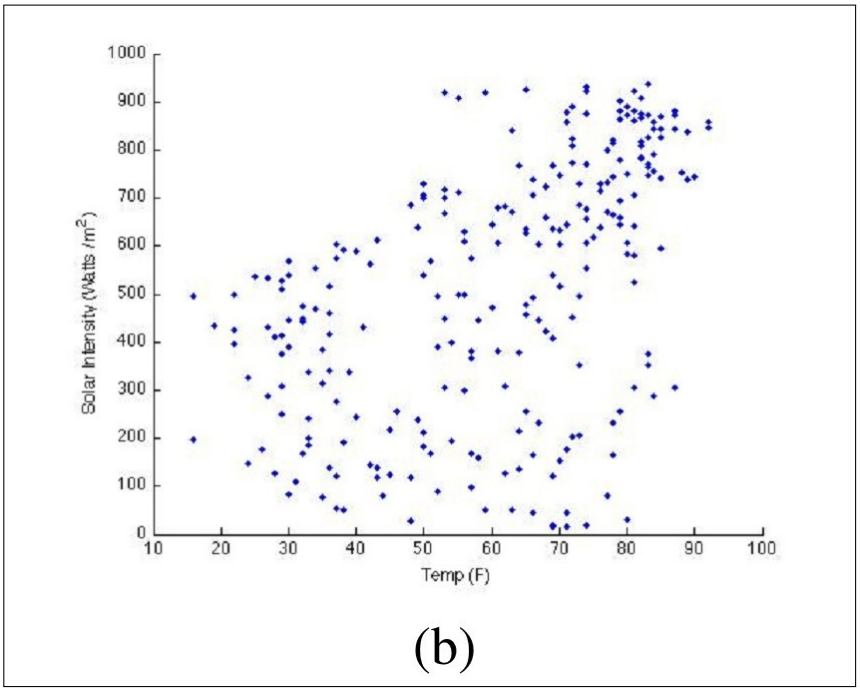


Figure 2.3: Solar intensity shows some correlation with temperature at high temperature

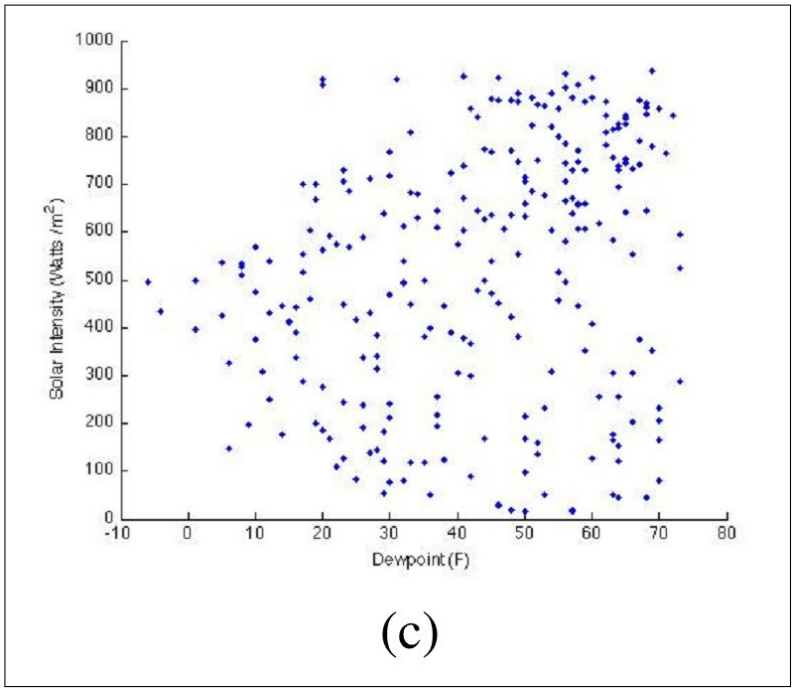


Figure 2.4: Solar intensity shows some correlation with dew point at high dew points

Thus, wind speed has nearly zero correlation with solar intensity and its value is not indicative of the solar intensity or solar panel power generation. Both temperature (b) and dew point (c) correlate with solar intensity at higher values: if the temperature or dew point is high, then the solar intensity is likely to be high. However, if the temperature or dew point is low, the solar intensity exhibits a more significant variation between high and low values. The results are intuitive. For example, in the summer a high temperature is often dependent on sunlight, while in the winter sunlight contributes less in raising the ambient temperature.

In contrast, Fig. 2.5,2.6,2.7 shows that sky cover, relative humidity, and chance of precipitation have high negative correlations with solar intensity. In each case, as the value of the metric increases, the solar intensity reading generally decreases. However, as with the day of the year, there must

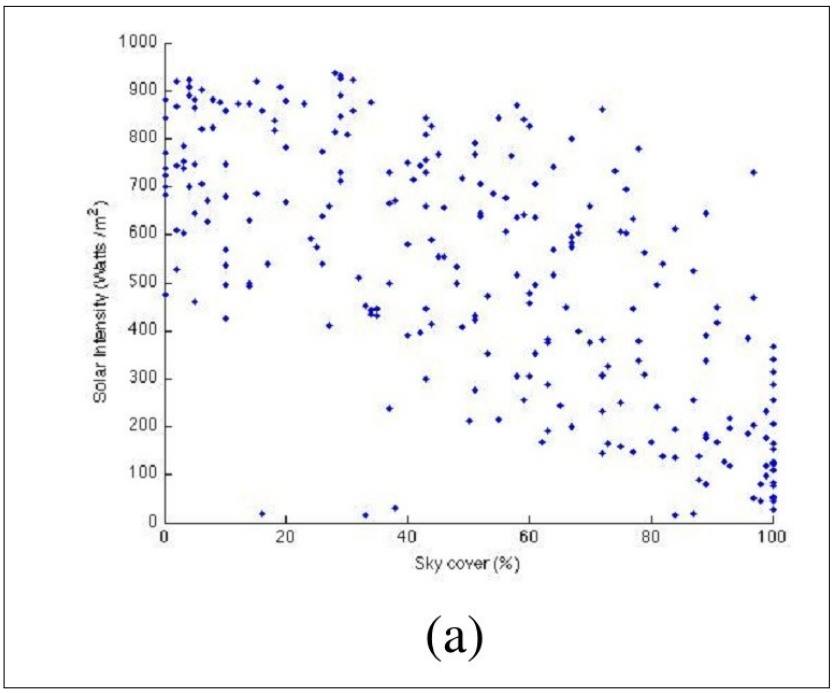


Figure 2.5: Solar intensity generally decreases with increasing values of sky cover

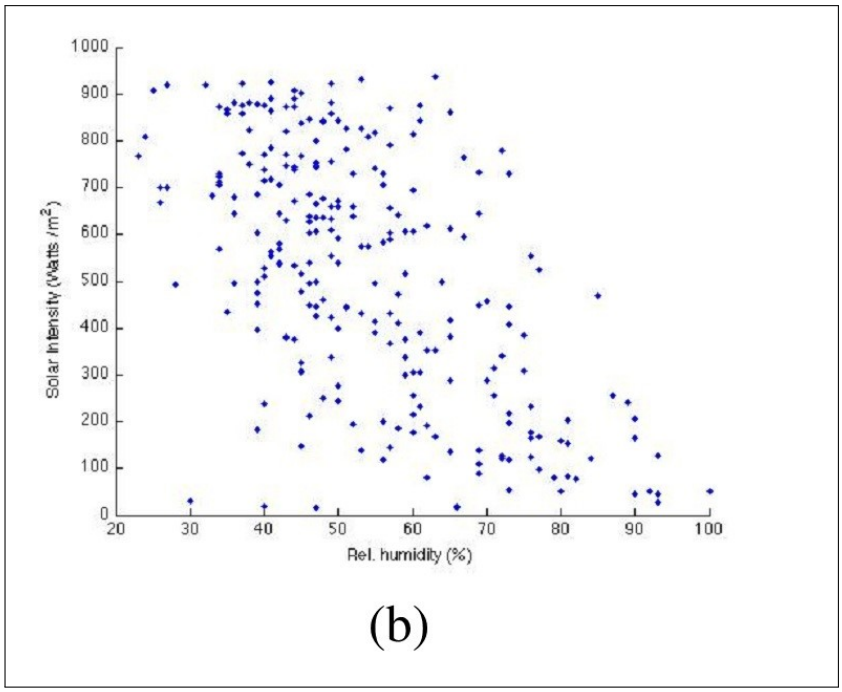


Figure 2.6: Solar intensity generally decreases with relative humidity

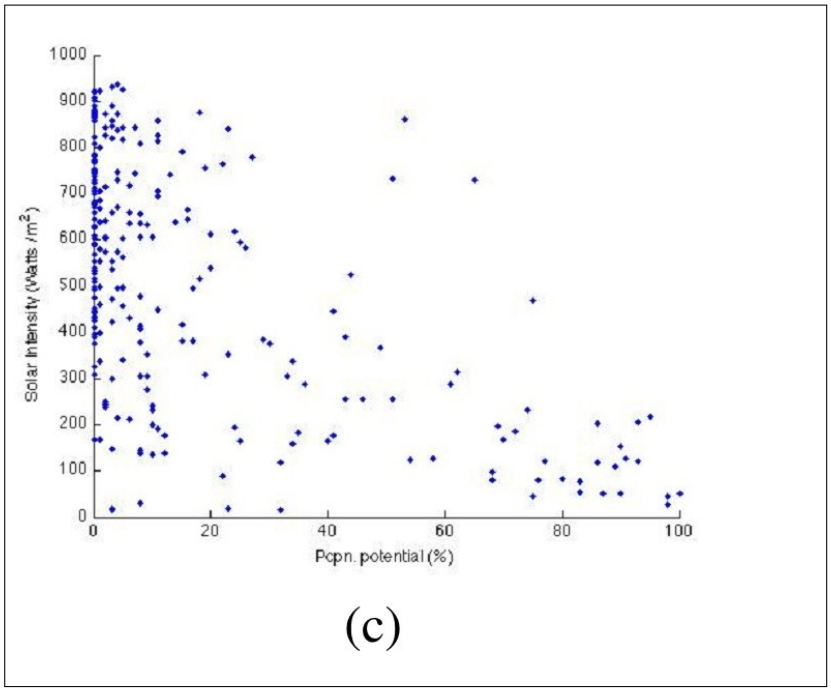


Figure 2.7: Solar intensity generally decreases with precipitation potential

be other factors that contribute to the solar intensity reading, since there are some days with a high sky cover, relative humidity, and precipitation probability, but a high solar intensity reading and vice versa. In addition to exhibiting complex relationships with solar intensity, each weather metric also exhibits a complex relationship with other weather metrics.

	Day	Temp.	Dew	Wind	Sky cover	Pcpn.	Humidity
Day	1.0000	0.7022	0.7007	-0.0711	-0.1034	-0.0571	0.0645
Temp.	0.7022	1.0000	0.9212	-0.1994	-0.2582	-0.1279	-0.0791
Dew point	0.7007	0.9212	1.0000	-0.2251	0.0455	0.1491	0.3081
Wind	-0.0711	-0.1994	-0.2251	1.0000	-0.0192	0.0340	-0.1025
Sky cover	-0.1034	-0.2582	0.0455	-0.0192	1.0000	0.7067	0.7525
Precipitation	-0.0571	-0.1279	0.1491	0.0340	0.7067	1.0000	0.7475
Humidity	0.0645	-0.0791	0.3081	-0.1025	0.7525	0.7475	1.0000

Table 2.1 Correlation Matrix Showing Correlation Between Different Forecast Parameters.

Table 2.1 shows correlation coefficients for each weather metric using the Pearson product-moment correlation coefficient, which divides the covariance of the two variables by the product of their standard deviations. The higher

the absolute value of the correlation coefficient, the stronger the correlation between the two weather metrics a positive correlation indicates an increasing linear relationship, while a negative correlation indicates a decreasing linear relationship. The complex relationships between weather metrics and solar intensity shown in this table motivate our study of automated prediction models using machine learning techniques in the next section.

2.4 Machine Learning Algorithm

2.4.1 Linear Regression Algorithm

Linear least squares regression is by far the most widely used modeling method. It is what most people mean when they say they have used "regression", "linear regression" or "least squares" to fit a model to their data. Not only is linear least squares regression the most widely used modeling method, but it has been adapted to a broad range of situations that are outside its direct scope. It plays a strong underlying role in many other modeling methods, including the other methods discussed in this section: nonlinear least squares regression, weighted least squares regression and LOESS.

2.4.1.1 Definition of Linear Regression Model

Used directly with an appropriate data set, linear least squares regression can be used to fit the data with any function of the form

F(x:$\beta = \beta 0 + \beta 1x1 + \beta 2x2 + ...$)[3]
In which

1. Each explanatory variable in the function is multiplied by an unknown parameter,
2. There is at most one unknown parameter with no corresponding explanatory variable, and
3. All of the individual terms are summed to produce the final function value.

In statistical terms, any function that meets these criteria would be called a "linear function". The term "linear" is used, even though the function may not be a straight line, because if the unknown parameters are considered to be variables and the explanatory variables are considered to be known coefficients corresponding to those "variables", then the problem becomes a system (usually over determined) of linear equations that can be solved for the values of the unknown parameters. To differentiate the various meanings of the word "linear", the linear models being discussed here are often said to be "linear in the parameters" or "statistically linear".

2.4.1.2 Advantages of Linear Regression Method

1. Linear least squares regression has earned its place as the primary tool for process modeling because of its effectiveness and completeness.

2. Though there are types of data that are better described by functions that are nonlinear in the parameters, many processes in science and engineering are well-described by linear models. This is because either the processes are inherently linear or because, over short ranges, any process can be well-approximated by a linear model.

3. The estimates of the unknown parameters obtained from linear least squares regression are the optimal estimates from a broad class of possible parameter estimates under the usual assumptions used for process modeling. Practically speaking, linear least squares regression makes very efficient use of the data. Good results can be obtained with relatively small data sets.

4. Finally, the theory associated with linear regression is well-understood and allows for construction of different types of easily-interpretable statistical intervals for predictions, calibrations, and optimizations. These statistical intervals can then be used to give clear answers to scientific and engineering questions.

2.4.1.3 Disadvantages of Linear Regression Method

1. The main disadvantages of linear least squares are limitations in the shapes that linear models can assume over long ranges, possibly poor extrapolation properties, and sensitivity to outliers.

2. Linear models with nonlinear terms in the predictor variables curve relatively slowly, so for inherently nonlinear processes it becomes increasingly difficult to find a linear model that fits the data well as the range of the data increases. As the explanatory variables become extreme, the output of the linear model will also always more extreme. This means that linear models may not be effective for extrapolating the results of a process for which data cannot be collected in the region of interest. Of course extrapolation is potentially dangerous regardless of the model type.

3. Finally, while the method of least squares often gives optimal estimates of the unknown parameters, it is very sensitive to the presence of unusual data points in the data used to fit a model. One or two

outliers can sometimes seriously skew the results of a least squares analysis. This makes model validation, especially with respect to outliers, critical to obtaining sound answers to the questions motivating the construction of the model.

2.4.2 Support Vector Machine (SVM)

Support Vector Machine can be applied not only to classification problems but also to the case of regression. Still it contains all the main features that characterize maximum margin algorithm: a non-linear function is leaned by linear learning machine mapping into high dimensional kernel induced feature space. The capacity of the system is controlled by parameters that do not depend on the dimensionality of feature space.

In the same way as with classification approach there is motivation to seek and optimize the generalization bounds given for regression. They relied on defining the loss function that ignores errors, which are situated within the certain distance of the true value. This type of function is often called epsilon intensive loss function. The figure below shows an example of one-dimensional linear regression function with epsilon intensive band. The variables measure the cost of the errors on the training points. These are zero for all points that are inside the band.[6]

Support vector regression was compared to bagging and a feature space representation on four nonlinear problems. On three of these problems a feature space representation was best, bagging was worst, and SVR came in second. On the fourth problem, Boston Housing, SVR was best and we were unable to construct a feature space representation because of the high dimensionality required of the feature space.On linear problems,forward subset selection seems to be the method of choice for the two linear problems we tried at varying signal to noise ratios. In retrospect, the problems we decided to test on were too simple.SVR probably has greatest use when the dimensionality of the input space and the order of the approximation creates a dimensionality of a feature space representation much larger than that of the number of examples. This was not the case for the problem we considered. We thus need real life examples that fulfill these requirements.[5]

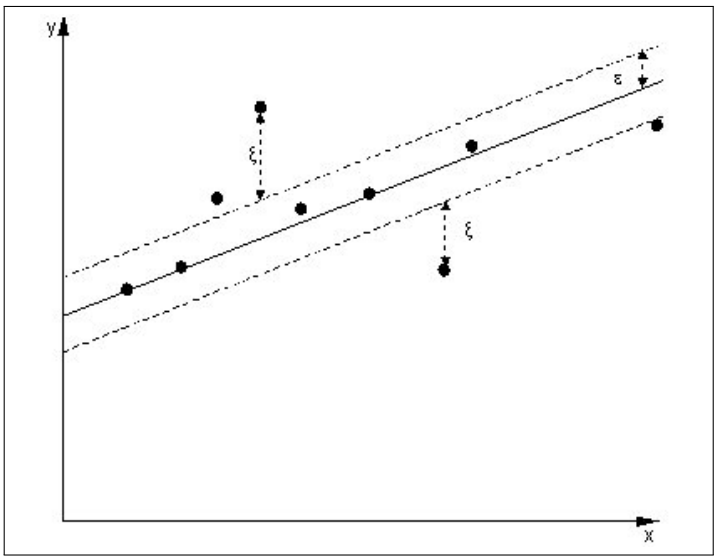


Figure 2.8: One-dimensional linear regression with epsilon intensive band

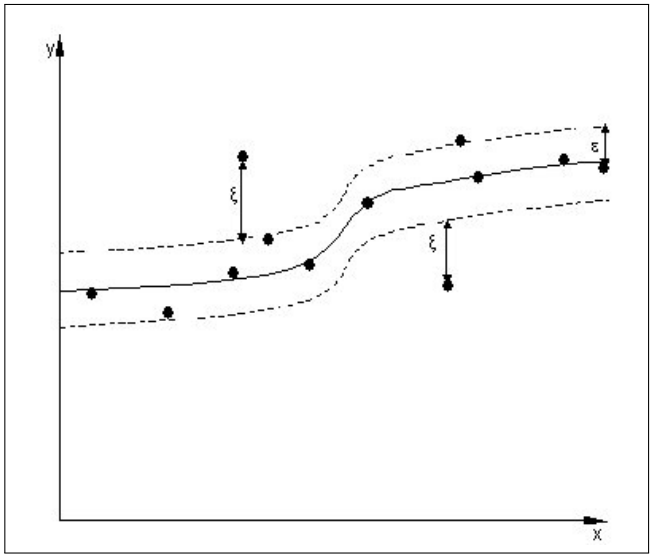


Figure 2.9: Non-linear regression function

Another picture shows a similar situation but for non-linear regression case.

One of the most important ideas in Support Vector Classification and Regression cases is that presenting the solution by means of small subset of training points gives enormous computational advantages. Using the epsilon intensive loss function we ensure existence of the global minimum and at the same time optimization of reliable generalization bound.

2.4.2.1 Advantages of SVM

The Support Vector Machine (SVM) classifier is a powerful classifier that works well on a wide range of classification problems, even problems in high dimensions And that are not linearly separable.

2.4.2.2 Disadvantages of SVM

1. Perhaps the biggest limitation of the support vector approach lies in choice of the kernel.

2. A second limitation is speed and size, both in training and testing

3. However, from a practical point of view perhaps the most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements of the required quadratic programming in large-scale tasks.

2.4.3 Artificial Neural Network (ANN)

Backpropagation Algorithm

In machine learning and cognitive science, artificial neural networks (ANNs) are a family of models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) which are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

We are using back propagation algorithm for prediction here. Back propagation, an abbreviation for "backward propagation of errors", is a common

method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

Back propagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as auto encoders. It is a generalization of the delta rule to multi-layered feed forward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Back propagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.

The goal and motivation for developing the back propagation algorithm was to find a way to train a multi-layered neural network such that it can learn the appropriate internal representations to allow it to learn any arbitrary mapping of input to output.[11]

2.4.3.1 Advantages of ANN

1. Relatively simple implementation.
2. Standard method and generally works well.
3. Mathematical Formula used in algorithm can be applied to any network.
4. Computing time is reduced if the weights chosen are small at the beginning.
5. Batch update of weights exists, which provides a smoothing effect on the weight correction terms.

2.4.3.2 Disadvantages of ANN

1. Slow and inefficient. Can get stuck in local minima resulting in sub-optimal solutions.

2. A large amount of input/output data is available, but youre not sure how to relate it to the output.

2.4.4 ARIMA (Autoregressive Integrated Maving Average)

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. These models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

They are applied in some cases where data show evidence of non-stationary, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationary.

Non-seasonal ARIMA models are generally denoted *ARIMA(p,d,q)* where parameters p, d, and q are non-negative integers, is the order of the Autoregressive model, is the degree of differencing, and is the order of the Moving-average model.

Seasonal ARIMA models are usually denoted *ARIMA(p,d,q)(P,D,Q)m* , where refers to the number of periods in each season, and the uppercase refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model. ARIMA models form an important part of the Box-Jenkins approach to time-series modelling.

When two out of the three terms are zeros, the model may be referred to base on the non-zero parameter, dropping "AR", "I" or "MA" from the acronym describing the model. For example, ARIMA (1,0,0) is AR(1), ARIMA(0,1,0) is I(1), and ARIMA(0,0,1) is MA(1).

Definition
A statistical analysis model that uses time series data to predict future trends. It is a form of regression analysis that seeks to predict future movements along the seemingly random walk taken by stocks and the financial market by examining the differences between values in the series instead of using the actual data values. Lags of the differenced series are referred to as "autoregressive" and lags within forecasted data are referred to as "moving average.

Chapter 3

Project Statement

3.1 Problem Definition

To create an application for solar energy prediction using various machine learning algorithms. System focuses on analysing amount of energy production. It is easy to generate analysis report.

3.2 Objectives

- Creating automatic prediction model.
- Predicting future solar intensity.
- Make better decisions based on highly accurate information.

3.3 What Is To Developed

Predicting solar generation from whether forecasts using machine learning:
We compare multiple machine learning algorithms for generating prediction models, including linear least squares, support vector machines (SVM) using multiple kernel functions, Artificial Neural Network (ANN) and ARIMA model. We evaluate the accuracy of each model using historical NDC forecasts and solar intensity readings from a weather station deployment for nearly a year.
These are following Algorithm Steps:

1. Obtained Data are used as input.

2. Map generation Data Analysis using R-language in R-Studio.
3. Generate Model using regression techniques.
4. Generate correlation matrix table by calculating standard deviation covariance of two variables.
5. Apply linear least squares regression method to predict solar intensity.
6. Apply SVM algorithm to predict solar intensity.
7. Apply ANN back propagation algorithm.
8. Apply ARIMA model is used for weather forecasting.

Step I: Obtained Dataset are used as input.

	Months	Total.Daily.Radiation	Temprature	Humidity
1	Jan	12.68	24.84	87.90
2	Feb	16.65	30.40	77.46
3	Mar	18.67	39.54	56.32
4	Apr	20.63	38.36	47.90
5	May	20.90	39.17	56.48
6	Jun	18.49	32.45	66.20
7	Jul	12.02	29.31	75.61
8	Aug	12.13	28.33	87.83
9	Sep	14.72	29.58	81.76
10	Oct	14.82	30.99	79.33
11	Nov	13.81	27.87	67.13
12	Dec	12.81	28.79	83.54

Figure 3.1: Sample Dataset

Step II: Map generation data analysis using R-Language

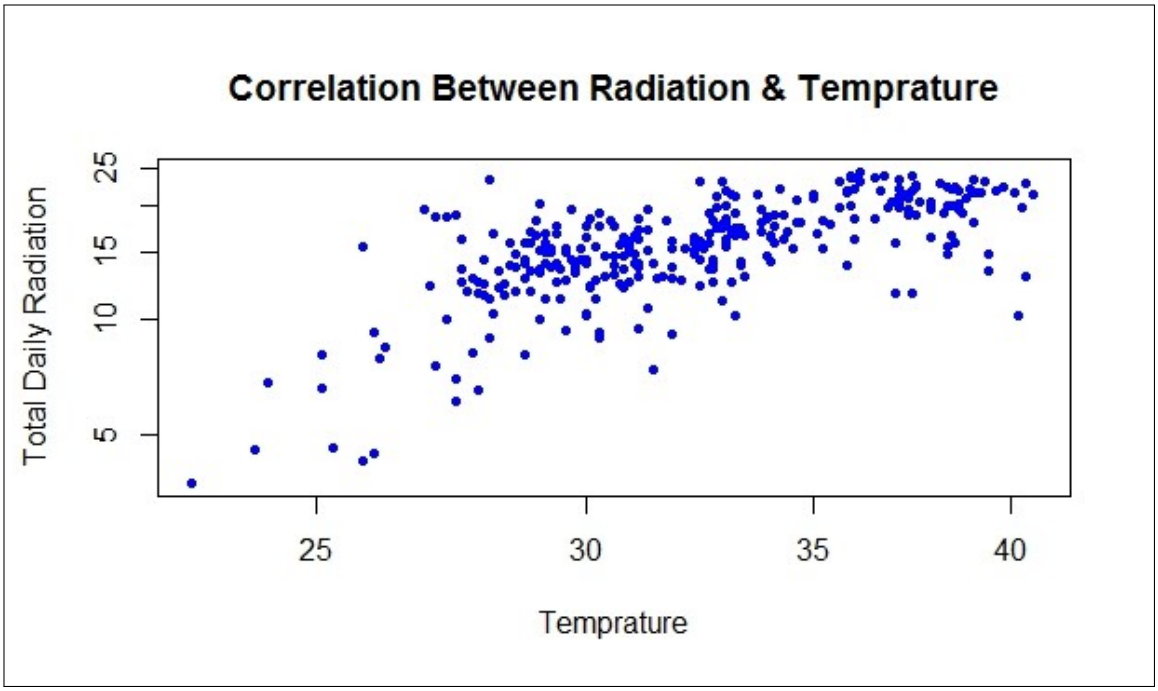


Figure 3.2: Daily radiation shows some correlation with temperature

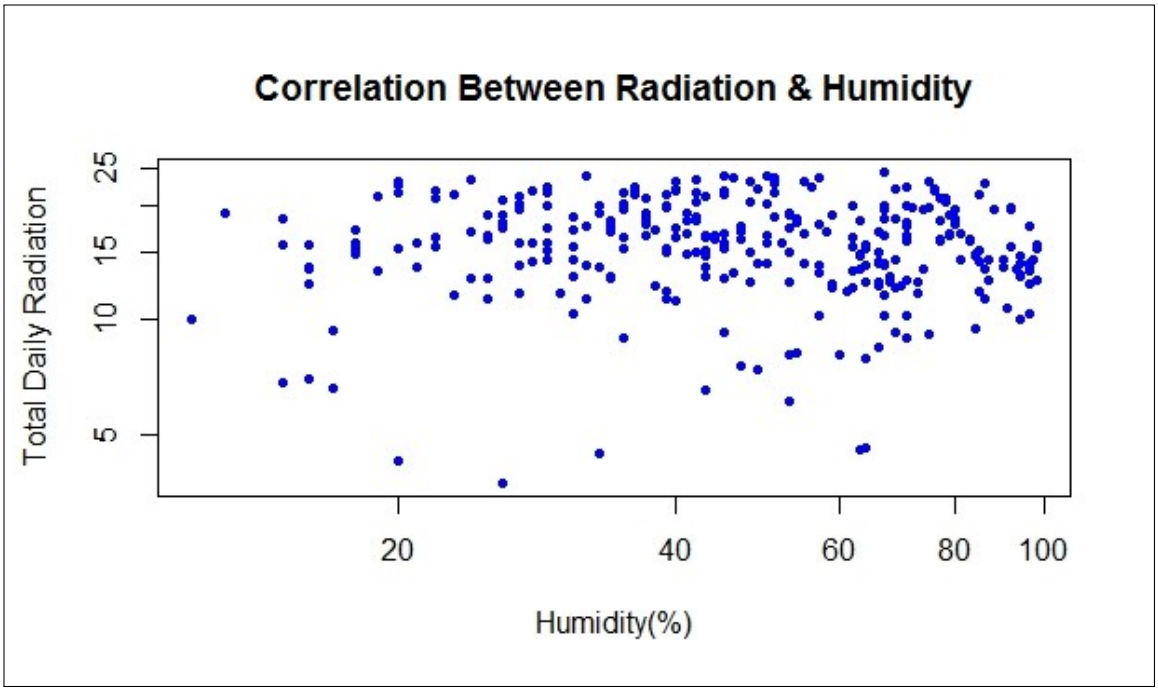


Figure 3.3: Daily radiation shows some correlation with humidity

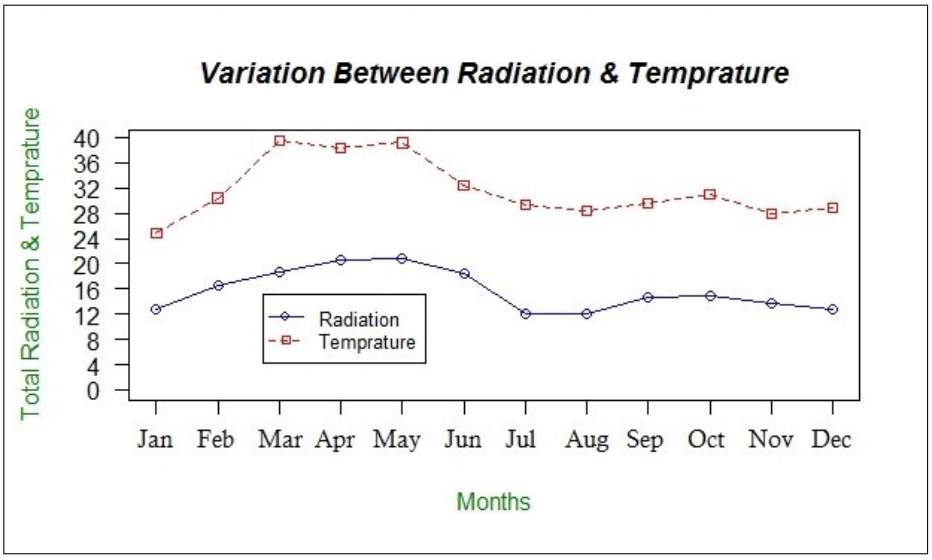


Figure 3.4: Shows that when radiation increases then temperature also increases so its directly proportional to each other.

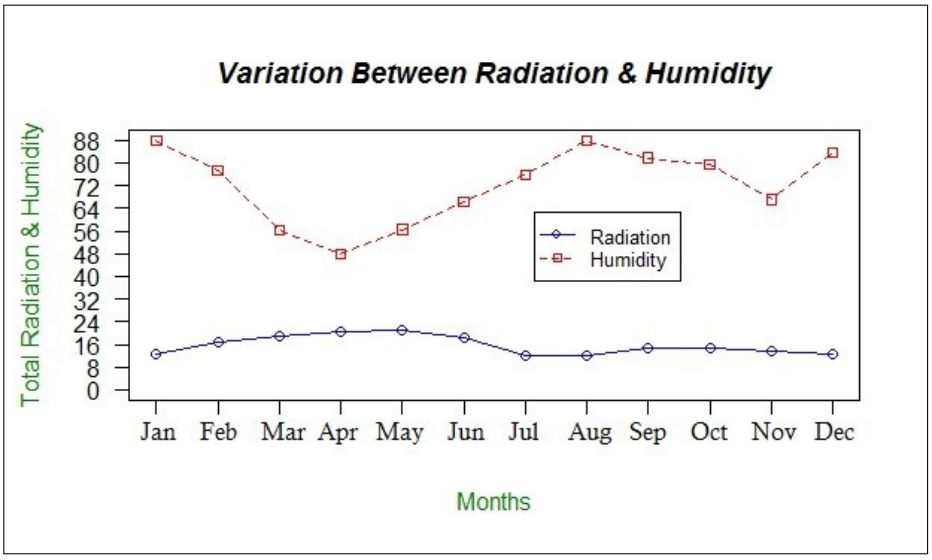


Figure 3.5: Shows that when radiation increases then humidity is decreases so its inversely proportional to each other.

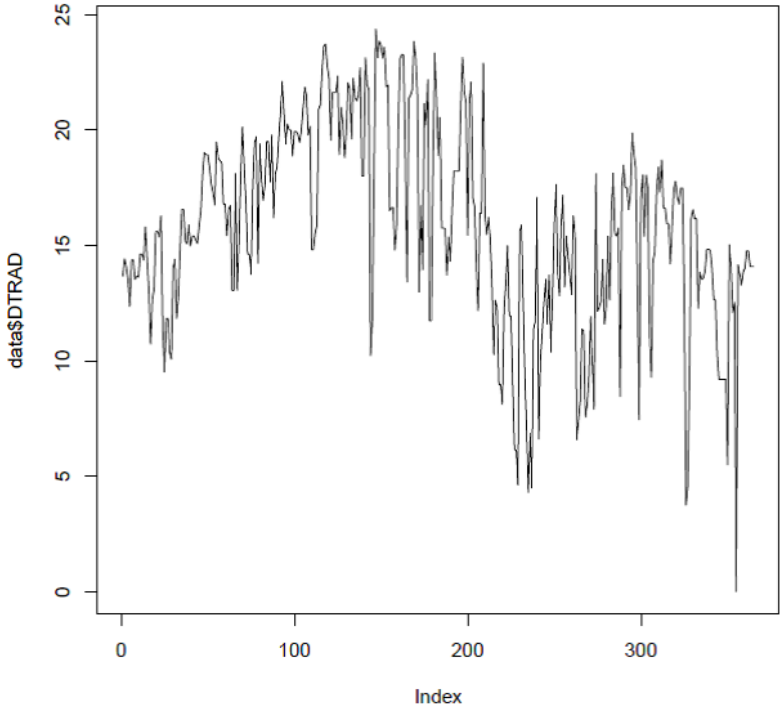
Step III: Generate correlation matrix table by calculating standard deviation covariance of two variables.

Table 3

	Days	AW	RH AVG	DPT AVG	TC AVG	TEMP	DTRAD
Days	1						
AW	0.046016	1					
RH AVG	-0.21174	-0.0656	1				
DPT AVG	-0.03581	0.026333	0.673691	1			
TC AVG	-0.15852	-0.13606	0.552019	0.6717994	1		
TEMP	0.578412	0.350137	-0.37548	-0.015422	-0.20911	1	
DTRAD	0.306473	0.316587	-0.695	-0.518749	-0.71033	0.431339	1

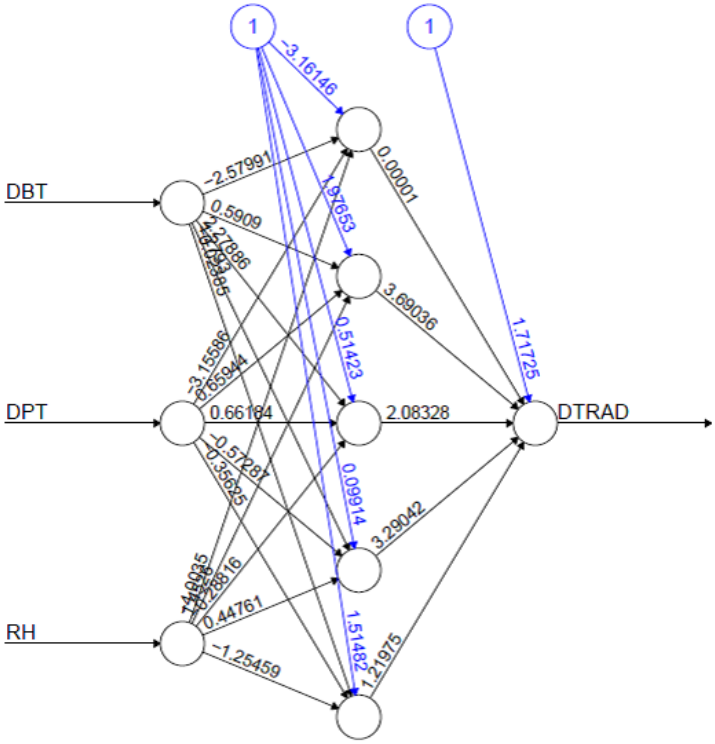
Table 3 Correlation Matrix Showing Correlation Between Different Forecast Parameters.

Step IV: Apply linear least squares regression method to predict solar intensity.
Solar Radiation = F (Days, Temperature, Dew Point, Wind Speed, Total Amount of Cloud, Humidity) Solar Radiation = 37.9585 - 0.3959 * Temperature 0.0401 * Humidity + 0.1723 * Wind Speed 1.4512* Amt. of Cloud - 0.0461 * Days + 0.2091 * Dew Points.
Step V: Apply SVM algorithm to predict solar intensity.



SVM generated graph daily radiation vs. index

Step VI: Apply ANN algorithm to predict solar radiation.



Step VI: Apply ARIMA model is used for weather forecasting.

Chapter 4

Design Model

4.1 Software Requirement Specification(SRS)

4.1.1 Purpose

The purpose of the Software Requirements Specification (SRS) document is to describe the external behavior of the Solar-GIS. It defines and describes the various operations of the system, what the system will do, how the system will react to external stimuli and under what constraints will it function. The document also describes how the user will interact with the system through the user interfaces, and the various design constraints under which the system was designed. It serves as system documentation between both the stakeholders and the developer of the system. Hence, we can say that the Software Requirements Specification (SRS) helps define the basic requirements of the system, along with its constraints, and the functionality expected out of the system, as a whole.

4.1.2 Scope

The Software Requirements Specification captures all the requirements in a single document.

Solar radiation varies with variation in weather factors like temperature, dew points, humidity, wind speed, etc. Thus it is important to find out the exact factors affecting radiation at a particular location. The correlation between different parameters is found by applying regression techniques. Various machine learning algorithms are applied on the weather dataset. The performance of the algorithms are compared. Finally the efficient algorithm is selected to make the decision about location. The input parameters used

in the datasets of the system are solar radiation, temperature, wind speed, cloud, dew, humidity and month. So we predict future solar radiation using various machine learning algorithm.

4.1.3 Overview

The SRS will provide a detailed description of the solar energy resource planning using machine learning algorithm. This document will provide the outline of the requirements, overview of the characteristics and constraints of the system.

Section 1: This section will provide the basic outline of the system. It describes the various elements of the system such as product perspective, constraints, user characteristics, functionalities , and the various assumptions and dependencies. This section basically describes the informal requirements of the system and provides a context for the next section.

Section 2: This section of SRS contains the technical details to provide the basic functionality of the system. It is primarily written for the developers of the system. It describes elements such as the functionality, performance and the user interfaces of the system.

4.1.4 Hardware Interface

Specifications	particulars
OS	Windows 7
CPU	Core i3
RAM	4 GB
Storage	500 GB

4.1.5 Software Requirement

- R Studio, R - Language
- Visual Studio 2012
- Microsoft Excel 2013

4.2 System Architecture & UML Diagrams

4.2.1 Block Diagram

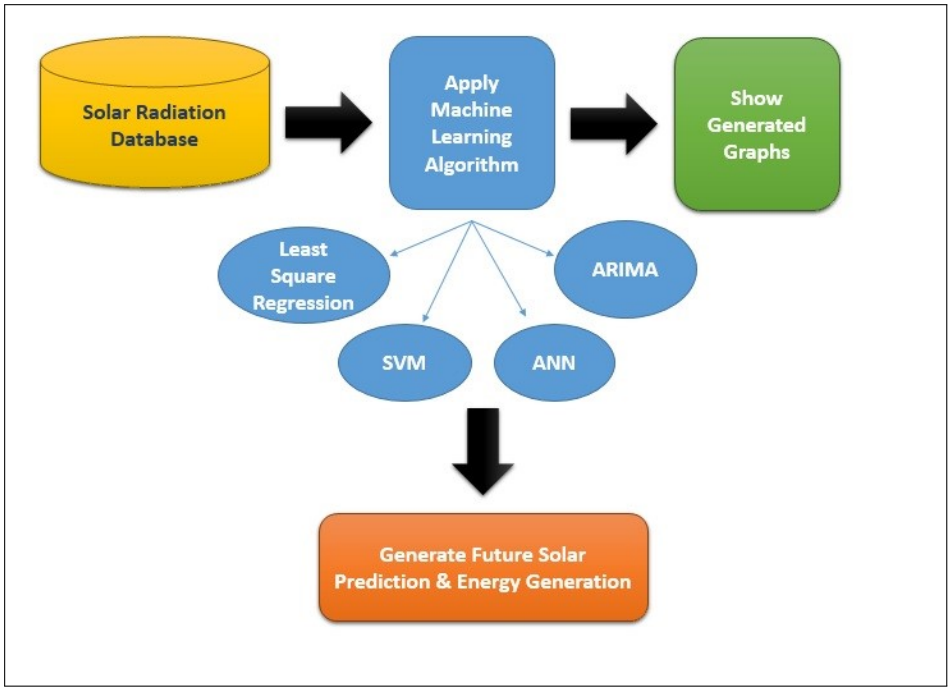


Figure 4.1: Block Diagram

4.2.2 Usecase Diagram

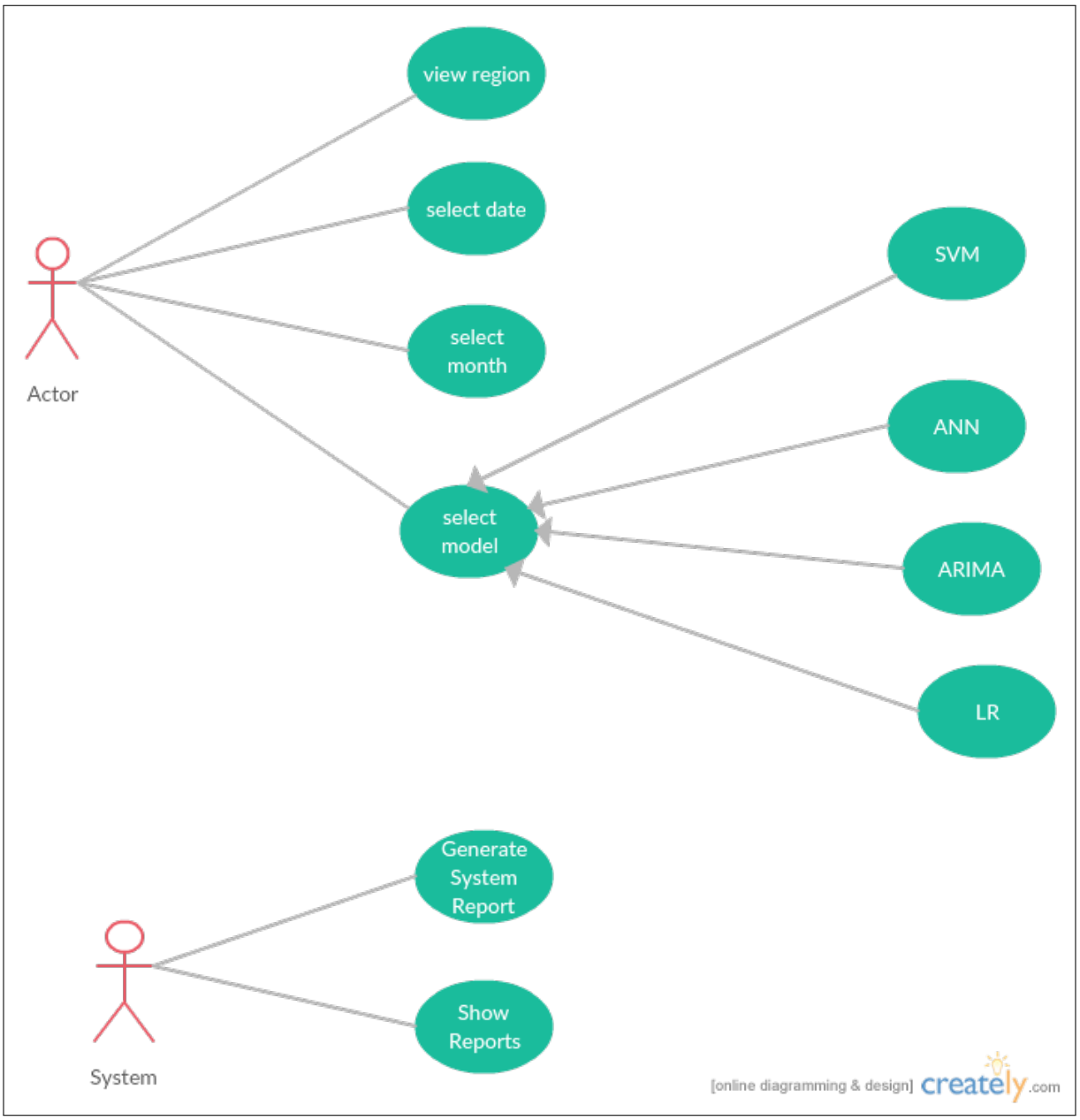


Figure 4.2: Usecase Diagram

4.2.3 Class Diagram

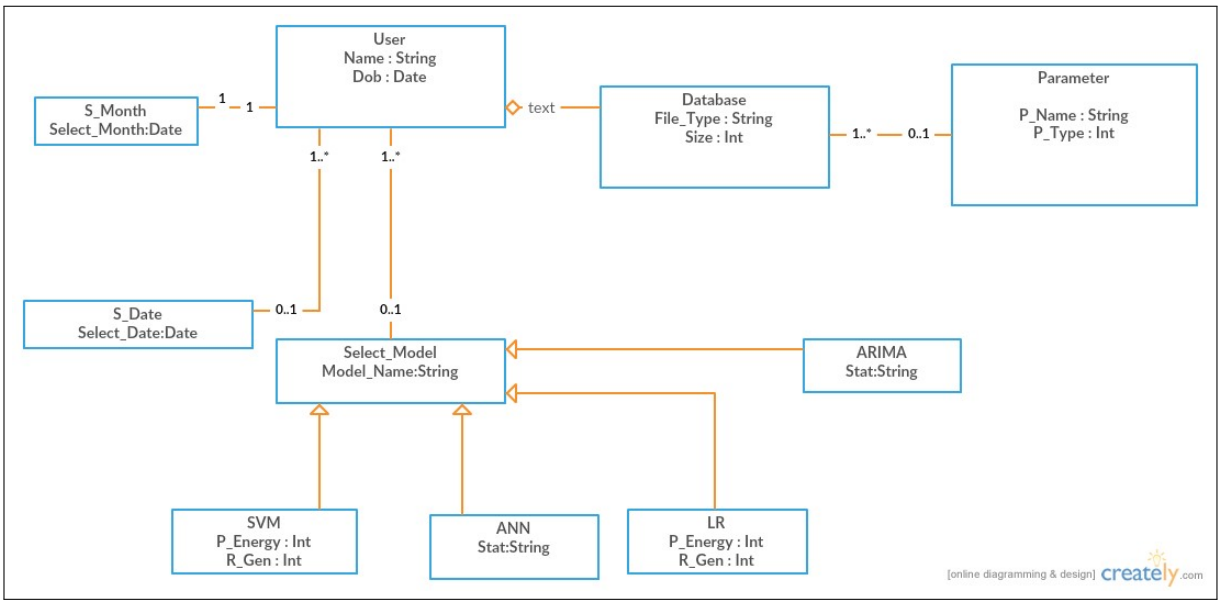


Figure 4.3: Class Diagram

4.2.4 Snapshots

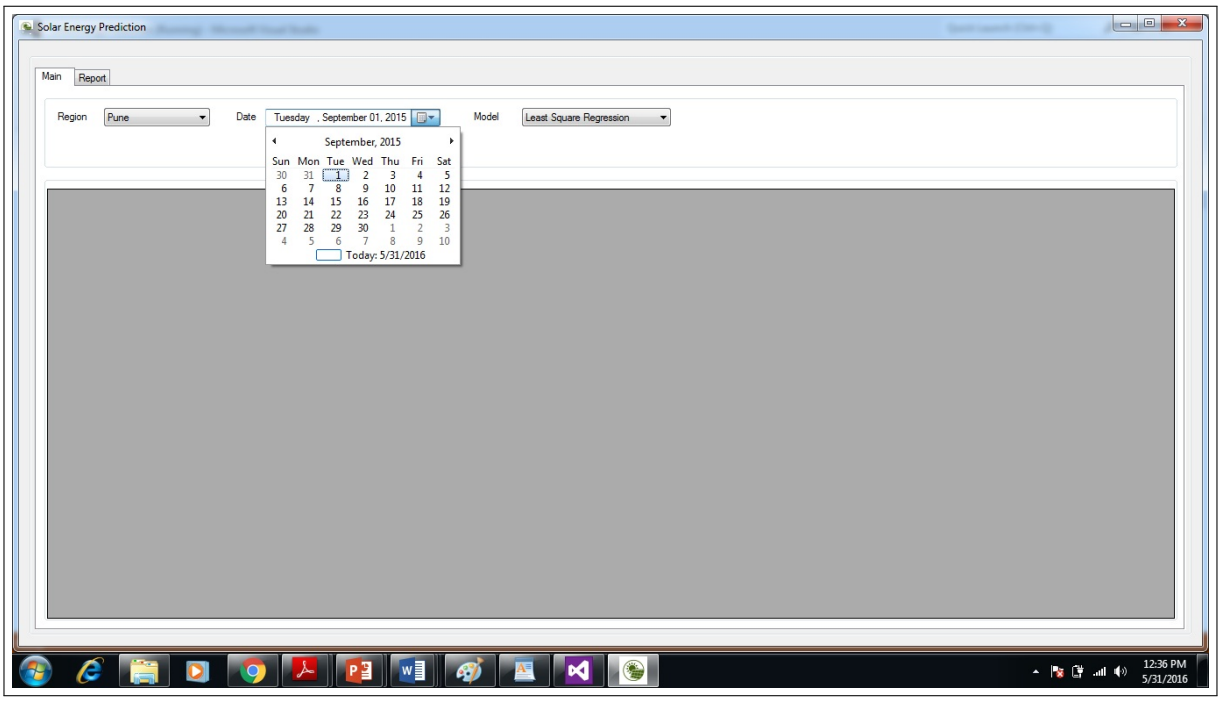


Figure 4.4: Snapshot 1

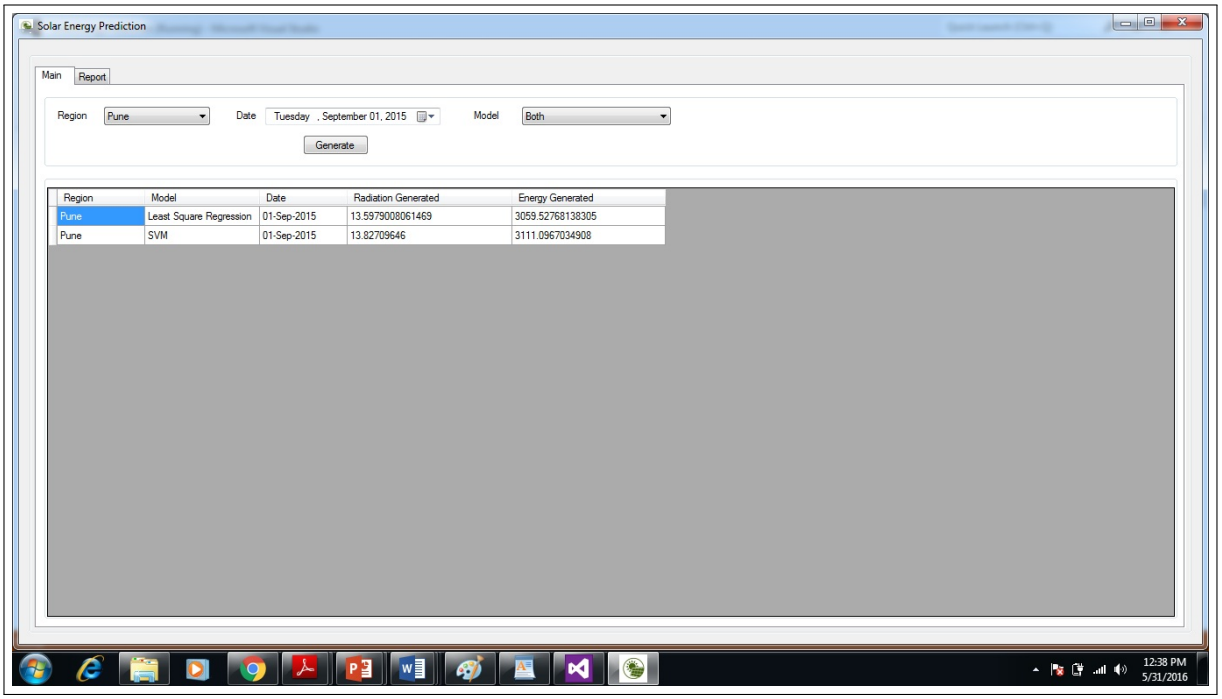


Figure 4.5: Snapshot 2

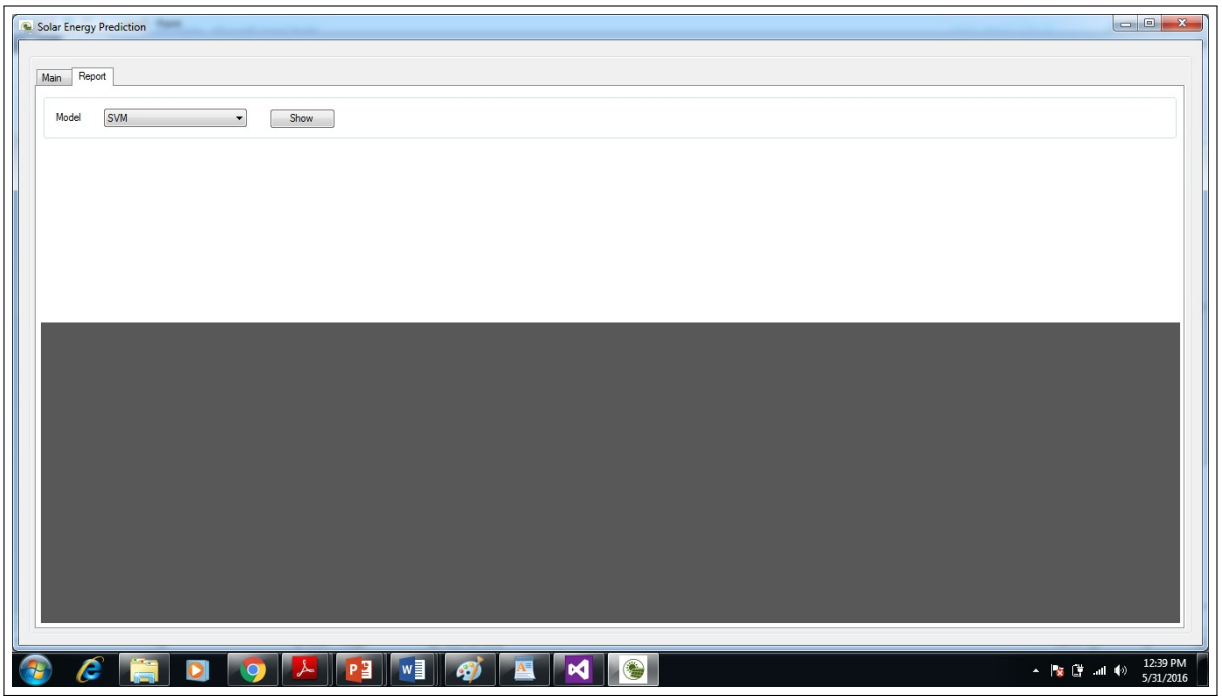


Figure 4.6: Snapshot 3

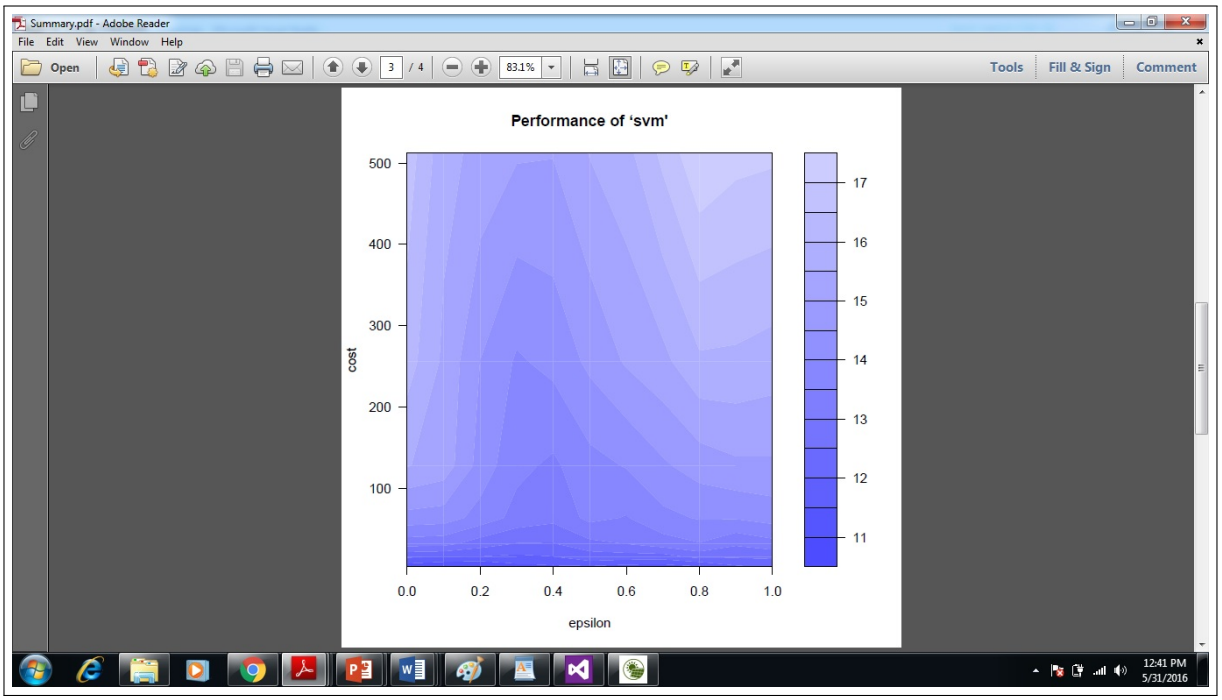


Figure 4.7: Snapshot 4

Chapter 5

Research Methodology

5.1 Explaination with Examples

5.1.1 Linear Regeression

Linear Least square regression is a method for finding a line that summarizes the relationship between the two variables, at least within the domain of the explanatory variable x.
Equation for it is:

$$Y = a + b\,X$$

where

$$b = r\,\frac{SDy}{SDx}$$

$$a = \overline{Y} - b\overline{X}$$

Another formula for Slope:
Slope = $(N\Sigma XY - (\Sigma X)(\Sigma Y))/(N\Sigma X2 - (\Sigma X)2)$
where,
b = the slope of the regression line
a = the intercept point of the regression line and the y axis.
X = Mean of x value
Y = Mean of y values
SDx = Standard Deviation of x
SDy = Standard Deviation of y

X Values	Y Values
60	3.1
61	3.6
62	3.8
63	4
65	4.1

To Find,
Least Square Regression Line Equation
Solution:
Step 1: Count the number of given x values.
N=5

Step 2: Find XY, X2 for the given values. See the below table.

X Value	Y Value	X*Y	X*X
60	3.1	60 * 3.1 = 186	60 * 60 = 3600
61	3.6	61 * 3.6 = 219.6	61 * 61 = 3721
62	3.8	62 * 3.8 = 235.6	62 * 62 = 3844
63	4	63 * 4 = 252	63 * 63 = 3969
65	4.1	65 * 4.1 = 266.5	65 * 65 = 4225

Step 3: Now, Find $\Sigma X, \Sigma Y, \Sigma\Sigma Y, \Sigma X2$
$for the values \Sigma X = 311 \Sigma Y = 18.6 \Sigma XY = 1159.7 \Sigma X2 = 19359$

Step 4:$Substitute the values in the above slope formula given.$
$Slope(b) = (N\Sigma XY - (\Sigma X)(\Sigma Y))/(N\Sigma X2 - (\Sigma X)2) = ((5) * (1159.7) -$
$(311)*(18.6))/((5)*(19359)-(31\Sigma)\Sigma) = (5\Sigma 98.5-5784.6)/(96795-96721) =$
0.18783783783783292

Step 5:$Now, again substitute in the above intercept formula given. Intercept(a) =$
$(\Sigma Y - b(\Sigma X))/N = (18.6 - 0.1878378378378292(311))/5 = -7.964$

Step 6: Then substitute these values in regression equation formula Regression Equation(y) = a + bx = -7.964 + 0.188x Suppose if we want to calculate the approximate y value for the variable x = 64 then, we can substitute the value in the above equation Regression Equation(y) = a + bx = -7.964 + 0.188(64) = 4.068

5.1.2 SVM Algorithm

Step 1: we have dataset and we want to classify it Most of the time your data will be composed of 'n' vectors 'xi' Each 'xi' will also be associated with a value 'yi' Indicating if the element belongs to the class (+1) or not (-1). Note that 'yi' can only have two possible values -1 or +1. Moreover, most of the time, for instance when you do text classification, your vector 'xi' ends up having a lot of dimensions. We can say that 'xi' is a p-dimensional vector if it has p dimensions.
So our dataset say 'D' is the set of n couples of element (xi,yi) The more formal definition of an initial dataset in set theory is:

D = (xi , yi) xi $\in R^p, yi \in -1, 1ni = 1$
Step 2: You need to select hyperplanes separating the data with no points between them. We saw previously, that the equation of a hyper plane can be written 'wTx=0'. Any hyperplane can be written as the set of points 'x' Which satisfying wxb=0 [6]

Step 3: Maximize the distance between the two hyperplanes

1. calculate the difference between hyperplane
   'H0' be the hyperplane having the equation wxb=1
   'H1' be the hyperplane having the equation wxb=1

   'x0' be a point in the hyperplane 'H0'
   We will call 'm' the perpendicular distance from 'x0' to the hyperplane 'H1'. m is what we are used to call the margin. As x0 is in H0, m is the distance between hyperplanes H0 and H1 We will now try to find the value of 'm'

2. calculate margin
   We now have a formula to compute the margin:
   m=2 / w
   When w=1 then m=2
   When w=2 then m=1
   When w=4 then m=1/2
   we will choose the hyperplane with the smallest w because it is the one which will have the biggest margin.

Example take iris data as an input to the system to apply svm algorithm

1. import the iris liabrary
   library(e1071)

2. set the labels and classify into test and train data
   x ¡- iris[25:75,]
   labels ¡- x[,5]
   $\text{test}_data < -iris[1:25,]$

3. Calculate the svm model and predict the data
   model ¡- svm(Species ., data=x, cost = 5, degree = 7)
   pred ¡- predict(model, test_data)

4. Print the output
   print(pred)

5.1.3 ANN Algorithm

The backpropagation learning algorithm can be divided into two phases: propagation and weight update.

Phase 1: Propagation
Each propagation involves the following steps:
Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

Phase 2: Weight update
For each weight-synapse follow the following steps:
Multiply its output delta and input activation to get the gradient of the weight.
Subtract a ratio (percentage) of the gradient from the weight. This ratio (percentage) influences the speed and quality of learning; it is called the learning rate. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing, this is why the weight must be updated in the opposite direction.
Repeat phase 1 and 2 until the performance of the network is satisfactory.
Steps for Algorithm:

The project describes teaching process of multi-layer neural network employing back propagation algorithm. To illustrate this process the three

layer neural network with two inputs and one output, which is shown in the picture below, is used:

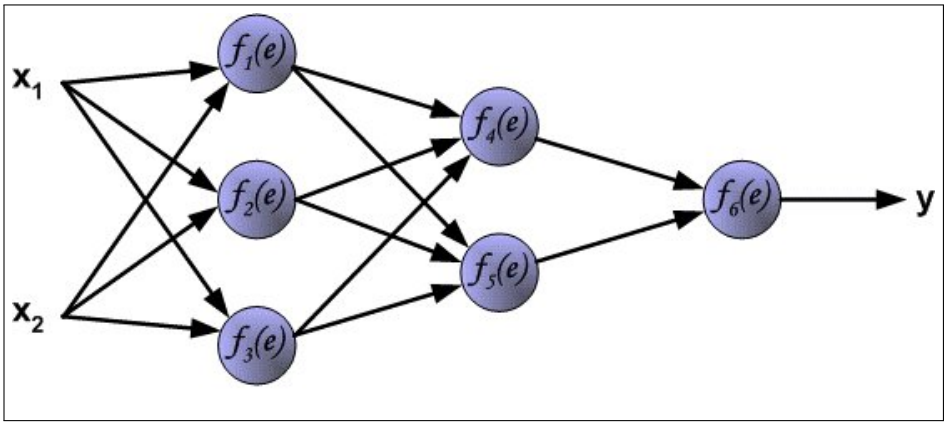


Figure 5.1: ANN model

Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realise nonlinear function, called neuron activation function. Signal e is adder output signal, and y = f(e) is output signal of nonlinear element. Signal y is also output signal of neuron.

To teach the neural network we need training data set. The training data set consists of input signals (x1 and x2) assigned with corresponding target (desired output) z. The network training is an iterative process. In each iteration weights coefficients of nodes are modified using new data from training data set. Modification is calculated using algorithm described below: Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer. Pictures below illustrate how signal is propagating through the network, Symbols w(xm)n represent weights of connections between network input xm and neuron n in input layer. Symbols yn represents output signal of neuron n.

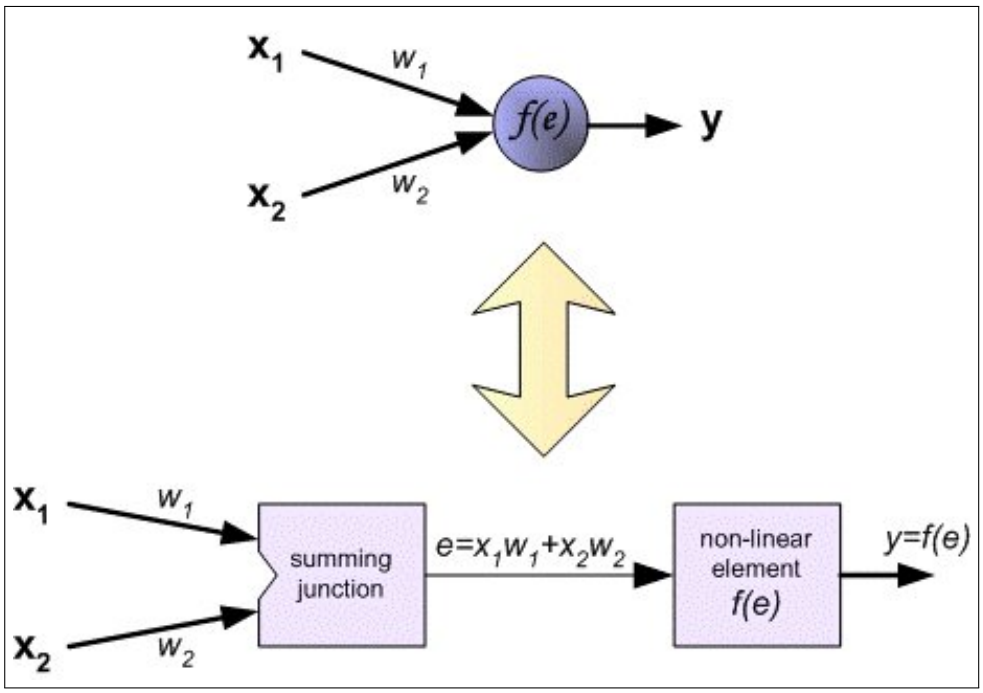


Figure 5.2: ANN model

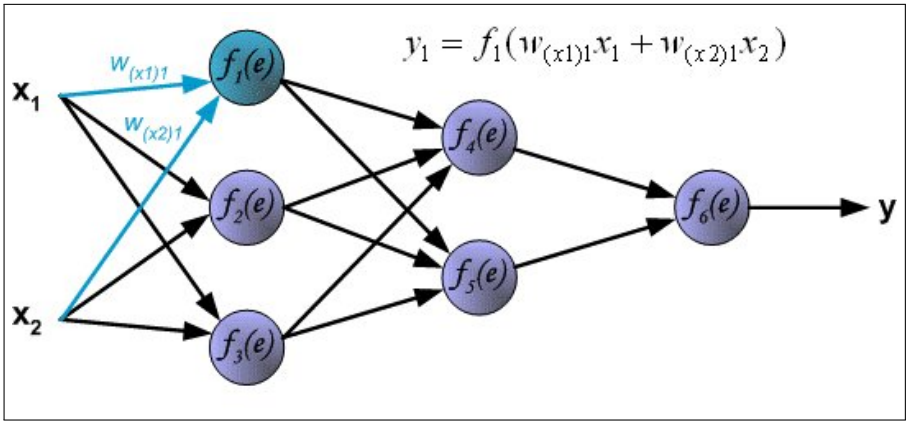


Figure 5.3: ANN model

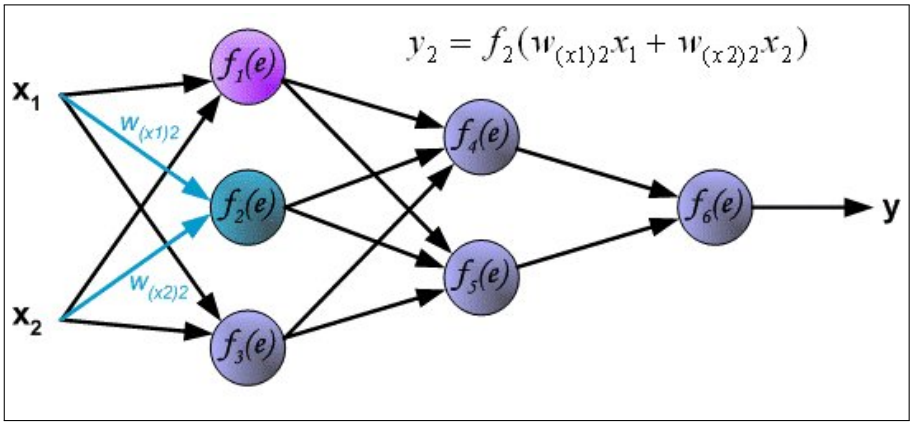


Figure 5.4: ANN model

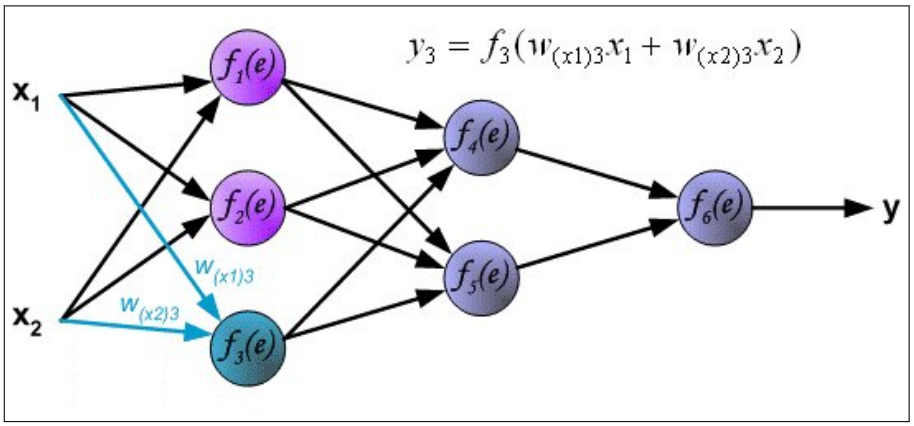


Figure 5.5: ANN model

Propagation of signals through the hidden layer. Symbols wmn represent weights of connections between output of neuron m and input of neuron n in the next layer.

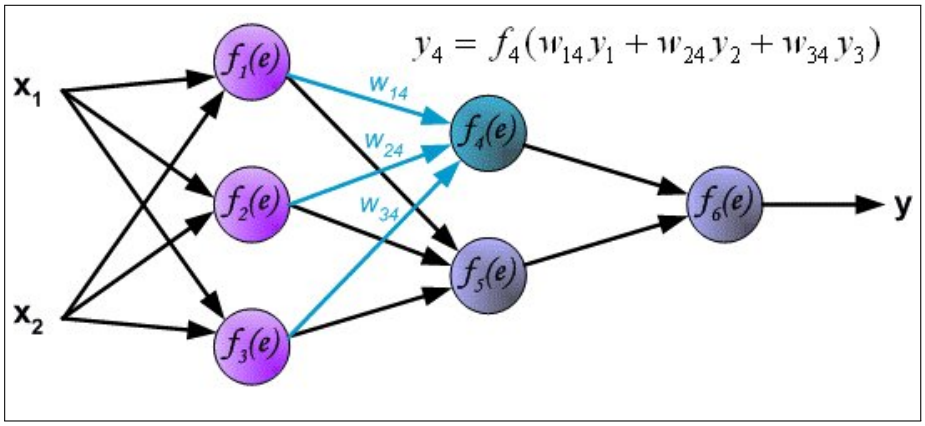


Figure 5.6: ANN model

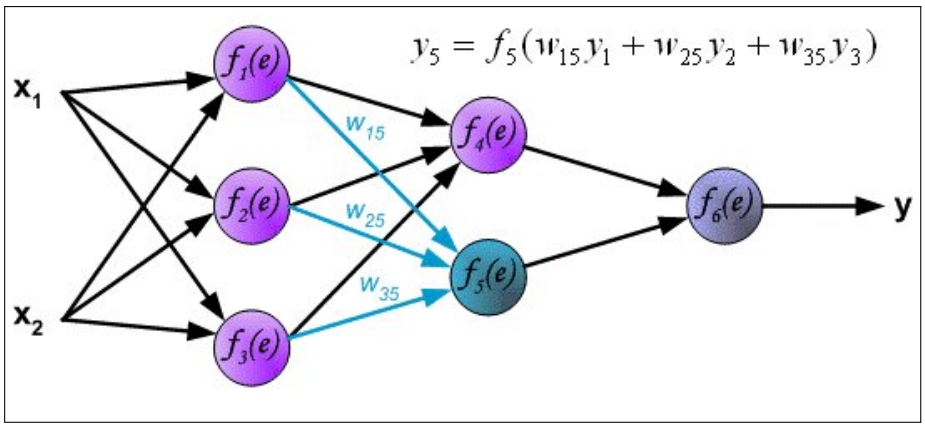


Figure 5.7: ANN model

Propagation of signals through the output layer.

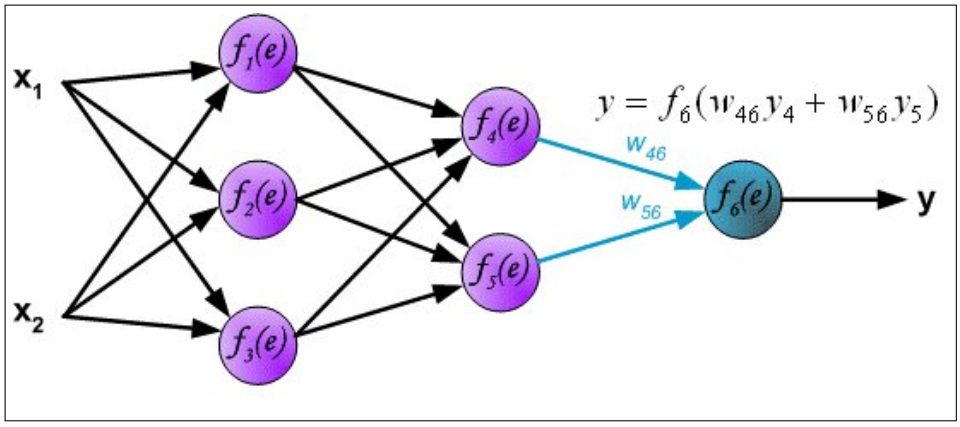


Figure 5.8: ANN model

In the next algorithm step the output signal of the network y is compared with the desired output value (the target), which is found in training data set. The difference is called error signal of output layer neuron.

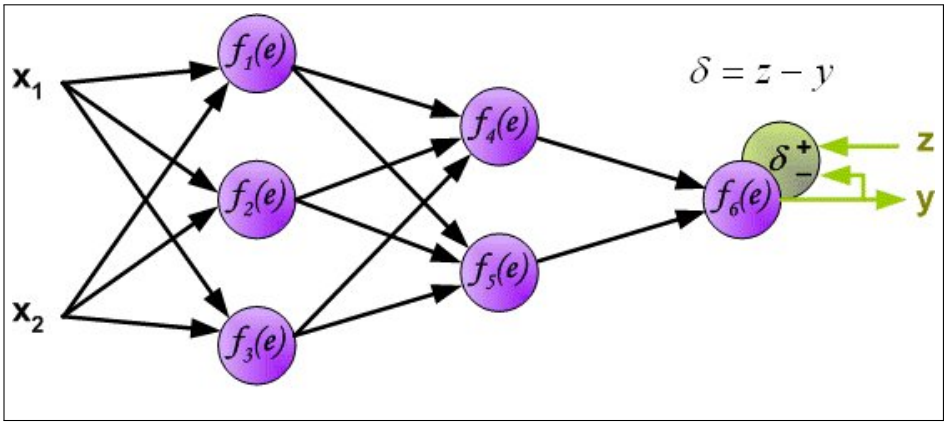


Figure 5.9: ANN model

It is impossible to compute error signal for internal neurons directly, because output values of these neurons are unknown. For many years the effective method for training multiplayer networks has been unknown. Only in the middle eighties the backpropagation algorithm has been worked out. The idea is to propagate error signal (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.

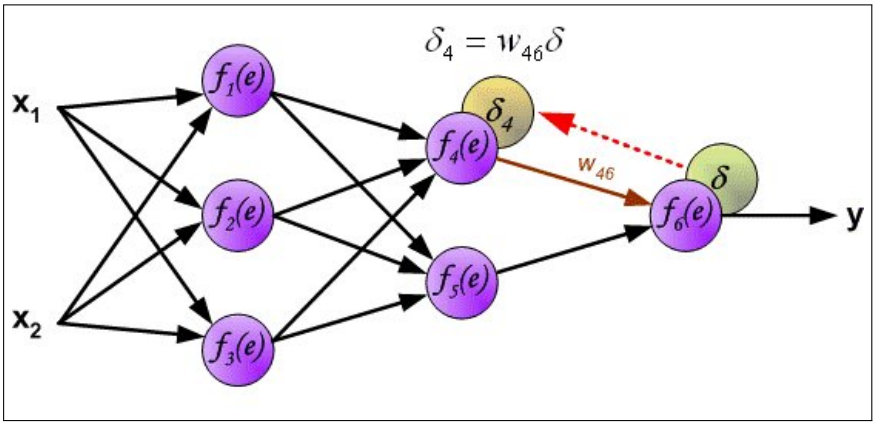


Figure 5.10: ANN model

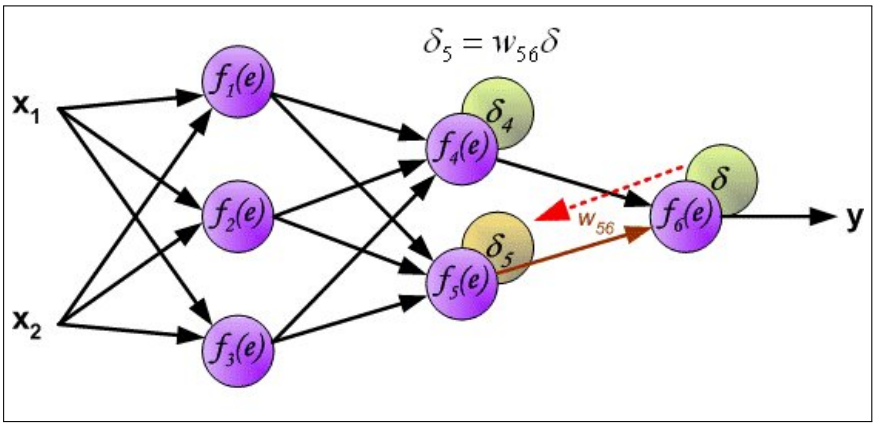


Figure 5.11: ANN model

The weights' coefficients wmn used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers. If propagated errors came from few neurons they are added. The illustration is below:

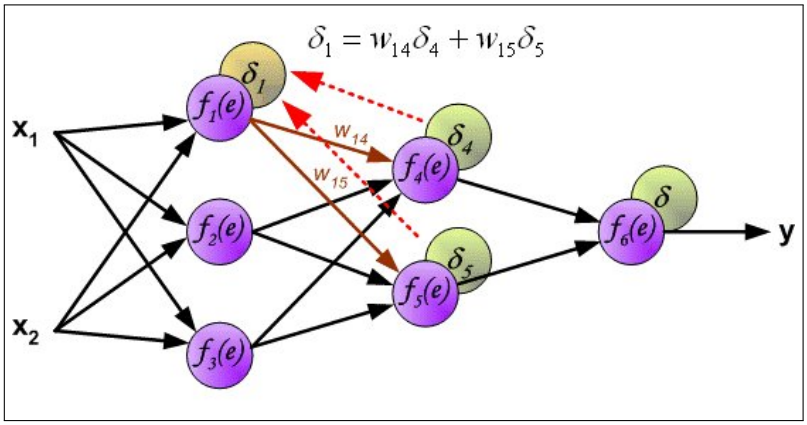


Figure 5.12: ANN model

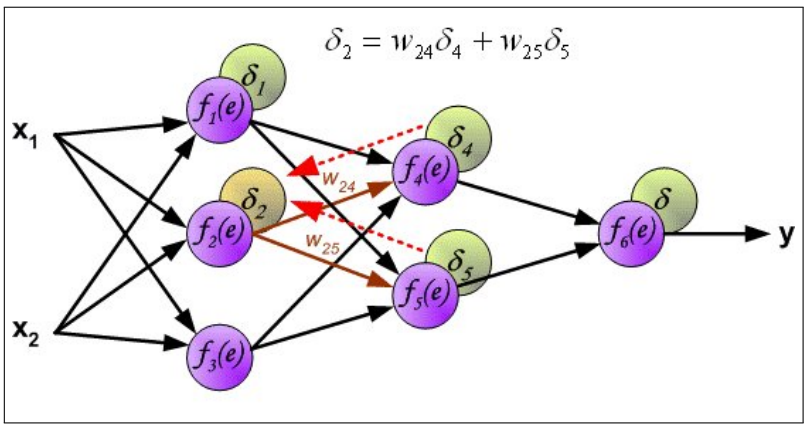


Figure 5.13: ANN model

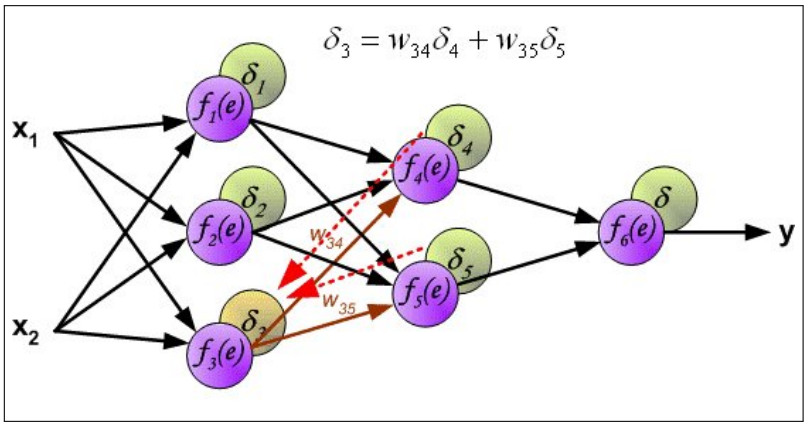


Figure 5.14: ANN model

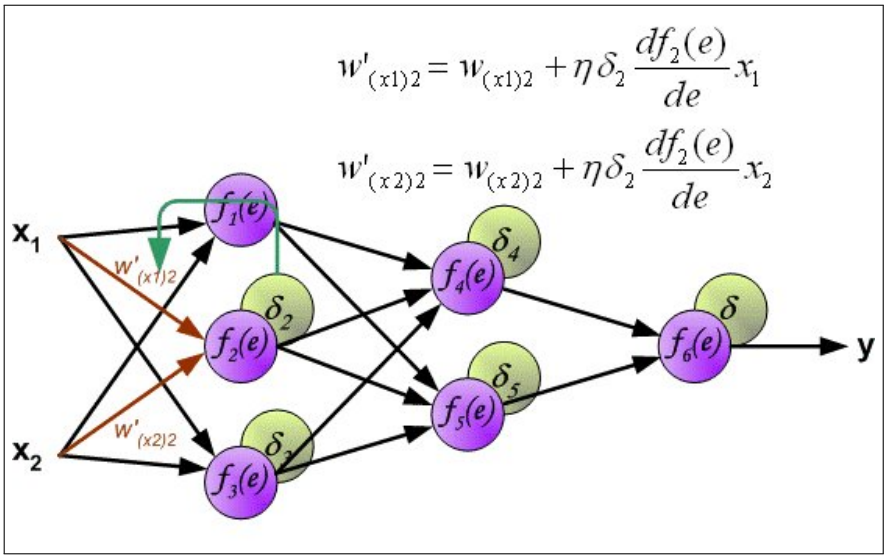


Figure 5.15: ANN model

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below df(e)/de represents derivative of neuron activation function (which weights are modified).

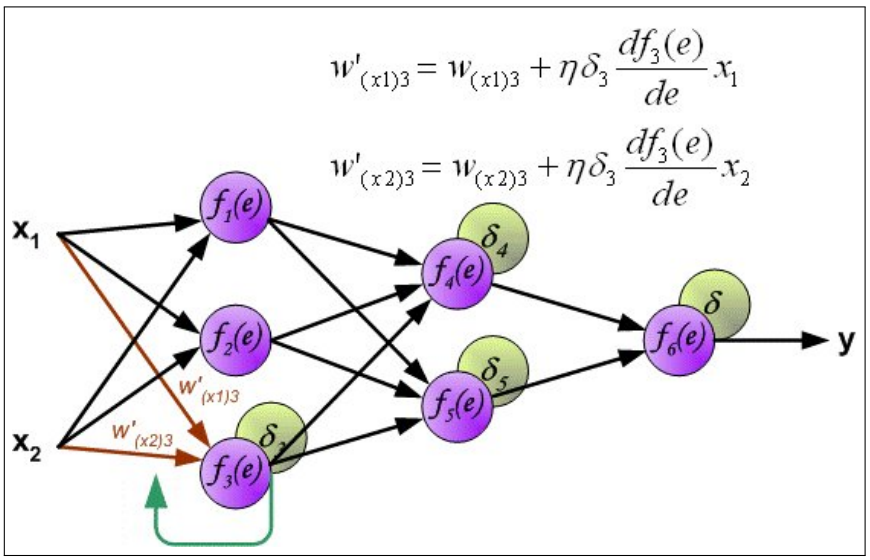


Figure 5.16: ANN model

Coefficient affects network teaching speed. There are a few techniques to select this parameter. The first method is to start teaching process with large value of the parameter. While weights coefficients are being established the parameter is being decreased gradually. The second, more complicated, method starts teaching with small parameter value. During the teaching process the parameter is being increased when the teaching is advanced and then decreased again in the final stage. Starting teaching process with low parameter value enables to determine weights coefficients signs.[11]

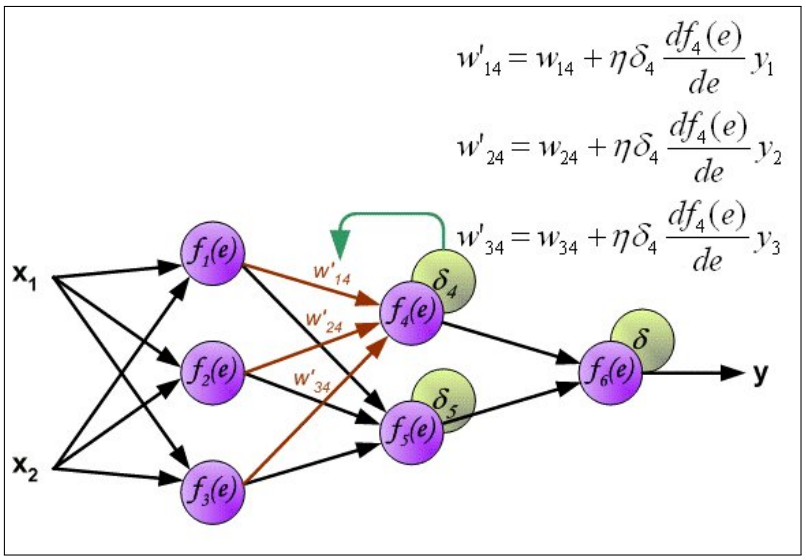


Figure 5.17: ANN model

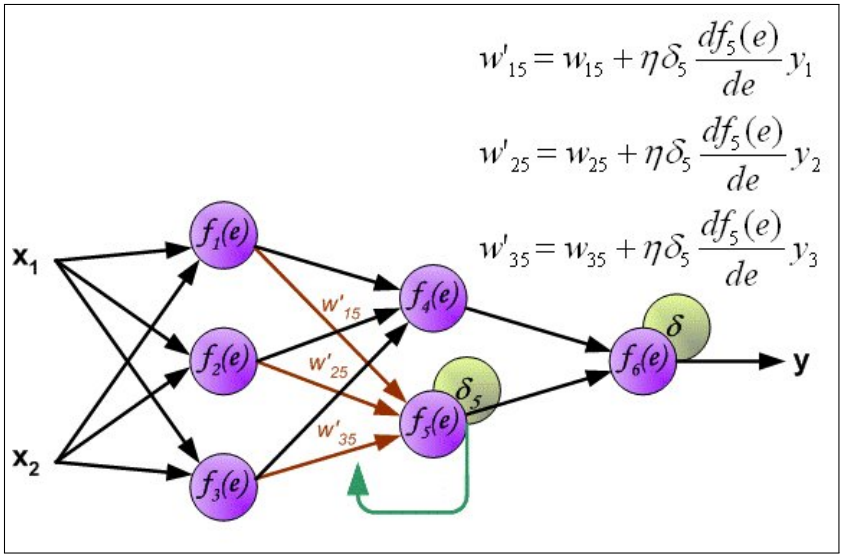


Figure 5.18: ANN model

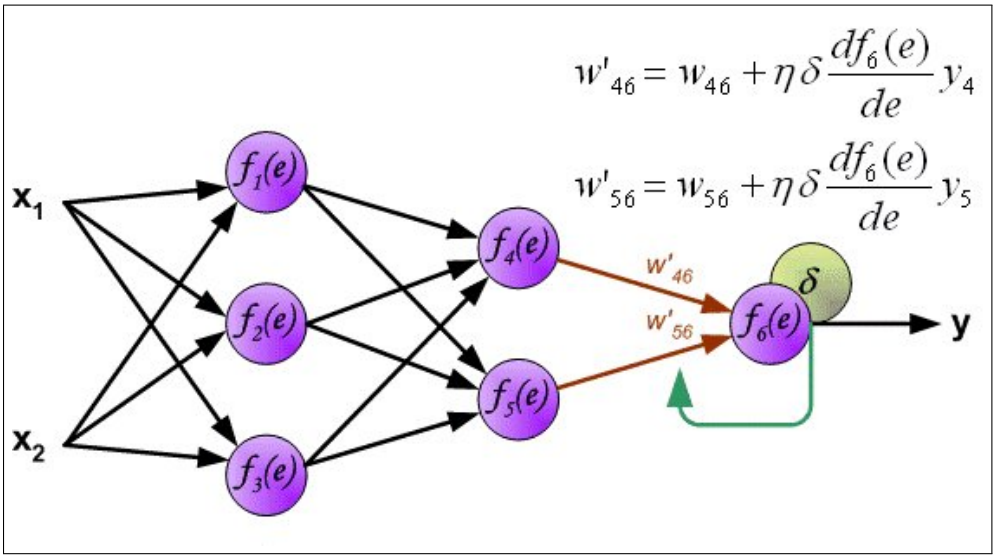


Figure 5.19: ANN model

5.1.4 ARIMA Model

Cases of Arima Models:

1. ARIMA(1,0,0) = first-order autoregressive model: if the series is stationary and autocorrelated, perhaps it can be predicted as a multiple of its own previous value, plus a constant.
   The forecasting equation in this case is
   $\hat{Y}t = \mu + \phi 1t - 1$

2. ARIMA(0,1,0) = random walk: If the series Y is not stationary, the simplest possible model for it is a random walk model, which can be considered as a limiting case of an AR(1) model in which the autoregressive coefficient is equal to 1, i.e., a series with infinitely slow mean reversion. The prediction equation for this model can be written as:

   $\hat{Y}t - Yt - 1 = \mu orequivalently \hat{Y}t = \mu + Yt - 1$

3. ARIMA(1,1,0) = differenced first-order autoregressive model
   If the errors of a random walk model are autocorrelated, perhaps the problem can be fixed by adding one lag of the dependent variable to the prediction equation–i.e., by regressing the first difference of Y on itself lagged by one period. This would yield the following prediction equation:
   $\hat{Y}t - Yt - 1 = \mu + \phi 1(Yt - 1 - Yt - 2)$
   $\hat{Y}t - Yt - 1 =$
   $which can be rearranged to \hat{Y}t = \mu + Yt - 1 + \phi 1(Yt - 1 - Yt - 2)$

4. ARIMA (0,1,1) without constant = simple exponential smoothing: Another strategy for correcting auto correlated errors in a random walk model is suggested by the simple exponential smoothing model. Equation is:
$\hat{Y}t = \hat{Y}t - 1 + \alpha et - 1$

5. ARIMA(0,1,1) with constant = simple exponential smoothing with growth: By implementing the SES model as an ARIMA model, you actually gain some flexibility. First of all, the estimated MA(1) coefficient is allowed to be negative. The ARIMA(0,1,1) model with constant has the prediction equation:
$\hat{Y}t = \mu + Yt - 1 - \theta 1et - 1$

6. ARIMA(0,2,1) or (0,2,2) without constant = linear exponential smoothing: Linear exponential smoothing models are ARIMA models which use two nonseasonal differences in conjunction with MA terms.
The second difference of a series Y is not simply the difference between Y and itself lagged by two periods, but rather it is the first difference of the first difference i.e., the change-in-the-change of Y at period t. Thus, the second difference of Y at period t is equal to
(Yt - Yt-1) - (Yt-1 - Yt-2) = Yt - 2Yt-1 + Yt-2.
A second difference of a discrete function is analogous to a second derivative of a continuous function: it measures the "acceleration" or "curvature" in the function at a given point in time. Equation can be written as:
$\hat{Y}t - 2Yt - 1 + Yt - 2 = -\theta 1et - 1 - \theta 2et - 2$
$which can rearrange das:$
$\hat{Y}t = 2Yt - 1 - Yt - 2 - \theta 1et - 1 - \theta 2et - 2$

7. ARIMA(1,1,2) without constant = damped-trend linear exponential smoothing:
$\hat{Y}t = Yt - 1 + \phi 1(Yt - 1 - Yt - 2) - \theta 1et - 1 - \theta 1et - 1$

Examples:
Some well-known special cases arise naturally or are mathematically equivalent to other popular forecasting models like

1.An ARIMA(0,1,0) model (or I(1) model) is given by $X_t = X_t - 1 + \epsilon_t which is simply a random walk$

2. An ARIMA(0,1,0) with a constant, given by $X_t = c + X_t - 1 + \epsilon_t which is a random walk with drift.$

3.An ARIMA(0,0,0) model is a white noise model

4.An ARIMA(0,1,2) model is a Damped Holt's model.

5.An ARIMA(0,1,1) model is a Basic Exponential Smoothing.

6.An ARIMA(0,2,2) model is given by $X_t = X_t - 1 + X_t - 2 + (+\beta - 2)\epsilon_t - 1 +$
$(1-)\epsilon_t - 2 + \epsilon_t which is equivalent to Holt's linear method with additive errors, or Double exponential s$

7.An ARIMA(0,3,3) model is a Triple exponential smoothing.

Chapter 6

Software Testing

6.1 Introdution

Testing is a process used to help identify the correctness, completeness and quality of developed computer software. With that in mind, testing can never completely establish the correctness of computer software.

There are many approaches to software testing, but e active testing of complex products is essentially a process of investigation, not merely a matter of creating and following rote procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are things the tester tries to do with the product, and the product answers with its behavior in reaction to the probing of the tester. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product-putting the product through its paces.

The quality of the application can and normally does vary widely from system to sys-tem but some of the common quality attributes include reliability, stability, portability, maintainability and usability.

Testing helps is verifying and validating if the Software is working as it is intended to be working. Thins involves using Static and Dynamic methodologies to Test the application.

1. Testing is a process of executing a program with the intent of ending an error.

2. A good test case is one that has a high probability of ending an as yet

undiscovered error.

3. A successful test is one that uncovers an as yet undiscovered error.

6.2 Test Cases

Figure 6.1: Test case 1

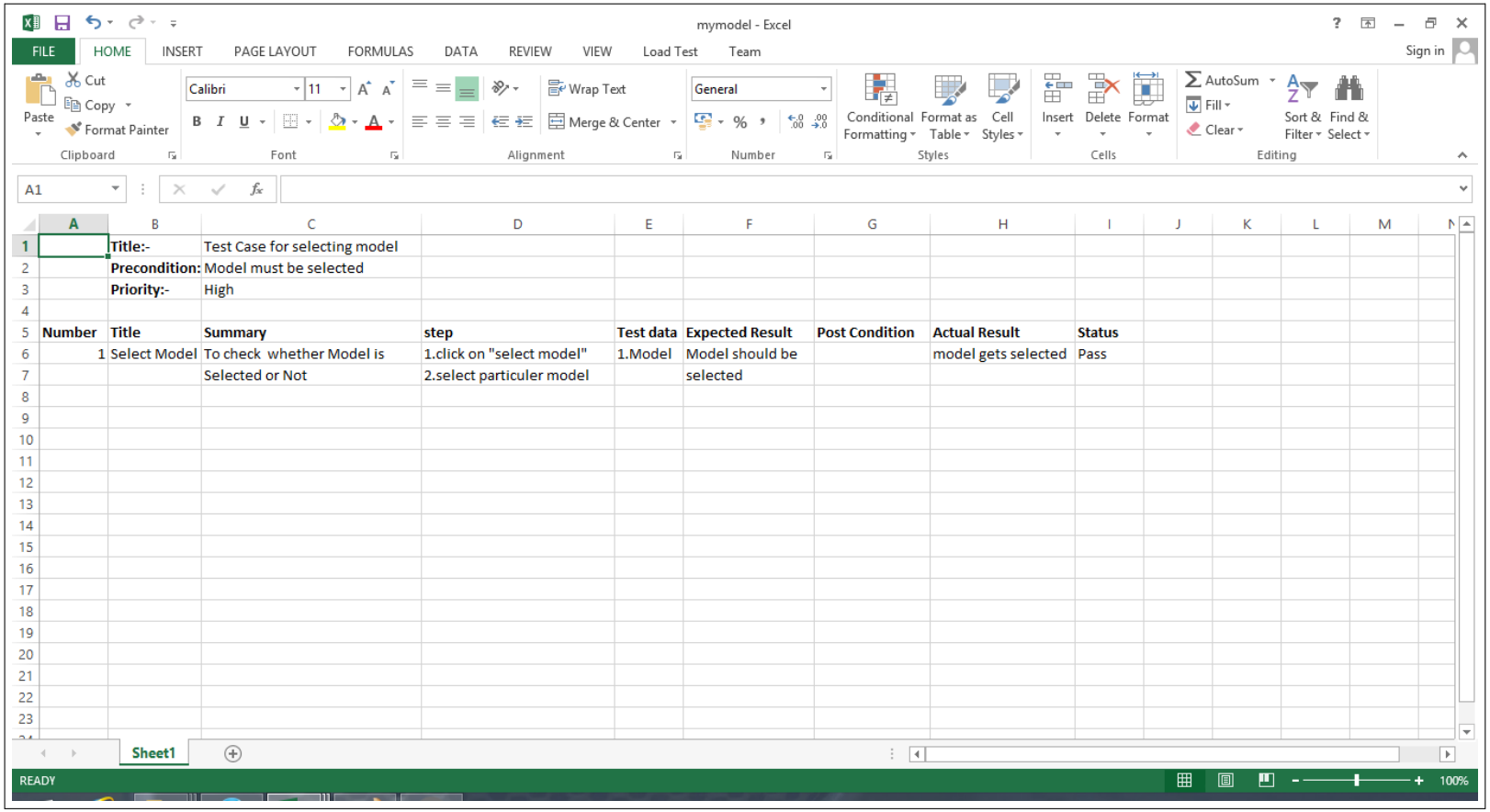
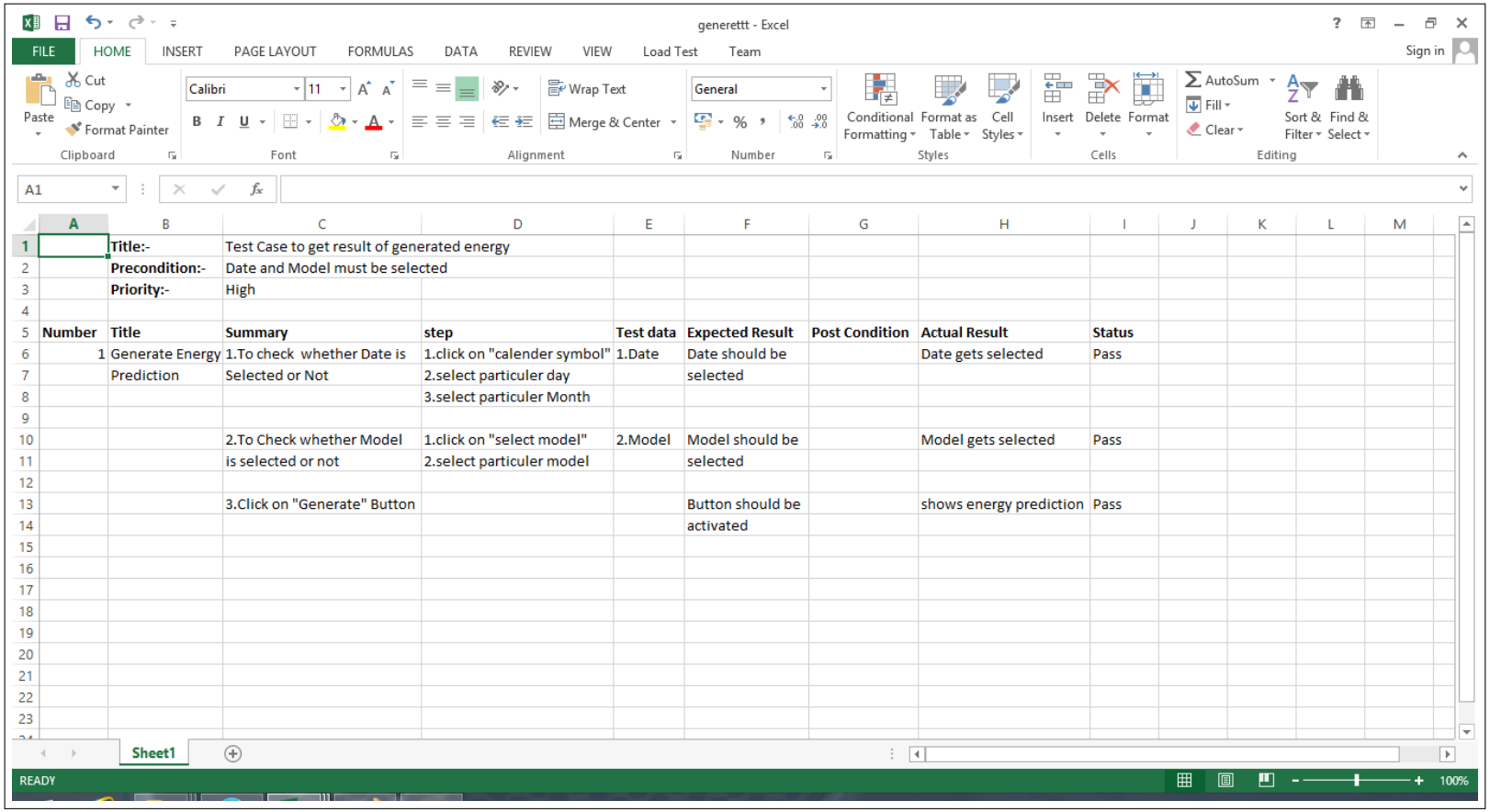


Figure 6.2: Test case 2



generettt - Excel

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		**Title:-**	Test Case to get result of generated energy										
2		**Precondition:-**	Date and Model must be selected										
3		**Priority:-**	High										
4													
5	**Number**	**Title**	**Summary**	**step**	**Test data**	**Expected Result**	**Post Condition**	**Actual Result**	**Status**				
6	1	Generate Energy	1.To check whether Date is	1.click on "calender symbol"	1.Date	Date should be		Date gets selected	Pass				
7		Prediction	Selected or Not	2.select particuler day		selected							
8				3.select particuler Month									
9													
10			2.To Check whether Model	1.click on "select model"	2.Model	Model should be		Model gets selected	Pass				
11			is selected or not	2.select particuler model		selected							
12													
13			3.Click on "Generate" Button			Button should be		shows energy prediction	Pass				
14						activated							
15													
16													
17													
18													
19													
20													
21													
22													
23													

Figure 6.3: Test case 3

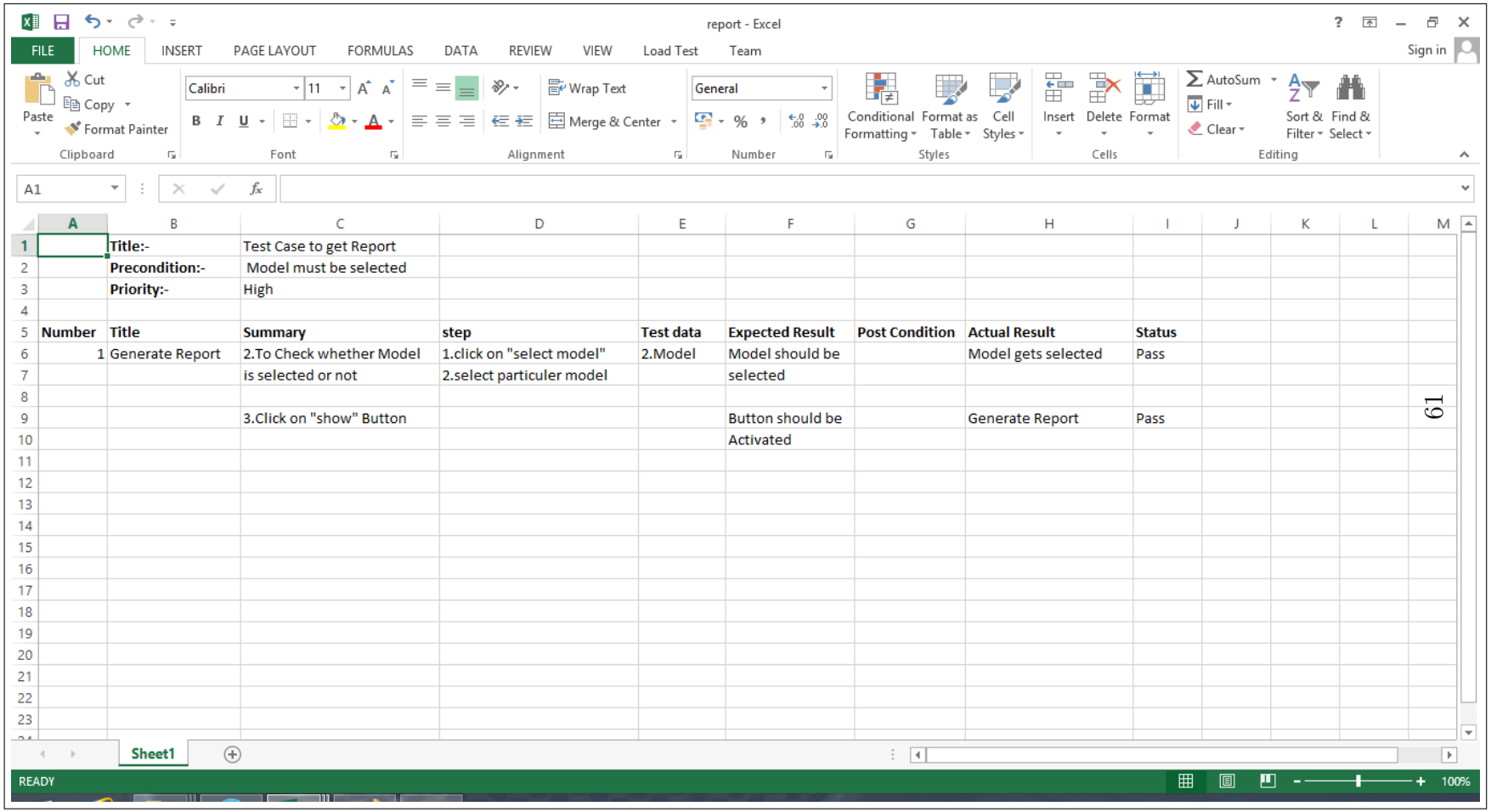


Figure 6.4: Test case 4

Chapter 7

Implementation

7.1 Sample Code

Regression Model

```
# The R script that sources this script should define the variables
data.file     <- "p12_12.csv" # data file
output.prefix <- "test"     # prefix for output files
covariates    <- 1:4            # covariate column indices
response      <- 5              # response variable column index
#
# See example car-mileage.r.
#
 stopifnot( exists( "data.file" ),     is.character( data.file ),
            exists( "covariates" ),    is.numeric( covariates ),
             length( covariates ) > 0,
            exists( "response" ),      is.numeric( response ),
             length( response ) == 1,
            exists( "output.prefix" ), is.character( output.prefix ) )

# read data
#
 data <- read.csv( data.file, header = T, as.is = T, comment = '#' )

 N               <- nrow( data )                 # data size
 covariate.names <- colnames( data )[covariates] # covariate names
 n               <- length( covariates )         # total number of
 response.name   <- colnames( data )[response]   # response variable name
```

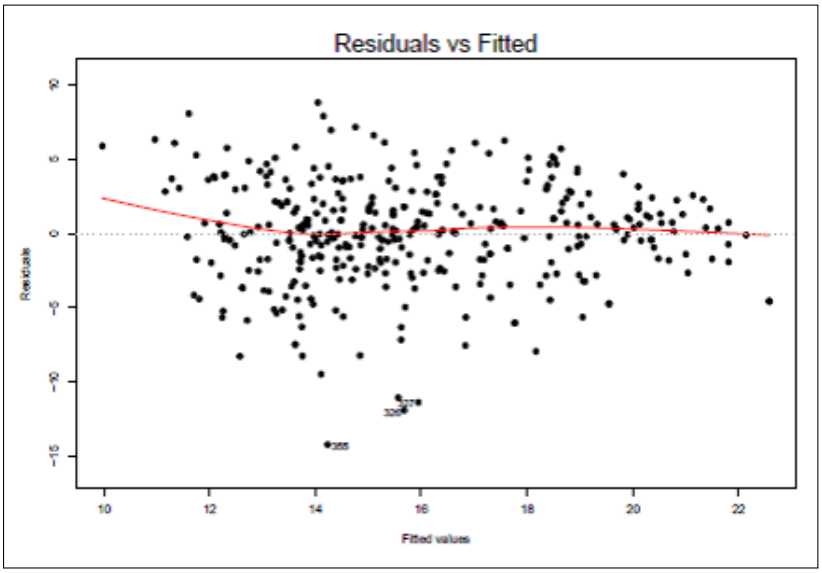


Figure 7.1: Linear Model:Residuals vs Fitted

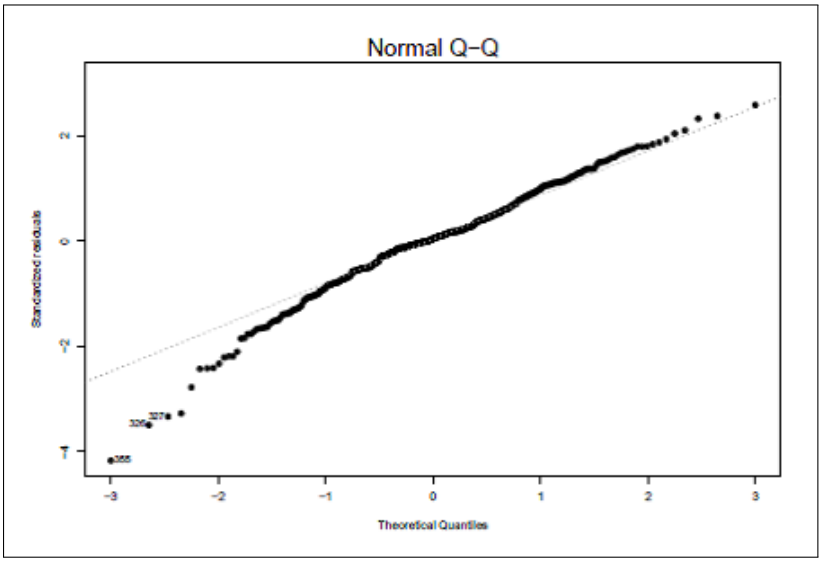


Figure 7.2: Normal Q-Q

SVM Model

```
library(e1071)
data<-read.csv("p12_12.csv",header=TRUE)

#par(mfrow=c(3,1))
#cor(data)
#pdf("Summary.pdf")
#plot(data)
#plot(data$DTRAD,type="l")

train<-data[1:292,]
test<-data[293:365,]

svm.model<-svm(DTRAD~.,data=train,cost=1000,gamma=0.0001)

svm.pred<-predict(svm.model, train[,-5])

svm.pred1<-predict(svm.model, test[,-5])
rmse <- function(error)
{
  sqrt(mean(error^2))
}
error<-(data$DTRAD[1:292] - svm.pred)
print(rmse(error))
error1<-(data$DTRAD[293:365] - svm.pred1)
print(rmse(error1))

tuneResult <- tune(svm, DTRAD ~ .,  data = data,
              ranges = list(epsilon = seq(0,1,0.1), cost = 2^(2:9))
)
print(tuneResult)
```

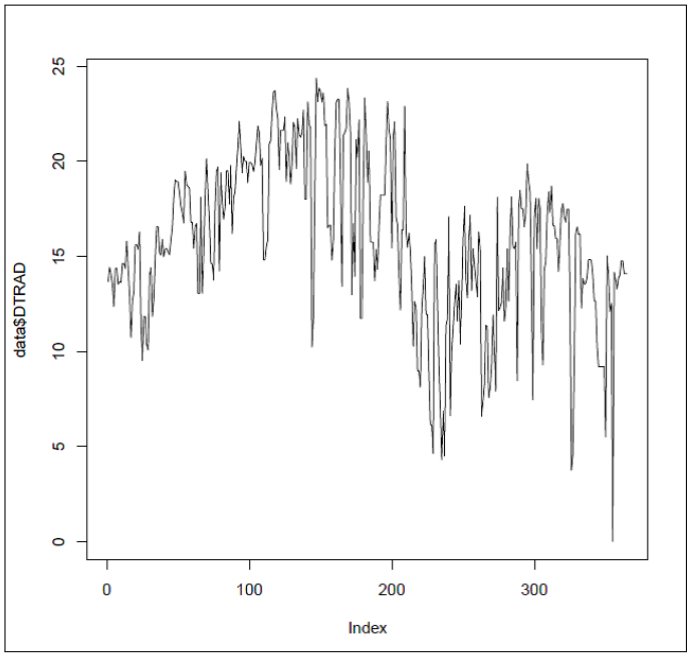


Figure 7.3: SVM: Yearly radiation data

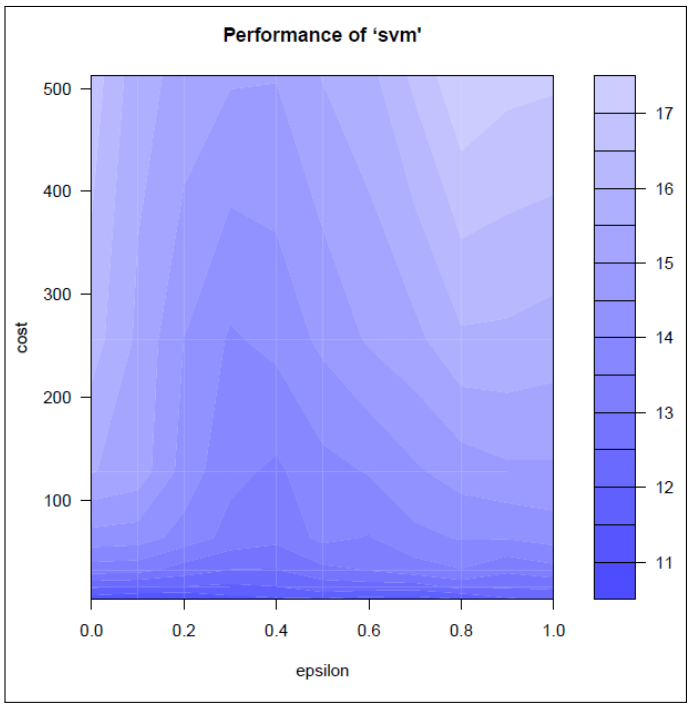


Figure 7.4: SVM model tuning

ANN model

```
library(MASS)
library("neuralnet")
library(nnet)
data<-read.csv("p12_12.csv",header=TRUE
data$RH<-as.numeric(data$RH)
#data<-sort(data[,5])
train<-(data[1:250,-4])
test<-(data[251:330,c(-5,-4)])
#new_train<-scale(train,center=TRUE,scale=TRUE);
#new_test<-scale(test,center=TRUE,scale=TRUE);
#response<-data$DTRAD
#resp_ind1<-class.ind(response[1:250])
#resp_ind2<-(response[251:330])

model<-neuralnet(DTRAD~DBT+DPT+RH, data=train, hidden =4, lifesign = "minimal",
#pdf("model.pdf")
plot(model, rep = "best")
#dev.off()
pred1<-compute(model,train[,1:3])
#result1<-data.frame(actual=test$DTRAD,prediction=pred1$net.result)
```

ARIMA Model

```
#par(mfrow=c(3,4))
data<-read.csv("p12_12.csv",header=TRUE)

ln_model<-lm(data$DTRAD~.,data=data)
Y<-residuals(ln_model)
#plot(Y)
pdf("sesonal.pdf")
par(mfrow=c(3,4))
plot(data$DTRAD,col="blue",xlab='Days',ylab='Radiation')
plot(Y,ylab='Residuals')
Z=diff(Y,12)
acf(Z,lag=36,lwd=3, col='red')
pacf(Z,lag=36,lwd=3,col='blue')
```

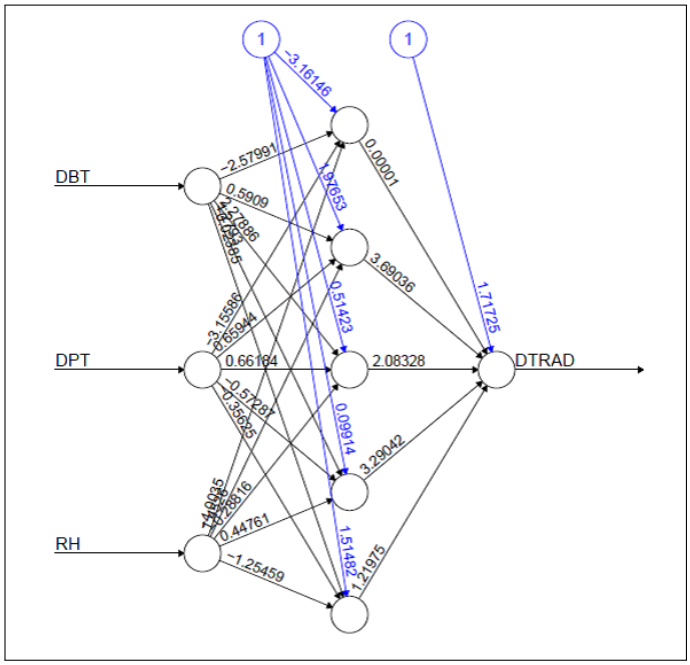


Figure 7.5: ANN model

```
model1=arima(Z,order=c(0,0,1))
E1=residuals(model1)
acf(E1,lag=36,lwd=3,col='red')

model2=arima(Z,order=c(1,0,0))
E2=residuals(model2)
acf(E2,lag=36,lwd=3,col='blue')

model2b=arima(Y,order=c(1,0,0), seasonal = list(order = c(0, 1, 0),period=12))
E3=residuals(model2b)
model3c=arima(Y,order=c(1,0,0), seasonal = list(order = c(1, 0, 0), period = 12)
```

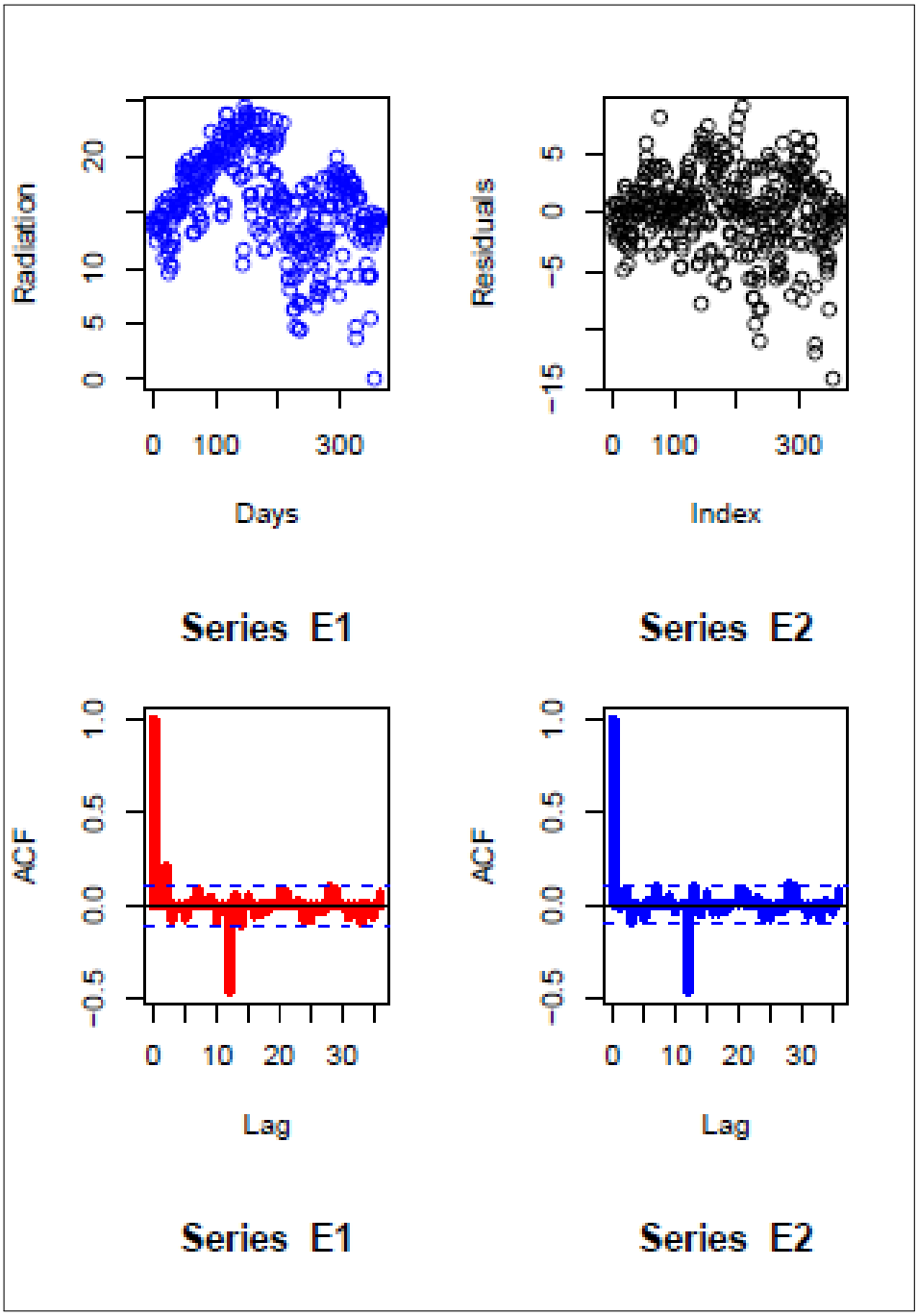


Figure 7.6: ARIMA model

7.2 Comparison of Linear Regression and SVM

TEST	LR	SVM
Observation	73	73
Performance	0.25	11.14
Error	3.76	4.31
Compile Time	less	more
complexity	Less	more
TRAIN	LR	SVM
Observation	292	292
Performance	0.68	11.01
Error	3.19	3.17
Compile Time	less	more
Complexity	less	more

Figure 7.7: Linear regression vs SVM

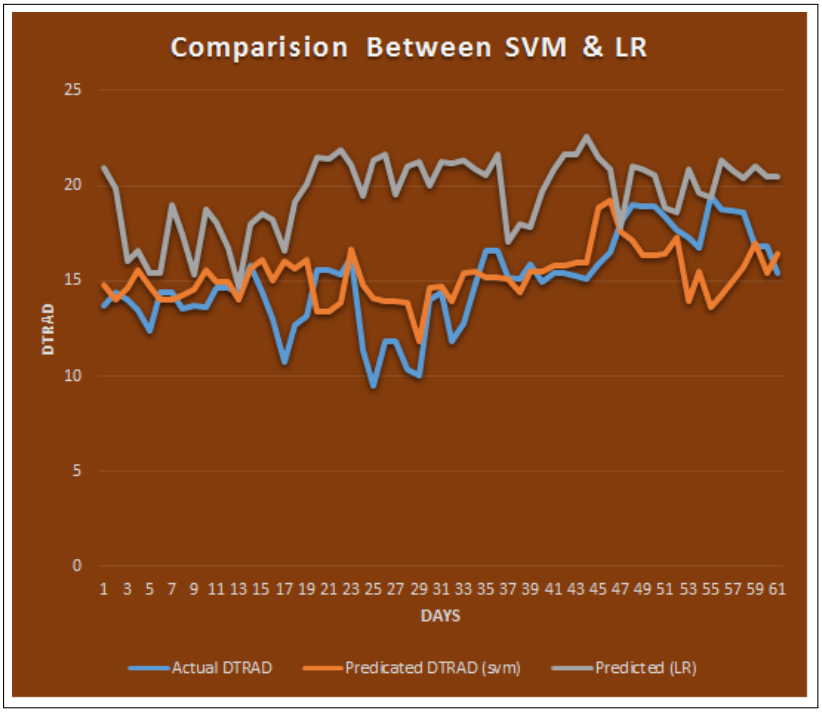


Figure 7.8: Graph: Linear regression vs SVM

Chapter 8

Planning & Scheduling

Schedule Activities Status of Development:

1. Literature survey is completed.
2. Study of existing system is done.
3. Data collection is completely done.
4. We visited National Data Center (NDC), Shivaji Nagar, Pune to get the real time datasets.
5. Participated in Avishkar Zonal Level Competition of Poster presentation.
6. Machine learning algorithm is decided.
7. Data analysis map generation using R-language is completely done.
8. Regression equation of solar radiation is calculated.
9. Apply supervised learning methods using SVM.
10. Study and Apply ANN algorithm.
11. Study and Apply ARIMA algorithm which is used for whether forecasting.
12. Reports are generated successfully.
13. Energy prediction is also done by system.
14. Solar radiation of next year is predicted successfully.

Future Scope

We need more data for accurate results of ANN and ARIMA models. So prediction of solar radiation by using ANN and ARIMA algorithm is done in future. We also intend to expand the region of work. In future we also intend to work on renewable energy like water energy and wind energy

Chapter 9

Conclusion

Prior prediction models for solar energy harvesting have been based primarily on the immediate past [9], [10], [11].Unfortunately, the methods are unable to predict changes in weather patterns in advance. Since weather forecasts from the NDC are based on aggregations of multiple data sources from across the country, they are able to provide advance warning. We show that the relationship between these forecast weather metrics and solar intensity is complex. Thus, we automatically derive linear regression prediction model which is 71% accurate from historical solar intensity and forecast data for radiation. The accuracy of the system will increase with more training. Our results indicate that automatically generating accurate models that predict solar intensity, and hence energy harvesting of solar arrays, from weather forecasts is a promising area. We find that models derived using linear least squares outperform a past-predicts future models and a simple model based on sky condition forecasts from prior work [4] and is a promising area for increasing the accuracy of solar power generation prediction with more training, which is essential to increase the fraction of renewables in the grid. Moving forward, we plan on using our prediction models to better match renewable generation to consumption in both smart homes and data centres that utilize on-site solar arrays to generate power.

Appendix VII

Bibliography

1. Database of State Incentives for Renewables and Efficiency, http://www.dsireusa.org, 2010.

2. State of California Executive Order S-21-09, http://gov.ca.gov/executive-order/13269, 2009.

3. Freeing the Grid: Best and Worst Practices in State Net Metering Policies and Interconnection Procedures, http://www.newenergychoices.org/uploads/FreeingTheGrid2009.pdf,2009.

4. N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, Cloudy Computing: Leveraging Weather Forecasts in Energy Harvesting Sensor Systems, in SECON, June 2010.

5. N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2000.

6. LibSVM: A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/cjlin/libsvm/, 2011.

7. I. Jolliffe, Principal Component Analysis. Springer Series in Statistics, 2002.

8. A. Kansal, J. Hsu, S. Zahedi, and M. Srivastava, Power Management in Energy Harvesting Sensor Networks, in Transactions on Embedded Computing Systems, September 2007.

9. D. Noh, L. Wang, Y. Yang, H. Le, and T. Abdelzaher, Minimum Variance Energy Allocation for a Solar-powered Sensor System, in DCSS, June 2009.

10. C. Moser, Power Management in Energy Harvesting Embedded Systems, Ph.D. Thesis, ETH Zurich, March 2009

11. Ch.Jyosthna Devi , B.Syam Prasad Reddy, K.Vagdhan Kumar,B.Musala Reddy,N.Raja Nayak "ANN Approach for Weather Prediction using Back Propagation",Department Of Computer Science and Engineering, KLCE, Vaddeswaram, Guntur Dt.-522502, Andhra Pradesh, India.