**Major Project Report**

**On**

**“**Weather Prediction Using Machine Learning**”**

Submitted in partial fulfillment of the

requirements for the award of the degree of

**Bachelor of Technology**

**In**

**Computer Science & Engineering**

## By

## YERRA VENKAT (15R21A05Q0)

**PRANEESH DIBBIDI (15R21A05N2)**

**VALLALA MANOHAR (15R21A05P6)**

**MOHAMMED OMAAR (15R21A05M6)**

**UNDER GUIDENCE OF**

**Mrs.P.MADHURAVANI (Assoc Professor)**

**Department of Computer Science & Engineering**

**2018-2019**



**Department of Computer Science & Engineering**

**CERTIFICATE**

This is to certify that the project entitled “**Weather Prediction Using Machine Learning”** by **YERRA VENKAT (15R21A05Q0), \_PRANEESH\_DIBBIDI (15R21A05N2), VALLALA MANOHAR (15R21A05P6), MOHAMMED OMAAR (15R21A05M6)** has been submitted in the partial fulfillment. This is to the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

Internal Guide Head of the Department

(Professor N.CHANDRA SEKHAR REDDY)

External Examiner

# DECLARATION

We hereby declare that the project entitled **“Weather Prediction Using Machine Learning”** is the work done during the period from **January 2019 to April 2019** and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of technology in computer Science and Engineering from Jawaharlal Nehru Technology University, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

**YERRA VENKAT (15R21A05Q0)**

**PRANEESH DIBBIDI (15R21A05N2)**

**VALLALA MANOHAR (15R21A05P6)**

**MOHAMMED OMAAR (15R21A05M6)**

# ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project

successfully. We would like to take this opportunity to thank one and all.

First of all we would like to express our deep gratitude towards our internal guide **Mrs.P.MADHURAVANI (Assoc Professor)** Department of CSE for her support in the completion of our dissertation. We wish to express our sincere thanks to **Mr.N.CHANDRASHEKHAR REDDY, HOD, Dept. of CSE** and also to our principal **Dr.K.SRINIVAS RAO** for providing the facilities to complete the dissertation.

We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

**YERRA VENKAT (15R21A05Q0)**

**PRANEESH DIBBIDI (15R21A05N2)**

**VALLALA MANOHAR (15R21A05P6)**

**MOHAMMED OMAAR (15R21A05M6)**

# ABSTRACT

Weather forecasting is always a big challenge for the meteorologists to predict the state of the atmosphere at some future time and the weather conditions that may be expected. It is obvious that knowing the future of the weather can be important for individuals and organizations. Accurate weather forecasts can tell a farmer the best time to plant, an airport control tower what information to send to planes that are landing and taking off, and residents of a coastal region when a hurricane might strike. So, we want to build a platform that is extremely flexible and scalable to be able to analyze pentabytes of data across an extremely wide increasing wealth of weather variables. We are working on data analysis using Machine Learning and RStudio.

Machine learning is to program computers to use example data or past experience to solve a given problem. . Many successful applications of machine learning exist already, including systems that analyze past sales data to predict customer behavior, optimize robot behavior so that a task can be completed using minimum resources, and extract knowledge from bioinformatics data. Since Machine Learning has overwhelming advantage in optimizing big data, we prefer to use machine learning to analyze large datasets of weather processing

# CONTENTS

**Certificate I**

**Declaration II**

**Acknowledgement** **III**

**Abstract IV**

1. **Introduction** 1
2. **Literature Survey** 2
   1. Existing System
   2. Proposed System
3. **Requirement Analysis** 4
   1. Hardware Requirements
   2. Software Requirements
   3. Feasibility Study
4. **Implementation** 12
   1. Problem Definition
   2. Experiment
   3. System Architecture
5. **System Design** 14
6. **UML Diagrams** 15
7. **Coding**  18
8. **System Testing** 22
9. **Screen Shots** 27

**10. Future Enhancement**  37

**11. Conclusion**  38

**12. Bibliography** 40

# 1. INTRODUCTION

Traditionally, weather forecasting has always been performed by physically simulating the atmosphere as a fluid. The current state of the atmosphere is sampled. The future state of the atmosphere is computed by solving numerical equations of thermodynamics and fluid dynamics. But this traditional system of deferential equations that govern the physical model is sometimes unstable under disturbances and uncertainties while measuring the initial conditions of the atmosphere. This leads to an incomplete understanding of the atmospheric processes, so it restricts weather prediction up to a 10 day period, because beyond that weather forecasts are significantly unreliable. But Machine learning is relatively robust to most atmospheric disturbances as compared to traditional methods. Another advantage of machine learning is that it is not dependent on the physical laws of atmospheric processes.

**Machine Learning Techniques**

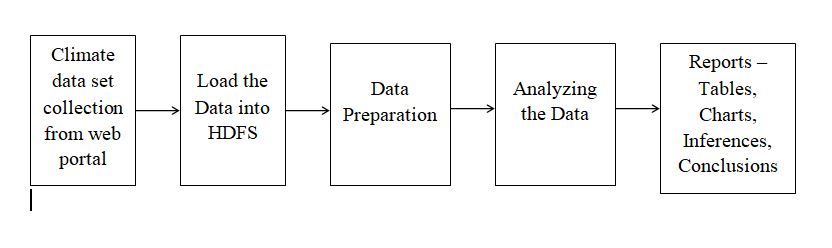
Linear Regression: it uses all the features present in the dataset and gives a linear graph combining high and low temperatures. Linear regression is not used for weather classification of each day because this algorithm cannot be used with classification data.

Functional Regression: The second algorithm to be used is a type of functional regression. It looks for historical weather patterns which are similar to the present day weather patterns, and then it predicts the future weather condition based upon the data of the historical weather patterns.

# 2. LITERATURE SURVEY

## 2.1 EXISTING SYSTEM:

The current system focuses on analyzing climate related data using Hadoop. The current system architecture is shown in the figure.



Step1: Data Preparation

Data Selection: The required data set is collected from the website

Data Loading: The collected data set loaded into Hadoop Distributed File System environment. Hadoop is the great tool to predict the climatic conditions, with processing of large and dynamic climate data.

Data Pre processing: The collected data set might consist of inconsistent data. If climatic analysis is performed on this data, it will produce wrong outcomes. Therefore, necessary pre processing techniques are applied before analyzing the data.

Step 2: Climatic Data Analysis

Climatic Data Analysis: Pre processed data set is now analyzed using big data. Prediction techniques like regression are applied to the climate data to predict the future climate in the city.

Step 3: Reports

Report Generation: After the climate data analysis, the necessary reports are generated.

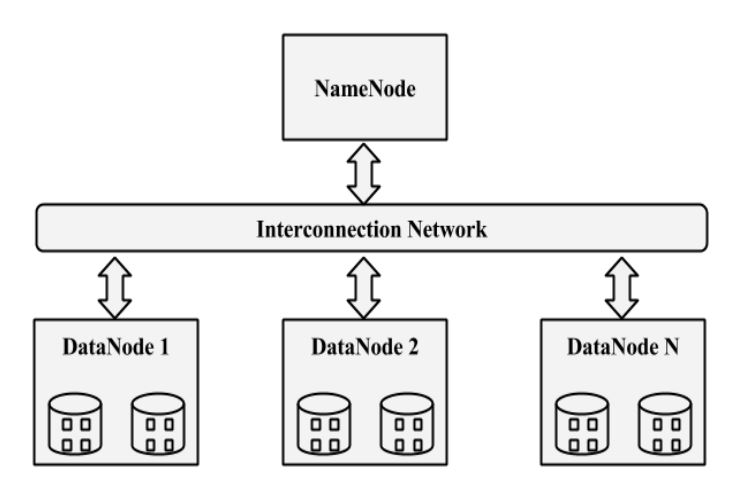
1. **Big Data**

Big Data can be defined as a large amount of data which requires advanced tools and technologies so that it becomes possible to extract value from it by capturing and analysis process. Big data is more than just size. There are three characteristics of Big Data a. Volume - It refers to the huge size of data which is being accumulated over the period of time. In case of weather data thousands of sensors are generating data for various weather parameters many times per hour from last many years. It poses the challenge to the storage and processing of huge data. b. Velocity - It refers to the speed at which data is arriving. If the tool cannot process the data in a time equal to or less than at which it is arriving then it will lead to huge accumulation of unprocessed data. c. Variety - It refers to various formats in which data is being generated. Like textual data, binary data, images and video data. In ASCII format XML, JSON, CSV are the different format in which data may come. Separate parser is a need to process each type of format.

1. **Hadoop**

Hadoop is the open source software library framework implemented in java for processing huge data on a cluster of commodity computers in a parallel and distributed manner. Hadoop is good for batch processing job. Hadoop has three major components viz. Hadoop Distributed File System (HDFS), Map Reduce Processing Engine and YARN.

1. **HDFS** - It is the open source implementation of the Google file system published in white paper. It is block oriented, distributed, fault tolerant, reliable, scalable and robust file system supporting huge amount of storage which can give streaming access to a data. The architecture of HDFS is shown in figure. It has master slave architecture and it has following main components



**Name Node** - It is the master node of HDFS. It manages the metadata about the file system. File operation take place through the name node only. It is the most important part of HDFS; its failure leads to crash of whole HDFS.

**Data Node** - It is the node where actual data in the form of block is stored. Two or three or may be more data node can store the same block to allow the redundancy. It leads to data availability even if one of the datanode fails. Data node periodically updates the namenode about their availability so that namenode is aware about the health of the HDFS cluster.

**Block Size** - HDFS is for storing file of big size in distributed fashion. HDFS stores file in the form of blocks. It is the minimum amount of data that HDFS can read or write. The default block size of file is 64MB in Hadoop 1.x while it is 128MB in Hadoop 2.x.

**2**. **MapReduce** - It is the processing engine of Hadoop [13]. MapReduce is the programming paradigm for implementing business logic in the key value pair format. Key value pair is the core of MapReduce programming. Typically it has Mapper and Reducer but it has intermediate steps also which user may rewrite to override their default behaviour. Mapper - User data is divided into chunks or blocks of size 128MB each. For each block, a separate Mapper is executed to process that block. Mapper transforms and or filters the input key value into another key value pair(s). Partitioner- It categorizes the output key value of Mapper and pushes it to a particular Reducer. Programmer can implement the custom partitioner to override the default HashPartitioner which work on the hash value of a key. Combiner - If the operation in the reducer is associative and commutative then at local Mapper level combiner can run which does exactly the same job as the reducer would do but at smaller data (data from only that Mapper) set so that load on final Reducer is reduced. It is optional but if used appropriately can help in great performance boost and saving of network traffic. Shuffle and Sort - Before key value is given to the Reducer, the pair is shuffled and sorted so that data goes to a proper reducer in a sorted format with respect to a key which simplifies the task of Reducer. Shuffle and sort are done internally by the Hadoop framework. Reducer - Reducer does the reduction or aggregation job from the key value generated from the Mapper to produce the final result.

3. **YARN** - It is Yet Another Resource Negotiator. It takes the responsibility of resource management from Map Reduce engine which was there in Hadoop 1.x. It gives the separation between job execution and resource management. It also enables the platform to run another type of programming model like MPI to be executed on top of it. It has following main components. Resource Manager - It is responsible for the overall management of the cluster. Node Manager - It is responsible for management of the particular node. Based on the request of Application Master it reserves and gives the resources to the Application master. Periodically it updates its information to the Resource Manager. Application Master - When any job runs on the YARN cluster the Resource Manager allocates one Node Manager where Application Master can be scheduled to run. Application Master is responsible for whole resource management for that particular job with the help of Resource Manager.

**C. Spark**

Apache spark is fast and general purpose engine for large scale data processing [14-15]. Architecture of spark has spark core at it bottom and on top of which Spark SQL, MLlib, Spark streaming and GraphX libraries are provided for data processing. Apache Spark is very good for in memory computing. Spark has its own cluster management but it can work with Hadoop also.

There are three core building blocks of Spark programming. Resilient Distributed Datasets (RDD), Transformations and Action.RDD is an immutable data structure on which various transformations can be applied. After transformation any action on RDD can lead to complete lineage execution of transformation before result is produced.

## 2.2 PROPOSED SYSTEM:

The proposed model will use linear regression, which will predict the high and low temperatures as a linear combination of all the features. Linear regression does not use weather classification data of each day because this algorithm cannot be used with classification data. Therefore initially in our project only eight parameters are selected for use which is maximum temperature, minimum temperature, mean humidity, and mean atmospheric.

The second algorithm to be used is a type of functional regression. It looks for historical weather patterns which are similar to the present day weather patterns, and then it predicts the future weather condition based upon the data of the historical weather patterns. The architecture of the proposed model is given in Fig.1. A Neural network model is used to train 80% of data values and remaining 20% for testing.

**Algorithm**

1. Train model with x(i) where x(i)∈ W. W={ mxT, mnT, meanH, meanAP} of past few years

2. Evaluate and optimize the model with test set

3. a. 4-fold cross validate the model with blind set b. Minimize the cost function

4. Input past two days mxT and mnT (after the model is ready)

5. Predict the mxT and mnT for the next day

\*mxT=maximum temperature, mnT=minimum temperature, meanH=mean humidity, meanAP= mean atmospheric pressure.

**RELATED WORK**

Related works included many different and interesting techniques to try to perform weather forecasts. While much of current forecasting technology involves simulations based on physics and differential equations, many new approaches from artificial intelligence used mainly machine learning techniques, mostly neural networks while some drew on probabilistic models such as Bayesian networks. Out of the three papers on machine learning for weather prediction we examined, two of them used neural networks while one used support vector machines. Neural networks seem to be the popular machine learning model choice for weather forecasting because of the ability to capture the non-linear dependencies of past weather trends and future weather conditions, unlike the linear regression and functional regression models that we used. This provides the advantage of not assuming simple linear dependencies of all features over our models. Of the two neural network approaches, one used a hybrid model that used neural networks to model the physics behind weather forecasting while the other applied learning more directly to predicting weather conditions. Similarly, the approach using support vector machines also applied the classifier directly for weather prediction but was more limited in scope than the neural network approaches. Other approaches for weather forecasting included using Bayesian networks. One interesting model used Bayesian networks to model and make weather predictions but used a machine learning algorithm to find the most optimal Bayesian networks and parameters which was quite computationally expensive because of the large amount of different dependencies but performed very well. Another approach focused on a more specific case of predicting severe weather for a specific geographical location which limited the need for fine tuning Bayesian network dependencies but was limited in scope.

**DATASET AND FEATURES**

The maximum temperature, minimum temperature, mean humidity, mean atmospheric pressure, and weather classification for each day in the years 2011-2015 for Stanford, CA were obtained from Weather Underground. Originally, there were nine weather classifications: clear, scattered clouds, partly cloudy, mostly cloudy, fog, overcast, rain, thunderstorm, and snow. Since many of these classifications are similar and some are sparsely populated, these were reduced to four weather classifications by combining scattered clouds and partly cloudy into moderately cloudy; mostly cloudy, foggy, and overcast into very cloudy; and rain, thunderstorm, and snow into precipitation. The data from the first four years were used to train the algorithms, and the data from the last year was used as a test set.

**METHODS**

The first algorithm that was used was linear regression, which seeks to predict the high and low temperatures as a linear combination of the features. Since linear regression cannot be used with classification data, this algorithm did not use the weather classification of each day. As a result, only eight features were used: the maximum temperature, minimum temperature, mean humidity, and mean atmospheric pressure for each of the past two days. Therefore, for the i-th pair of consecutive days, x (i) ∈ R 9 is a nine-dimensional feature vector, where x0 = 1 is defined as the intercept term. There are 14 quantities to be predicted for each pair of consecutive days: the high and low temperatures for each of the next seven days. Let y (i) ∈ R 14 denote the 14-dimensional vector that contains these quantities for the i-th pair of consecutive days. The prediction of y (i) given x (i) is hθ(x (i) ) = θ T x, where θ ∈ R 9×14. The cost function that linear regression seeks to minimize is

**J(θ) = 1 2 Xm i=1 khθ(x (i) ) − y (i) k 2** , .

where m is the number of training examples. Letting X ∈ R m×9 be defined such that Xij = x (i) j and Y ∈ R m×14 be defined such that Yij = y (i) j , the value of θ that minimizes the cost in equation 1 is

**θ = (XT X) −1XT Y**.

The second algorithm that was used was a variation of functional regression, which searches for historical weather patterns that are most similar to the current weather patterns, then predicts the weather based upon these historical patterns. Given a sequence of nine consecutive days, define its spectrum f as follows. Let f(1), f(2) ∈ R 5 be the feature vectors for the first day and the second day, respectively. For i in the range 3 to 9, let f(i) ∈ R 2 be a vector containing the maximum temperature and the minimum temperature for the i-th day in the sequence. Then define a metric on the space of spectra

**d(f1, f2) = X 2 j=1 h w11[f1(j)1 6= f2(j)1] + X 5 k=2 wk f1(j)k − f2(j)k 2 i ,**

where w is a weight vector that assigns weights to each feature. Since the first feature is the weather classification and the difference between classifications is meaningless, the squared difference has been replaced by an indicator function of whether the classifications are different. Define a kernel

**ker(t) = max{1 − t, 0}**,

and let neighk (f) denote the k indices i ∈ {1, . . . , m} of the k spectra in the training set that are the closest to f with respect to the metric d. That is,

**d(f (i) , f) < d(f (j) , f)**

for all i ∈ neighk (f) and j 6∈ neighk (f), and |neighk (f)| = k. Furthermore, define

**h = max i∈{1,...,m} d(f (i) , f).**

Then, given the values f(1), f(2) of the first two days of a spectrum f, the remainder of the spectrum f(i) for i in the range 3 to 9 can be predicted as

**ˆf(i) = P j∈neighk(f) ker(d(f (j) , f)/h)f (j) (i) P j∈neighk(f) ker(d(f (j) , f)/h)** .

The error of the estimator

**ˆf is defined to be Error = X 9 i=3 k ˆf(i) − f(i)k 2** .

A more useful error that will be used in lieu of this is the root mean square (rms) error, which is defined to be

**Errorrms = vuutX 9 i=3 k ˆf(i) − f(i)k 2 14** ,

and provides the standard deviation of the individual error terms.

# 3. REQUIRIMENT ANALYSIS

## 3.1 HARDWARE SPECIFICATION:

* Processor : Intel I5.
* Hard Disk : 120 GB.
* Monitor : 15’’ LED
* Input Devices : Keyboard, Mouse
* Ram : 4GB.

## 3.2 SOFTWARE SPECIFICATION:

* Operating Systems : Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
* Python : 2.7.x
* R-studio : 1.0.143
* R-Shiny :0.9.16

**3.3 FEASIBILITY STUDY**:

## Feasibility Analysis

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

* **Operational Feasibility**
* **Economic Feasibility**
* **Technical Feasibility**

## Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

## Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

## Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment as it is a web application.

# 4. IMPLEMENTATION

## 4.1 PROBLEM DEFINITION:

Weather prediction is a useful tool for informing populations of expected weather conditions. Weather prediction is a complex topic and poses significant variation in practice. The student will attempt to understand and implement a weather prediction application in attempts to investigate ways upon which this area could be improved.

## 4.2 EXPERIMENT:

Since weather forecasting inherently involves time series, k-fold cross-validation is a poor technique to analyze whether our model will generalize to an independent test set. Instead, a 4-fold forward chaining time-series cross validation was performed, wherein the test set consisted of the data from the year immediately following the training set, as in table II. This method more accurately models the weather at prediction time, since the model is based on past data and predicts on future data. A learning curve can also be generated, providing a useful gauge of the dependence of the model on the training set size.

With this in mind, the parameters of the functional regression model were chosen to minimize the rms error in equation 9 averaged over all 4 test sets in table II. The weights w2 = w3 = 1 in equation 3 were chosen since we believed that deviations in the maximum temperature and the minimum temperature should carry equal weight. Since the functional form of the estimator ˆf in equation 7 was too unwieldy to perform stochastic gradient descent on, an exhaustive grid search was instead performed to optimize the other weights w1, w4, w5. Alternating exhaustive grid searches over the weights w and the number of neighbor’s k were performed to optimize each of these values. An initial exhaustive grid search was performed with large increments to obtain crude estimates of these weights, with the values of each weight taken from the range 0-50 in increments of 10. The number of neighbors was taken to be k = 5. This yielded initial estimates of w1 = 20 and w4 = w5 = 0. w4 = 0 was the optimum weight of the mean humidity presumably since humidity correlates poorly with the maximum temperature and the minimum temperature, and humidity would be a more useful determinant of precipitation. w5 = 0 turned out to be the optimum weight of the mean atmospheric pressure since there were only small deviations in the atmospheric pressure which did not appear to be correlated with the maximum temperature and the minimum temperature. With this in mind, the mean humidity and the mean atmospheric pressure were removed as features.

The hyper parameter of the number of neighbor’s k was then chosen in a similar manner, with an exhaustive grid search over both constant values and values proportional to the data set size. Values of k in the range 5-50 in increments of 5 and values of k proportional to the data set size with proportionality constant in the range 0.05-0.50 in increments of 0.05 were considered. Taking k proportional to the data set size greatly outperformed taking k to be constant, and the optimum proportionality constant was 0.10. W1 and k were then fine-tuned together with one final exhaustive grid search, taking w1 from the range 15-25 in increments of 1, and the proportionality constant of k from the range 0.05-0.15 in increments of 0.05. This yielded a final value of w1 = 18 and k = 0.095|D|, where |D| is the number of data points.

**4.3 SYSTEM ARCHITECTURE**

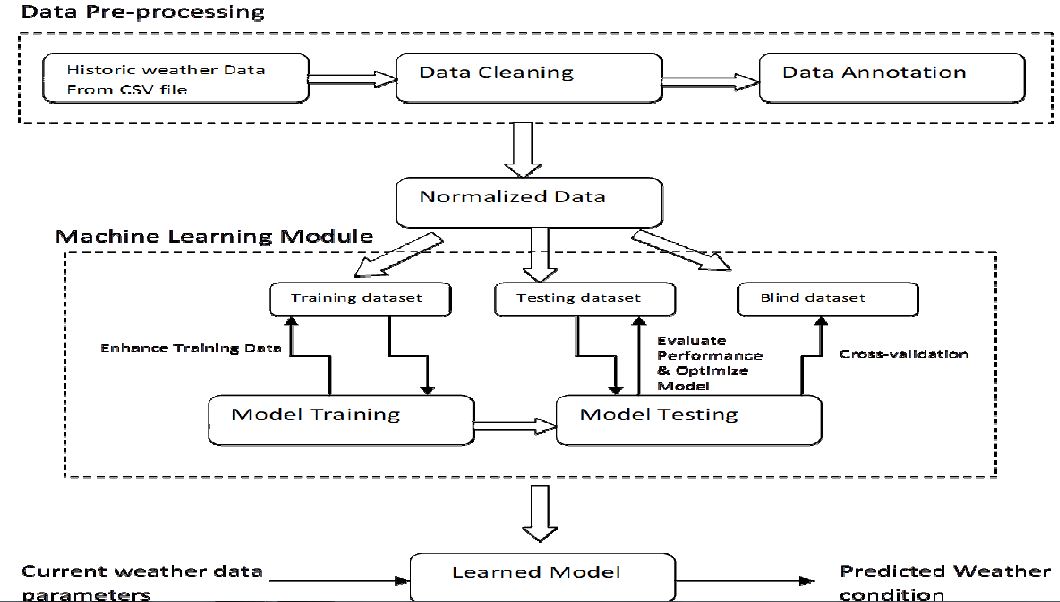


Fig: System Architecture

# 5. SYSTEM DESIGN

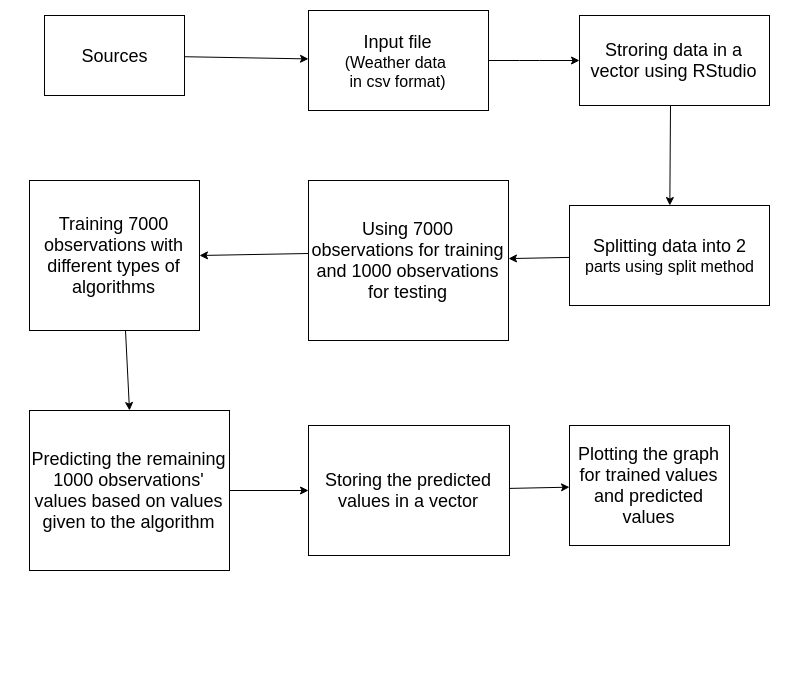


Fig: System flow architecture

This Fig shows the work flow of the input file work from source file selection to the creation and execution of R studio work.

# 6. UML DIAGRAMS

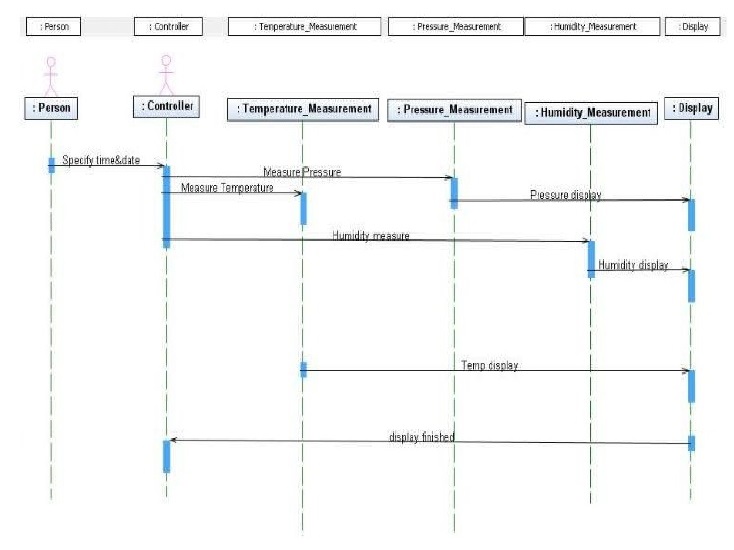
## UNIFIED MODELLING LANGUAGE:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

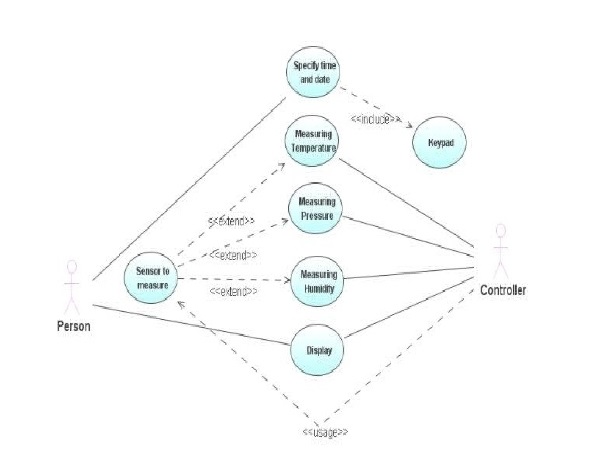
* **USER MODEL VIEW** 
  1. This view represents the system from the users perspective.
  2. The analysis representation describes a usage scenario from the end-users perspective.
* **STRUCTURAL MODEL VIEW** 
  1. In this model the data and functionality are arrived from inside the system.
  2. This model view models the static structures.
* **BEHAVIORAL MODEL VIEW**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

## Sequence Diagram



## Use Case Diagram



# 7. CODING

##### **SOURCE CODE:**

#----------- loading dataset ----------------------------------

data <- read.csv("~/Desktop/weather\_predict-master(1)/weather\_predict-master/pre\_data1.csv")

plot(data$Temp.)

library(caTools)

split <- sample.split(data, SplitRatio=0.8)

split

training <- subset(data,split=="TRUE")

testing <- subset(data,split=="FALSE")

#-------lm for temperature overall-----------

model <- lm(Temp. ~. , data = training)

summary(model)

res <- predict(model,testing)

actuals\_preds <- data.frame(cbind(actuals=testing$Temp., predicteds=res))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)

plot(res)

plot(testing$Temp.)

print(testing$Temp.)

#-------random forest for UV radiation ----------------------

library(randomForest)

model.uv <- randomForest(UV.Rad. ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.uv)

predTrain <- predict(model.uv, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$UV.Rad., predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #87% accuracy

plot(predTrain)

plot(testing$UV.Rad.)

#-------random forest for Rain --------------------------------

model.rain <- randomForest(Rain ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.rain)

predTrain <- predict(model.rain, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$Rain, predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #79% accuracy

plot(predTrain)

plot(testing$Rain)

#--------random forest for Temp ----------------------------------

model.temp <- randomForest(Temp. ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.temp)

predTrain <- predict(model.temp, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$Temp., predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #98% accuracy

plot(predTrain)

plot(testing$Temp.)

#--------- random forest for RH --------------------------------------

model.rh <- randomForest(RH ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.rh)

predTrain <- predict(model.rh, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$RH, predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #97.9% accuracy

plot(predTrain)

plot(testing$RH)

#--------- random forest for wind speed -----------------------------------

model.windspeed <- randomForest(Wind.Speed ~. ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.windspeed)

predTrain <- predict(model.windspeed, testing)

actuals\_preds <- data.frame(cbind(actuals=testing$Wind.Speed, predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #13.9% accuracy

plot(predTrain)

plot(testing$Wind.Speed)

#---------- random forest for Wind.Direction ---------------------------------

model.winddirection <- randomForest(Wind.Direction ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.winddirection)

predTrain <- predict(model.winddirection, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$Wind.Direction, predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #71.7% accuracy

plot(predTrain)

plot(testing$Wind.Direction)

#---------- random forest for Solar radiation ---------------------------------

model.solrad <- randomForest(Sol..Rad. ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.solrad)

predTrain <- predict(model.solrad, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$Sol..Rad., predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #93.9% accuracy

plot(predTrain)

plot(testing$Sol..Rad.)

#----------- random forest for atmospheric pressure ------------------------

model.atmpressure <- randomForest(Atm..Press. ~ date + Time ,data=training,ntree = 500, mtry = 6, importance = TRUE)

print (model.atmpressure)

predTrain <- predict(model.atmpressure, testing, type = "class")

actuals\_preds <- data.frame(cbind(actuals=testing$Atm..Press., predicteds=predTrain))

correlation\_accuracy <- cor(actuals\_preds)

print(correlation\_accuracy)  #98.8% accuracy

plot(predTrain)

plot(testing$Atm..Press.)

# 8. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

## Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

## System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

## White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

## Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

## 8.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

## Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail. **Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

## Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

## 8.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## 8.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered

# 9. SCREENSHOTS

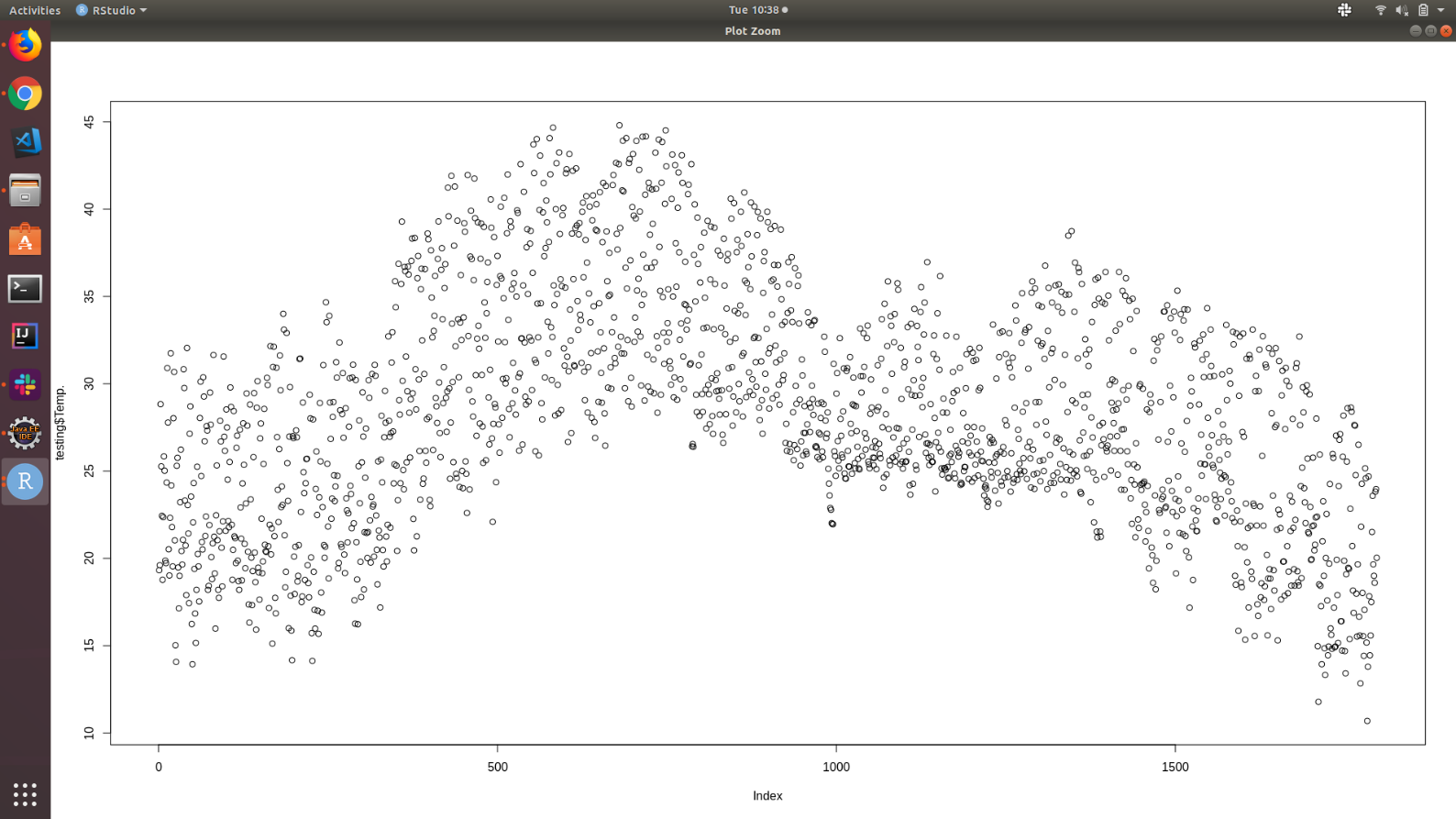


Fig: actual temperature

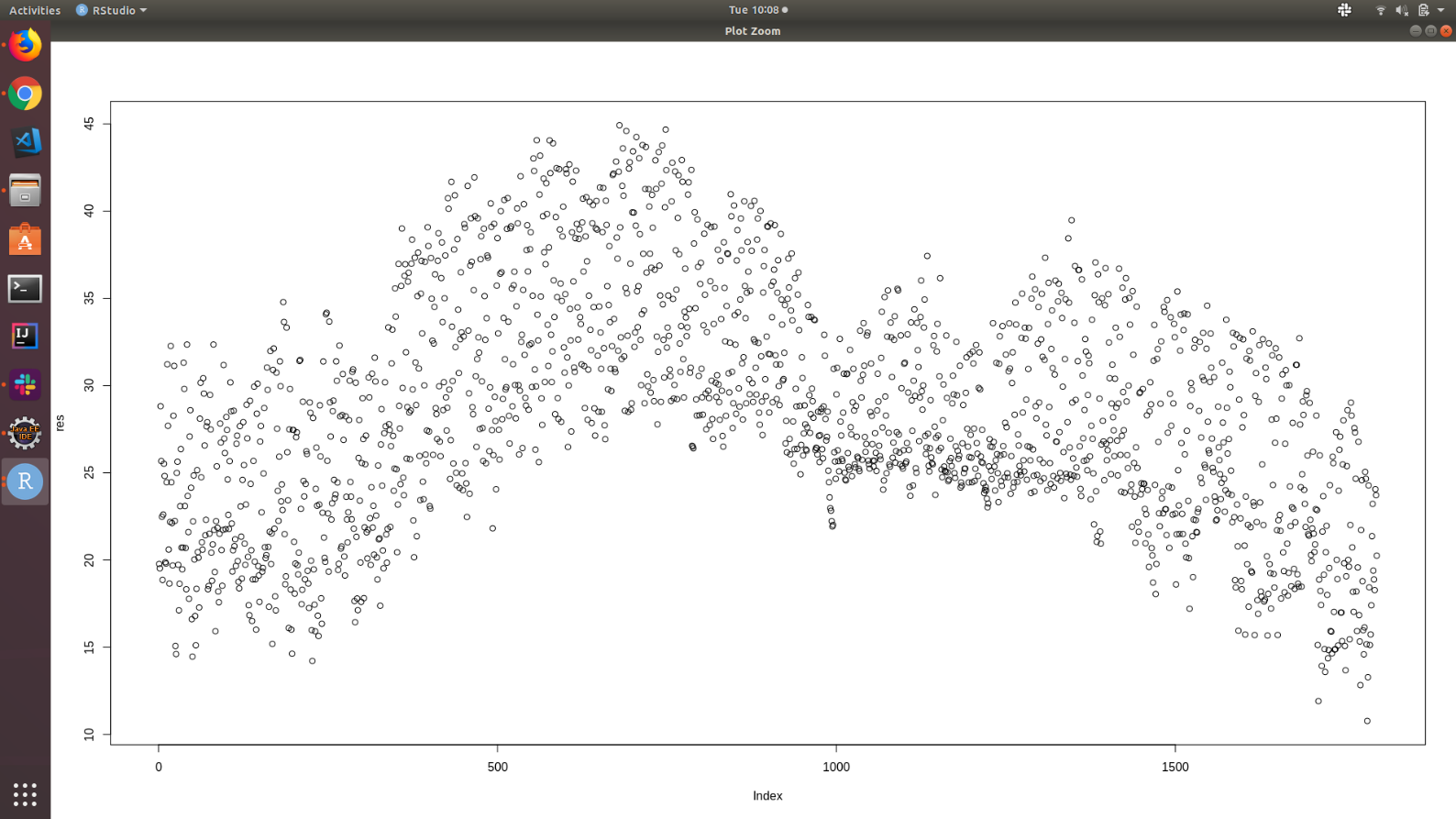


Fig:2 actual temp

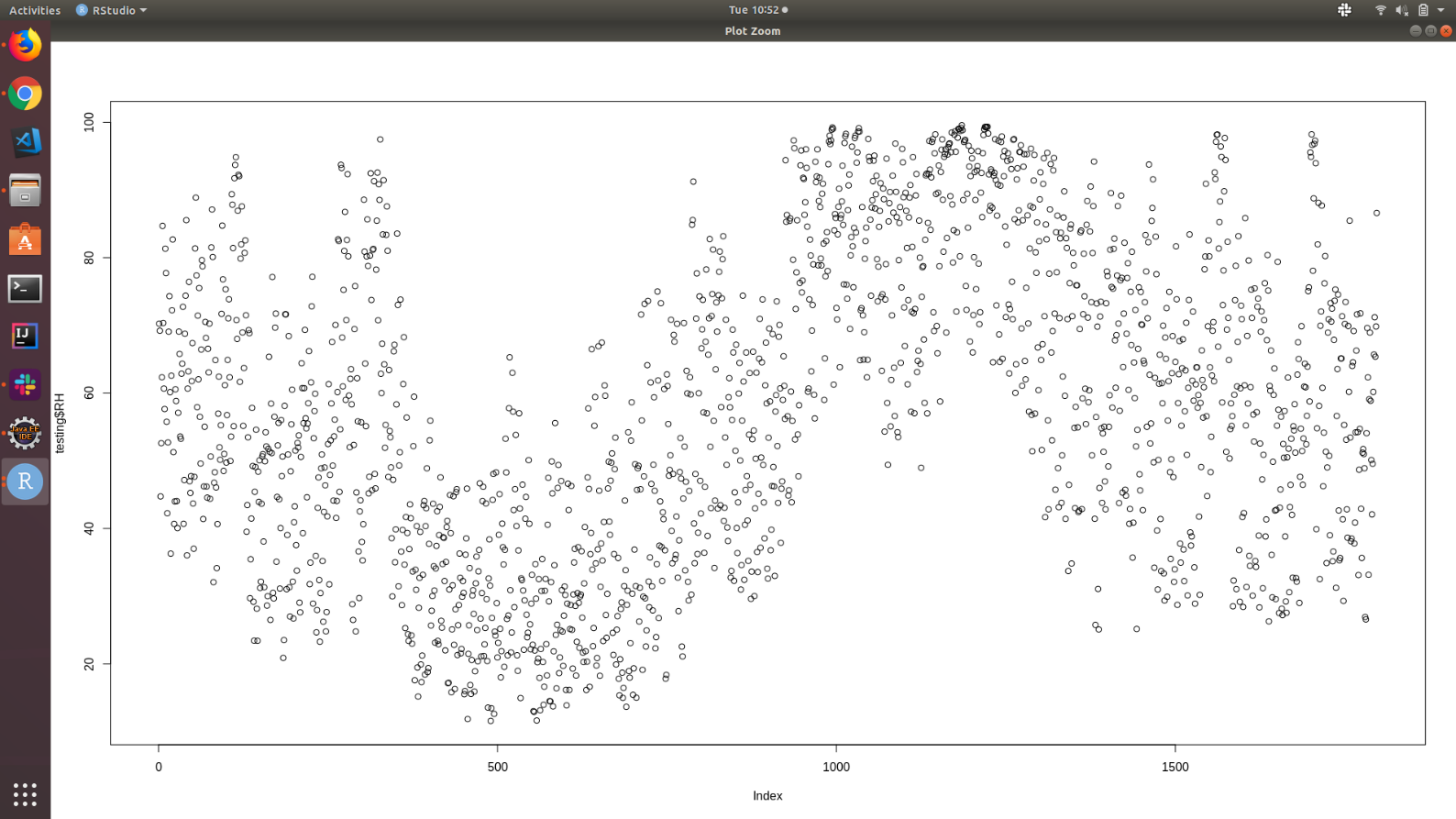


Fig :3 actualtempusingr

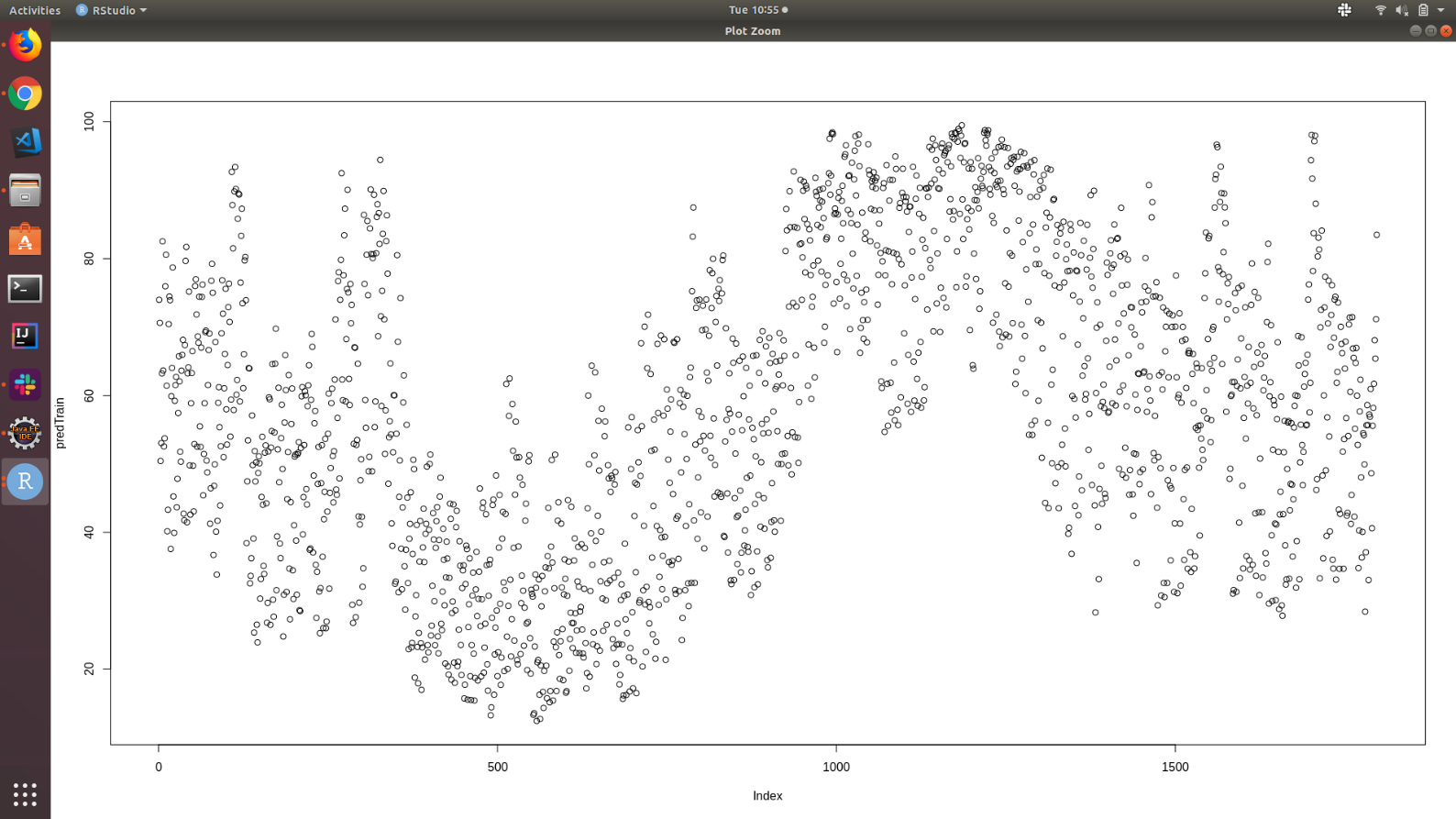
****

Fig: predicttempusingr

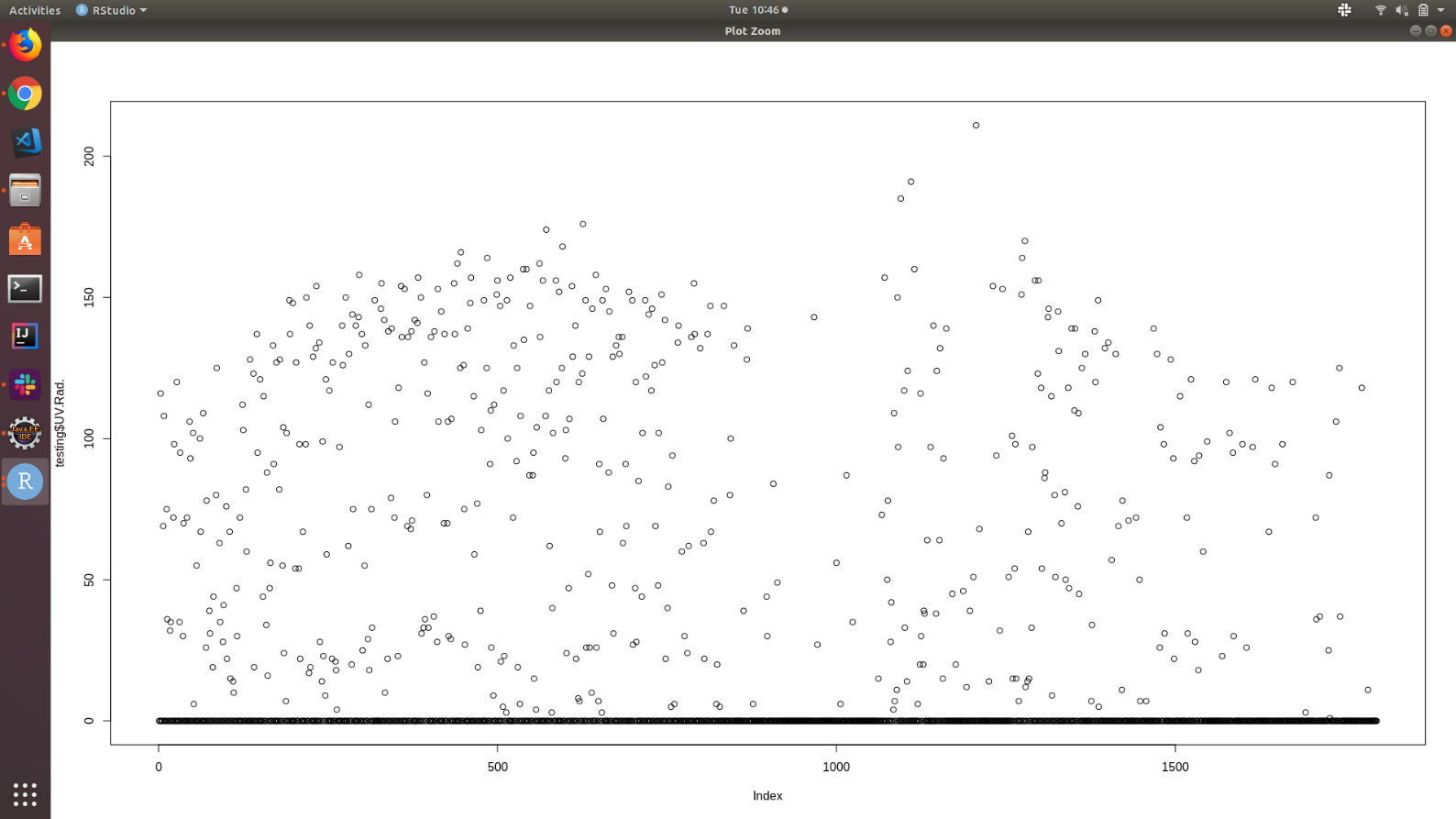


Fig : actualuv

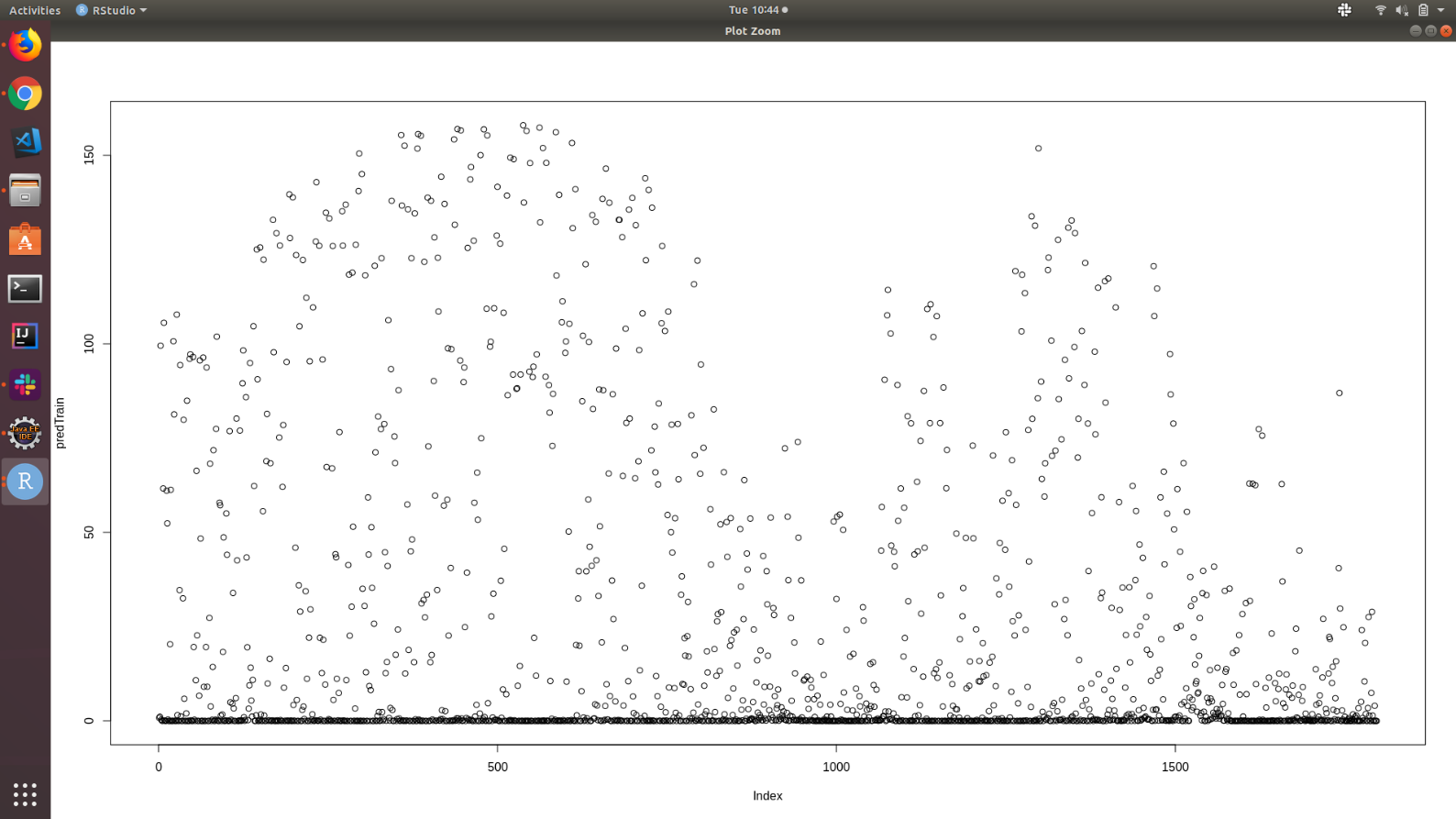


Fig :predictuvtemp.

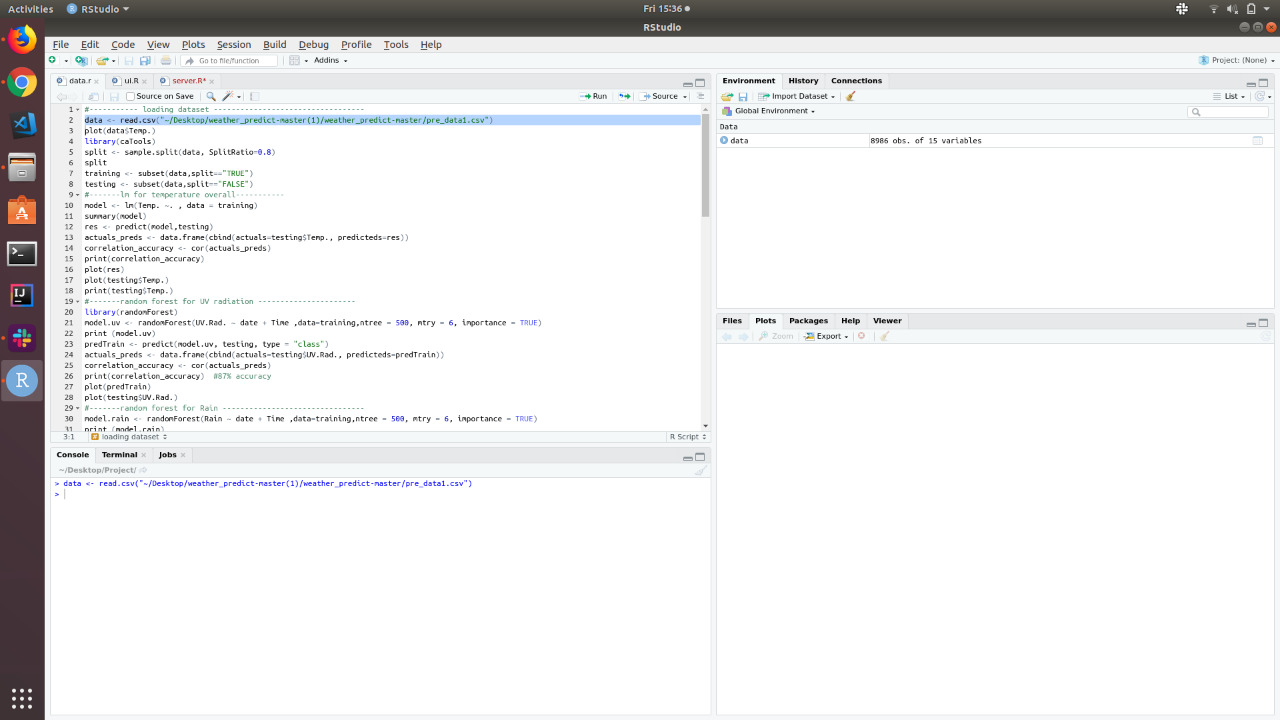


Fig: data <- read.csv("~/Desktop/weather\_predict-master(1)/weather\_predict-master/pre\_data1.csv")

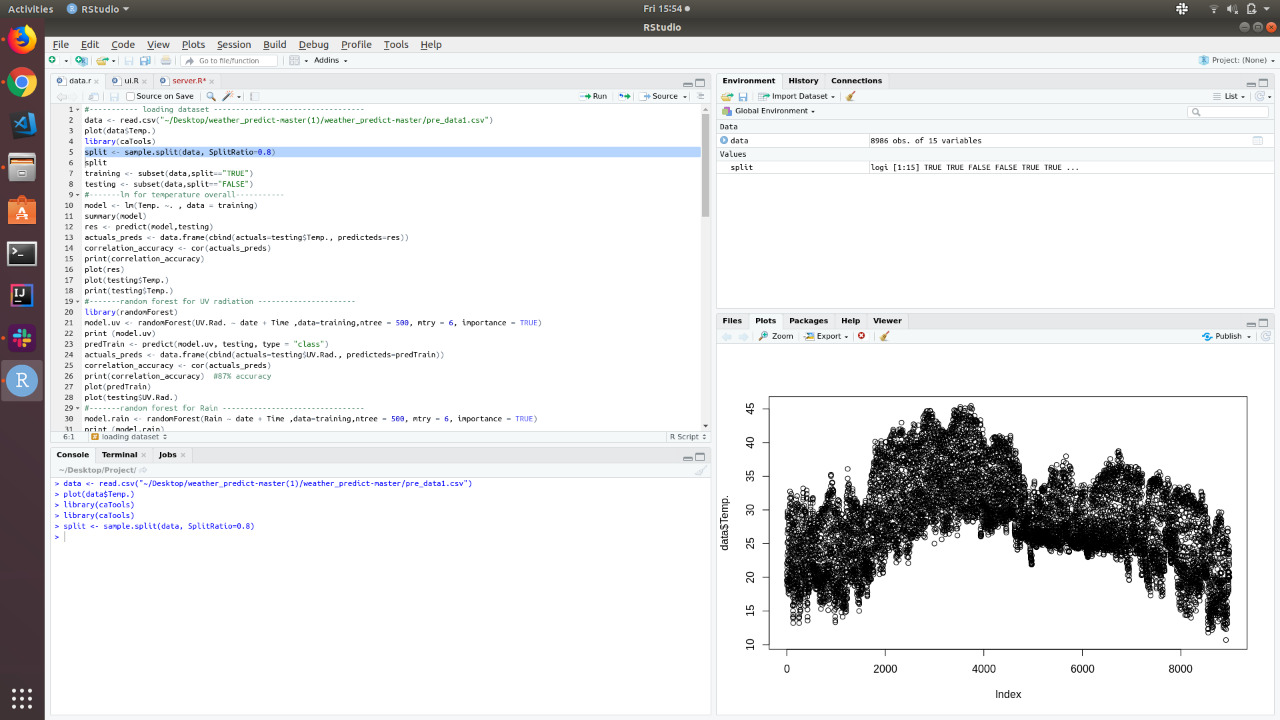


Fig: split <- sample.split(data, SplitRatio=0.8)

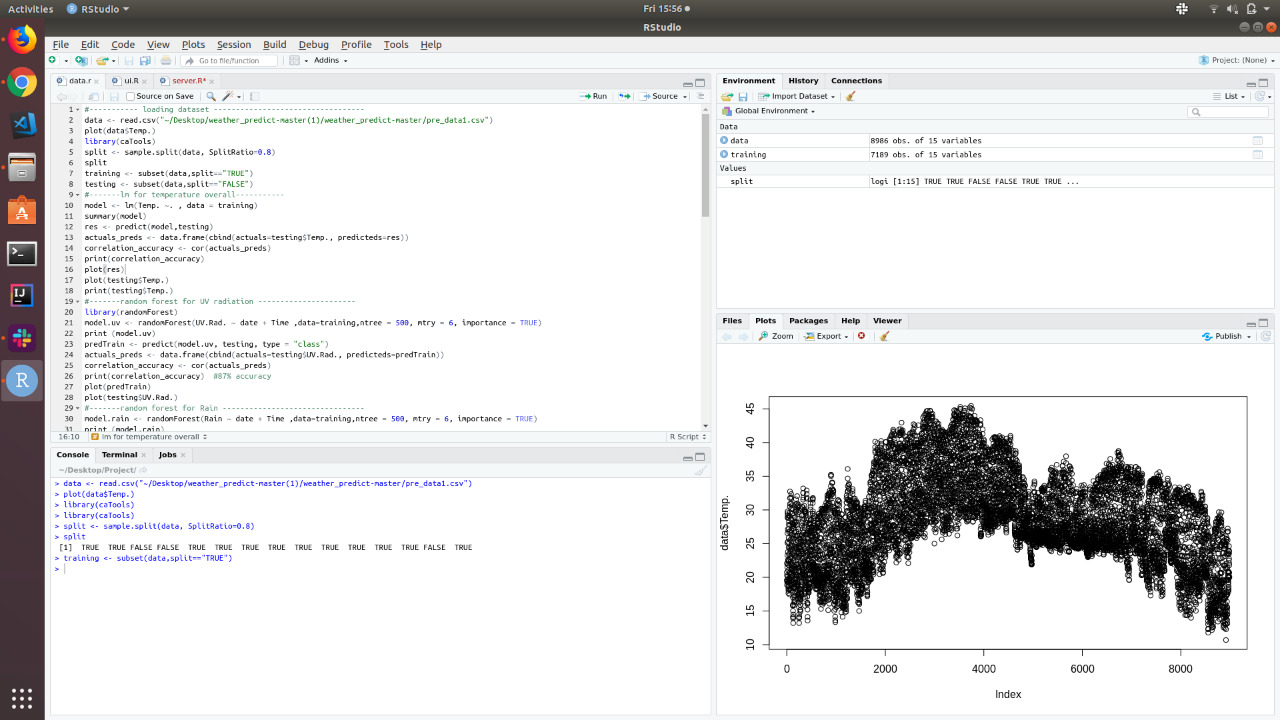


Fig: training <- subset(data,split=="TRUE")

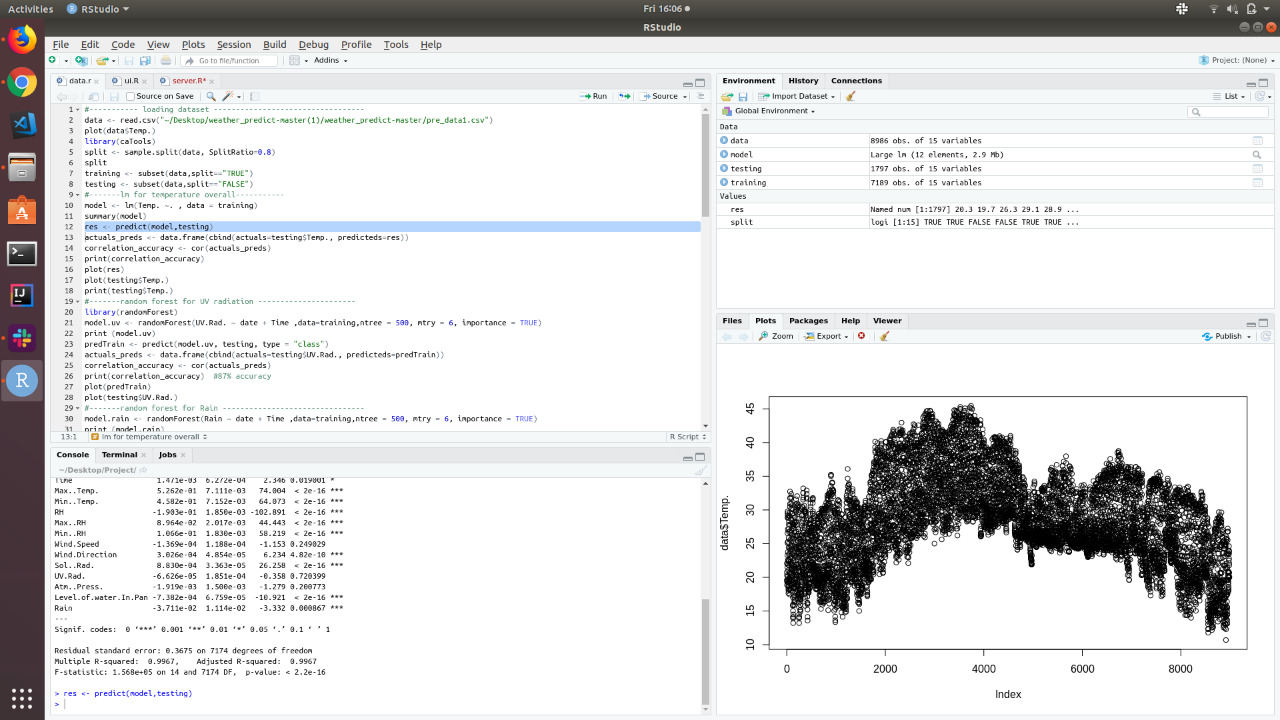


Fig: res <- predict(model,testing)

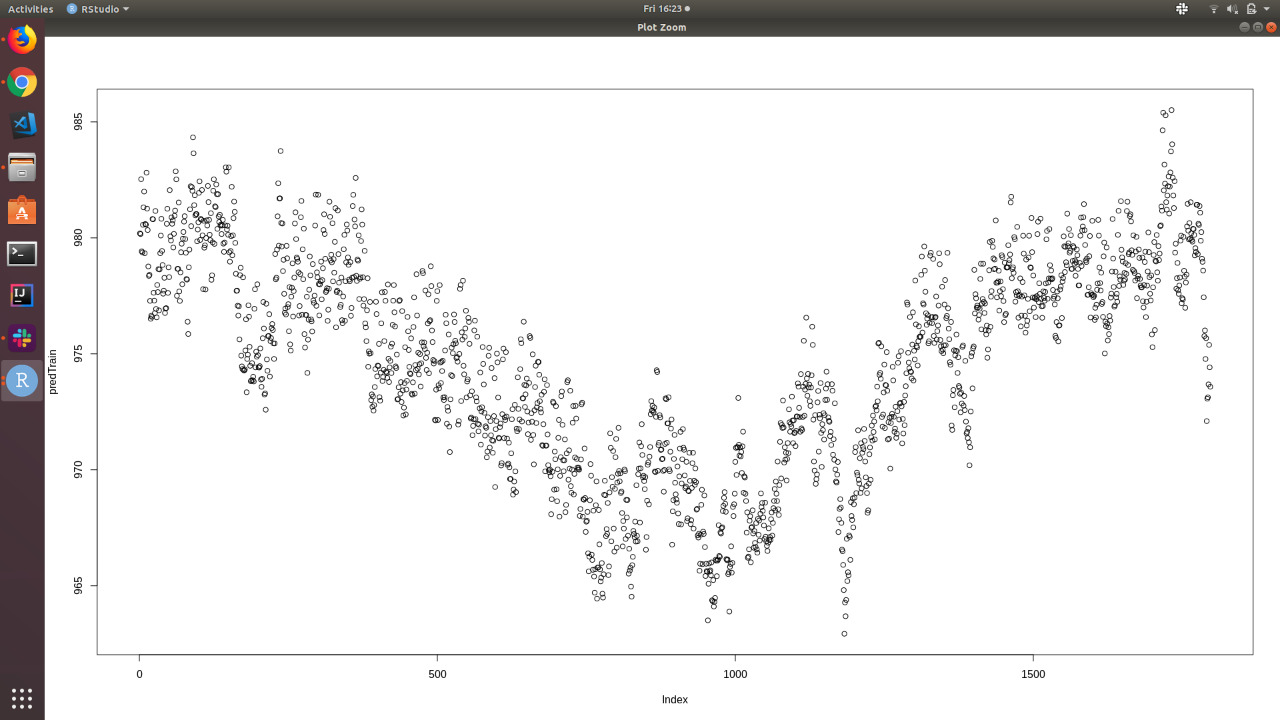


Fig: correlation\_accuracy <- cor(actuals\_preds)

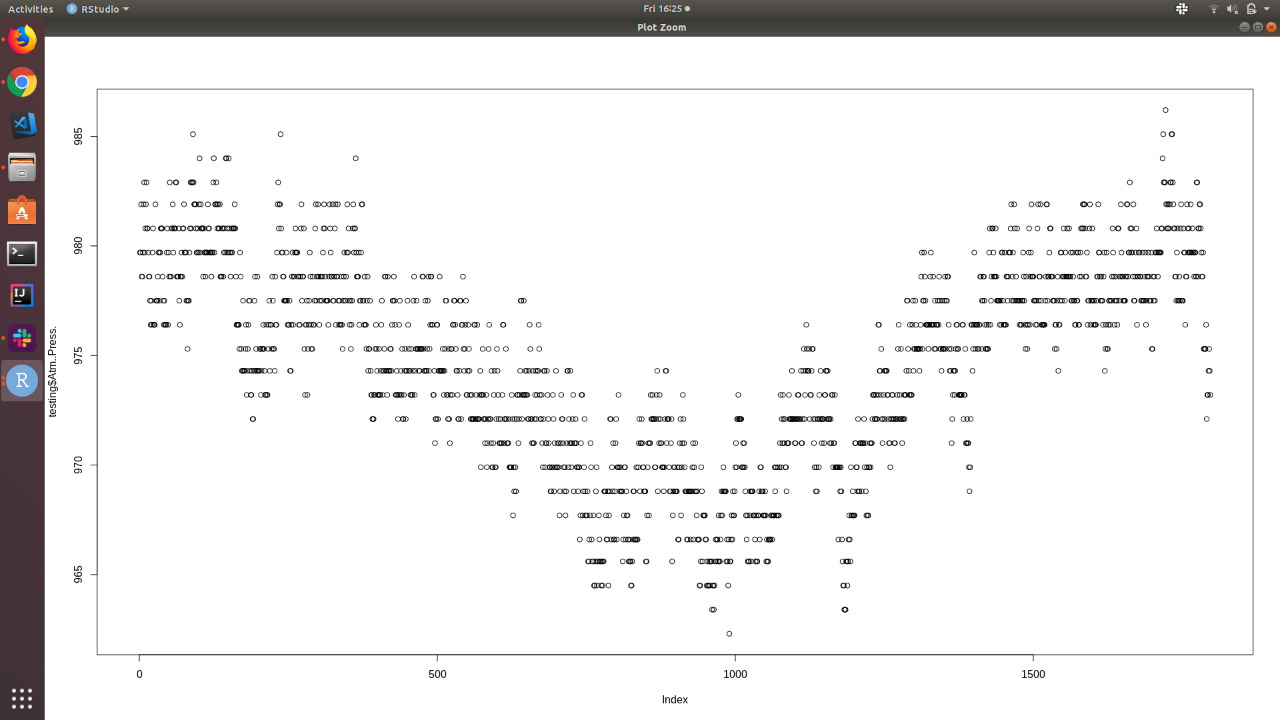


Fig: plot(testing$Atm..Press.)

# 10. FUTURE ENHANCEMENT

In this work, artificial neural networks are used to predict temperature. Four separate models were trained to predict the temperature 1, 3, 6, and 12 hours ahead. In the first experiment, only temperature was used as input to the networks. This constitutes an auto-regressive neural network (AR-NN). In the second experiment, precipitation data was introduced into the network, forming an autoregressive neural network with exogenous input (ARXNN). After extensive tuning of hyper parameters for all eight models, the prediction results of the models were compared. Introducing precipitation as an input in the ARX model was shown to slightly improve the prediction performance.

Hence, it may be interesting to extend the model with other inputs. Mainly, it is of interest to study whether introduction of data from other geographical locations can improve the prediction results. Based on knowledge of how the jet stream moves and influences the weather, together with local pressure variations, it would be natural to add weather information from, e.g., Kristiansand, Oslo, etc. as exogenous inputs. This will be a topic for future research.

# 11. CONCLUSION

In our work, both linear regression and functional regression are used to predict the weather parameters. The same dataset is used in both the algorithms, so that a comparative analysis could be made. Forecasting weather parameters for a longer duration with more parameters involves the use of artificial Neural Network. To check the validation of prediction model for weather conditions, both systems are compared to check the fitness of applicability. The future work involves building a prediction model using Deep Neural Network

# 12. BIBLIOGRAPHY&REFERENCES

1. https://www.tutorialspoint.com/
2. http://hadooptutorials.co.in/tutorials/hadoop/internals-of-hdfs-file-read- operations.html
3. http://www.hadooptpoint.com/hadoop-hive-architecture/
4. http://downloads.vmware.com/d/info/desktop\_downloads/vmware\_workstation/7\_0
5. http://www.cloudera.com/
6. https://dblp.uni-trier.de/xml/