VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590018

FOUNDATIONS OF DATA SCIENCE

Activity – 1: Introduction to R programming language

Submitted by:

SHARATH KUMAR V	1SI19CS104
SHASHWATH	1SI19CS106
SHREYANK	1SI19CS111

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR - 572103

(An Autonomous Institution Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi, Accredited by NAAC with 'A' grade & ISO 9001:2015 Certified)

2021-2022

What is R?

R is a programming language and software environment for statistical analysis, graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

Why use R?

- It is a great resource for data analysis, data visualization, data science and machine learning.
- It provides many statistical techniques (such as statistical tests, classification, clustering and data reduction).
- It is easy to draw graphs in R, like pie charts, histograms, box plot, scatter plot, etc.
- It works on different platforms (Windows, Mac, Linux).
- It is open-source and free.
- It has a large community support.
- It has many packages (libraries of functions) that can be used to solve different problems.

These R programming topics have been used in the following program:

- R Control Flow: **if-else**

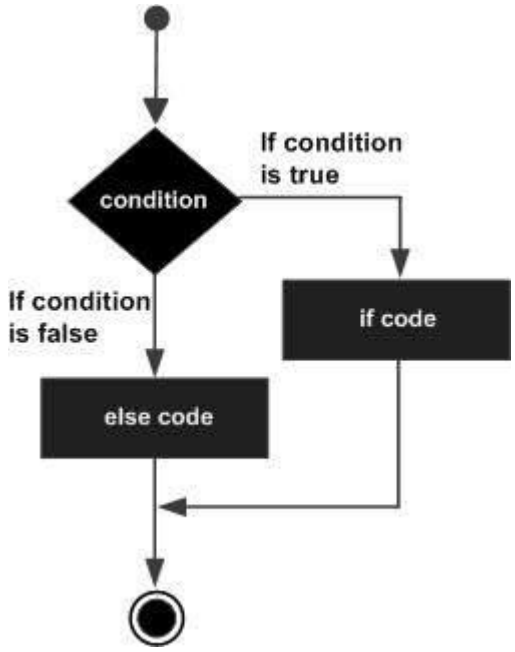
  An **if** statement can be followed by an optional **else** statement which executes when the Boolean expression is false.

Syntax:

```
if(boolean_expression) {

// statement(s) will execute if the Boolean expression is true.

} else {

// statement(s) will execute if the Boolean expression is false.

}
```

If the Boolean expression evaluates to be **true**, then the **if block** of code will be executed, otherwise **else block** of code will be executed.

Flow Diagram



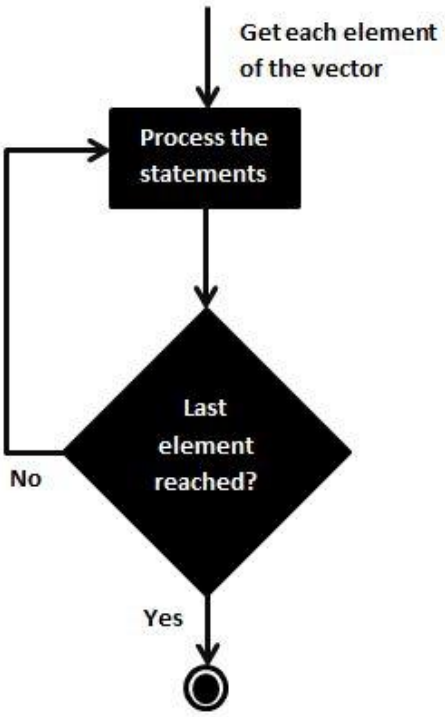
- R Control Flow: **for loop**

  A **for loop** is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for (value in vector) {
statements
}
```

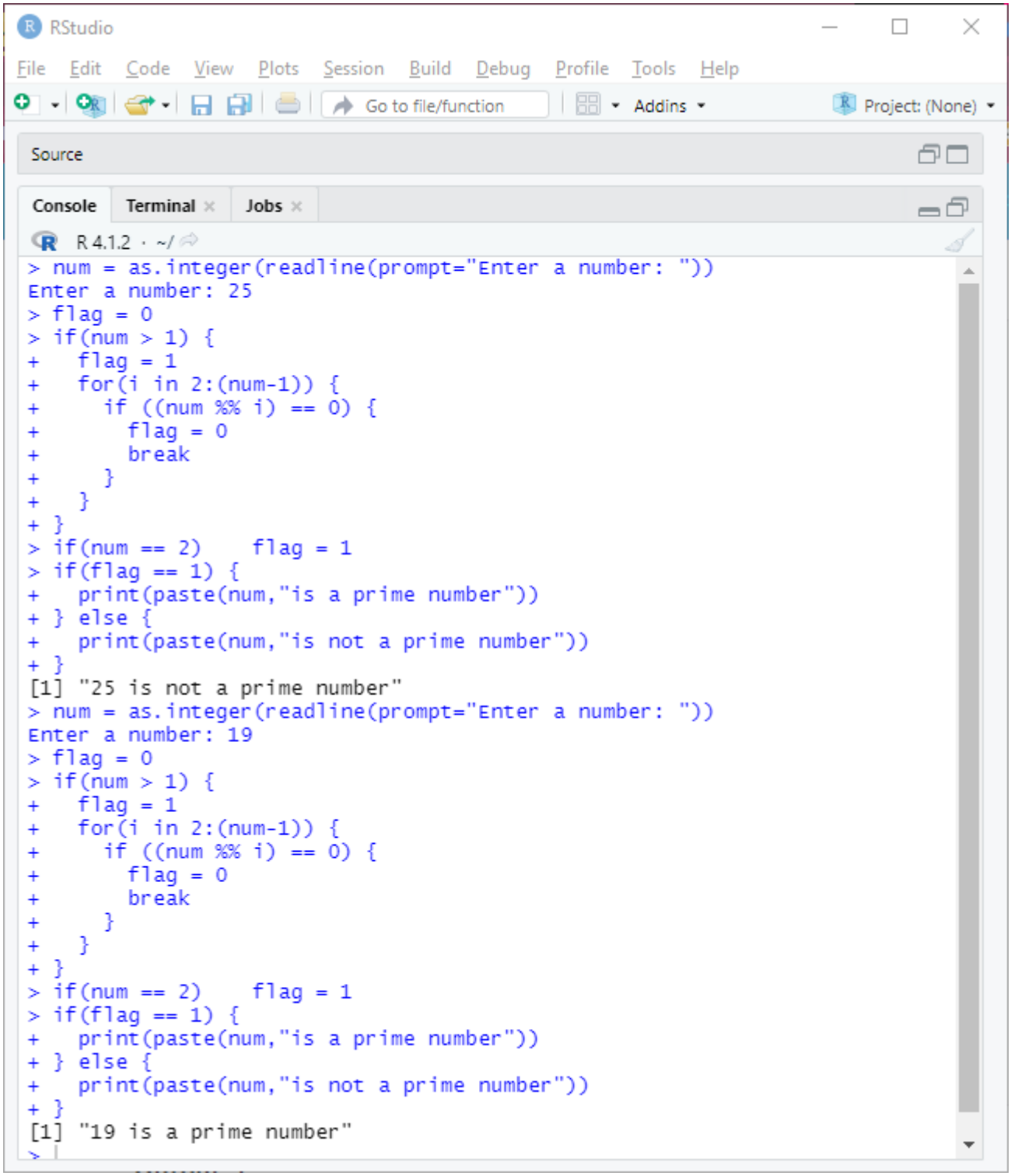
Flow Diagram



PROGRAM: To check weather a given number is prime or not using R language.

```
# Program to check if the input number is prime or not
# take input from the user
num = as.integer(readline(prompt="Enter a number: "))
flag = 0
# prime numbers are greater than 1
if(num > 1) {
        # check for factors
        flag = 1
        for(i in 2:(num-1)) {
                if ((num %% i) == 0) {
                        flag = 0
                        break
                }
        }
}
if(num == 2)    flag = 1
if(flag == 1) {
        print(paste(num,"is a prime number"))
} else {
        print(paste(num,"is not a prime number"))
}
```

OUTPUT:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console
Terminal
Jobs
R 4.1.2 · ~/
> num = as.integer(readline(prompt="Enter a number: "))
Enter a number: 25
> flag = 0
> if(num > 1) {
+ flag = 1
+ for(i in 2:(num-1)) {
+ if ((num %% i) == 0) {
+ flag = 0
+ break
+ }
+ }
+ }
> if(num == 2) flag = 1
> if(flag == 1) {
+ print(paste(num,"is a prime number"))
+ } else {
+ print(paste(num,"is not a prime number"))
+ }
[1] "25 is not a prime number"
> num = as.integer(readline(prompt="Enter a number: "))
Enter a number: 19
> flag = 0
> if(num > 1) {
+ flag = 1
+ for(i in 2:(num-1)) {
+ if ((num %% i) == 0) {
+ flag = 0
+ break
+ }
+ }
+ }
> if(num == 2) flag = 1
> if(flag == 1) {
+ print(paste(num,"is a prime number"))
+ } else {
+ print(paste(num,"is not a prime number"))
+ }
[1] "19 is a prime number"

INRODUCTION

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Ancient weather forecasting methods usually relied on observed patterns of events, also termed pattern recognition. For example, it might be observed that if the sunset was particularly red, the following day often brought fair weather. However, not all of these predictions prove reliable.

Here this system will predict weather based on parameters such as temperature, humidity and wind. User will enter current temperature; humidity and wind, System will take this parameter and will predict weather (rainfall in inches) from previous data in database(dataset). The role of the admin is to add previous weather data in database, so that system will calculate weather (estimated rainfall in inches) based on these data. Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

Data Warehousing

Data Warehouse is electronic storage of a large amount of information by a business which is designed for query and analysis instead of transaction processing. It is a process of transforming data into information and making it available to users for analysis.

Data Mining

Data mining is looking for hidden, valid, and potentially useful patterns in huge data sets. Data Mining is all about discovering unsuspected/ previously unknown relationships amongst the data. It is a multi-disciplinary skill that uses machine learning, statistics, AI and database technology.

It is important to exactly determine the rainfall for effective use of water resources, crop productivity and pre-planning of water structures.

Logistic Regression

Logistic regression is a classification model in machine learning. It uses probabilistic estimations which helps in understanding the relationship between the dependent variable and one or more independent variables. The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The general mathematical equation for logistic regression is −

y = 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

Following is the description of the parameters used −

y is the response variable.

x is the predictor variable.

a and **b** are the coefficients which are numeric constants.

The function used to create the regression model is the **glm()** function.

Syntax

The basic syntax for **glm()** function in logistic regression is −

glm(formula,data,family)

KNN

K-Nearest Neighbor or K-NN is a Supervised Non-linear classification algorithm. K-NN is a Non-parametric algorithm i.e. it doesn't make any assumption about underlying data or its distribution. It is one of the simplest and widely used algorithm which depends on its k value(Neighbors) and finds it's applications in many industries like finance industry, healthcare industry etc.

In the KNN algorithm, K specifies the number of neighbors and its algorithm is as follows:

- Choose the number K of neighbor.
- Take the K Nearest Neighbor of unknown data point according to distance.
- Among the K-neighbors, Count the number of data points in each category.
- Assign the new data point to a category, where you counted the most neighbors.
- For the Nearest Neighbor classifier, the distance between two points is expressed in the form of Euclidean Distance.

Decision Tree

Decision Trees are non-parametric supervised learning method used a for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Data Set

The dataset is a public weather dataset available on Kaggle.

Columns:

MaxTemp -

High temperature, in degrees Fahrenheit

MinTemp -

Low temperature, in degrees Fahrenheit

Humidity9am-

Humidity3pm-

Windspeed9am-

wind speed at 9am, in miles per hour

Windspeed3pm-

wind speed at 3pm, in miles per hour

WindGustMPH-

Highest wind speed gust, in miles per hour

Pressure9am-

Pressure3pm-

Rainfall-

Total precipitation, in inches ('T' if trace)

Downloading required packages

install.packages(mice)

install.packages(caret)

install.packages(outliers)

install.packages(e1071)

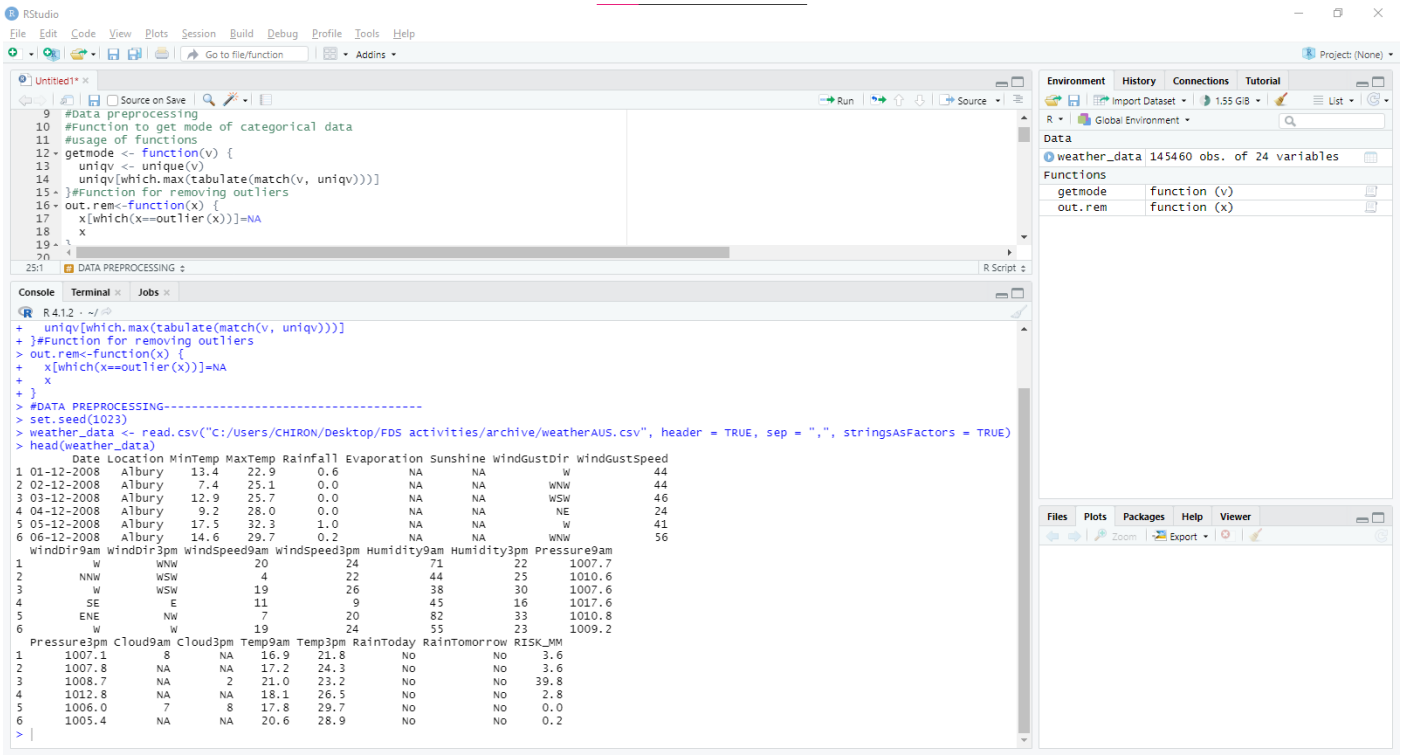
install.packages(moderndive)

install.packages(effects)

Data gathering and pre - processing

1. Collection of dataset:

The dataset was collected from www.kaggle.com , which consists of 145,460 tuples and 24 columns. The dataset was imported as a csv file into Rstudio.



2. DATA Processing

Now we are going to take a subset from our dataset by eliminating some columns. And only included participants for which we have no missing data. So reducing to 56,466 tuples.

```
> weather_data2 <- subset(weather_data, select = -c(Date, Location, RISK_MM, Rainfall, RainToday))
> colnames(weather_data2)
 [1] "MinTemp"       "MaxTemp"       "Evaporation"   "Sunshine"      "WindGustDir"   "WindGustSpeed" "WindDir9am"    "WindDir3pm"
 [9] "WindSpeed9am"  "WindSpeed3pm"  "Humidity9am"   "Humidity3pm"   "Pressure9am"   "Pressure3pm"   "Cloud9am"      "Cloud3pm"
[17] "Temp9am"       "Temp3pm"       "RainTomorrow"
> weather_data3 <- weather_data2[complete.cases(weather_data2),]
>
```

3. Feature extraction

```
>
> #2.Feature Extraction
> #Get the categorical variables
> #2.1Chi-Square to check whether the variables are dependent on RainTomorrow
> factor_vars1 <- names(which(sapply(weather_data3, class) == "factor"))
> factor_vars1
[1] "WindGustDir"  "WindDir9am"   "WindDir3pm"   "RainTomorrow"
> factor_vars1 <- setdiff(factor_vars1, "RainTomorrow")
> factor_vars1
[1] "WindGustDir" "WindDir9am"  "WindDir3pm"
> chisq_test_res <- lapply(factor_vars1, function(x) {
+   chisq.test(weather_data3[,x], weather_data3[, "RainTomorrow"], simulate.p.value = TRUE)
+ })
> names(chisq_test_res) <- factor_vars1
> chisq_test_res
$WindGustDir

        Pearson's Chi-squared test with simulated p-value (based on 2000 replicates)

data:  weather_data3[, x] and weather_data3[, "RainTomorrow"]
X-squared = 744.49, df = NA, p-value = 0.0004998


$WindDir9am

        Pearson's Chi-squared test with simulated p-value (based on 2000 replicates)

data:  weather_data3[, x] and weather_data3[, "RainTomorrow"]
X-squared = 988.96, df = NA, p-value = 0.0004998


$WindDir3pm

        Pearson's Chi-squared test with simulated p-value (based on 2000 replicates)

data:  weather_data3[, x] and weather_data3[, "RainTomorrow"]
X-squared = 685.77, df = NA, p-value = 0.0004998
```

As we can see p-value for above attributes is less than 0.05 so we can say that they are dependent variables. This is all for string data. Now we take another subset and apply same for numeric attributes. Caret library was used to determine the correlation between numeric variables.

```
>
>
> #Remove Categorical variables from dataset
> weather_data4 <- subset(weather_data2, select = -c(WindDir9am, WindDir3pm))
> colnames(weather_data4)
 [1] "MinTemp"       "MaxTemp"       "Evaporation"   "Sunshine"      "WindGustDir"   "WindGustSpeed" "WindSpeed9am"  "WindSpeed3pm"
 [9] "Humidity9am"   "Humidity3pm"   "Pressure9am"   "Pressure3pm"   "Cloud9am"      "Cloud3pm"      "Temp9am"       "Temp3pm"
[17] "RainTomorrow"
> weather_data5 <- weather_data4[complete.cases(weather_data4),]
> numeric_vars <- setdiff(colnames(weather_data5), factor_vars1)
> numeric_vars <- setdiff(numeric_vars, "RainTomorrow")
> numeric_vars_mat <- as.matrix(weather_data5[, numeric_vars, drop=FALSE])
> numeric_vars_cor <- cor(numeric_vars_mat)
> #Get the correlation between the numeric variables
> library(caret)
> fndCorrelation = findCorrelation(numeric_vars_cor, cutoff=0.6) # putt any value as a "cutoff"
> fndCorrelation = sort(fndCorrelation)
> reduced_Data = numeric_vars_mat[,c(fndCorrelation)]
> cols=colnames(reduced_Data)
> cols
[1] "MaxTemp"       "Sunshine"      "WindGustSpeed" "Humidity9am"   "Pressure3pm"   "Cloud3pm"      "Temp9am"       "Temp3pm"
> summary (reduced_Data)
    MaxTemp         Sunshine      WindGustSpeed     Humidity9am      Pressure3pm        Cloud3pm         Temp9am         Temp3pm
 Min.   : 4.10   Min.   : 0.000   Min.   :  9.00   Min.   :  0.00   Min.   : 977.1   Min.   :0.000   Min.   :-0.90   Min.   : 3.70
 1st Qu.:18.60   1st Qu.: 5.000   1st Qu.: 31.00   1st Qu.: 55.00   1st Qu.:1010.1   1st Qu.:2.000   1st Qu.:12.90   1st Qu.:17.30
 Median :23.80   Median : 8.600   Median : 39.00   Median : 67.00   Median :1014.8   Median :5.000   Median :17.70   Median :22.30
 Mean   :24.13   Mean   : 7.699   Mean   : 40.55   Mean   : 66.22   Mean   :1014.9   Mean   :4.328   Mean   :18.09   Mean   :22.63
 3rd Qu.:29.60   3rd Qu.:10.700   3rd Qu.: 48.00   3rd Qu.: 80.00   3rd Qu.:1019.5   3rd Qu.:7.000   3rd Qu.:23.20   3rd Qu.:27.80
 Max.   :48.10   Max.   :14.500   Max.   :124.00   Max.   :100.00   Max.   :1038.9   Max.   :9.000   Max.   :39.40   Max.   :46.10
>
>
>
>
```

After that we will merge both as shown below,

```
> # Get the numeric and categorical variables
> library(dplyr)
> weather_data7= weather_data2[c("WindGustDir","WindDir9am","WindDir3pm","RainTomorrow")]
> weather_data9= weather_data2[c(cols)]
>
>
> #remove outliers
> library(outliers)
> apply(weather_data9,2,out.rem)
      MaxTemp Sunshine WindGustSpeed Humidity9am Pressure3pm Cloud3pm Temp9am Temp3pm
1        22.9       NA            44          71    1007.100       NA    16.9    21.8
2        25.1       NA            44          44    1007.800       NA    17.2    24.3
3        25.7       NA            46          38    1008.700        2    21.0    23.2
4        28.0       NA            24          45    1012.800       NA    18.1    26.5
5        32.3       NA            41          82    1006.000        8    17.8    29.7
6        29.7       NA            56          55    1005.400       NA    20.6    28.9
7        25.0       NA            50          49    1008.200       NA    18.1    24.6
8        26.7       NA            35          48    1010.100       NA    16.3    25.5
9        31.9       NA            80          42    1003.600       NA    18.3    30.2
10       30.1       NA            28          58    1005.700       NA    20.1    28.2
11       30.4       NA            30          48    1008.700       NA    20.4    28.8
12       21.7       NA            31          89    1004.200        8    15.9    17.0
13       18.6       NA            61          76     993.000        8    17.4    15.8
14       21.0       NA            44          65    1001.800        7    15.8    19.8
15       24.6       NA            NA          57    1008.700       NA    15.9    23.5
16       27.7       NA            50          50    1010.300       NA    17.3    26.2
17       20.9       NA            22          69    1010.400        1    17.2    18.1
18       22.9       NA            63          80    1002.200        1    18.0    21.5
19       22.5       NA            43          47    1009.700        2    15.5    21.0
20       25.6       NA            26          45    1017.100       NA    15.8    23.2
```

```
> colnames(weather_data9)
[1] "MaxTemp"       "Sunshine"      "WindGustSpeed" "Humidity9am"   "Pressure3pm"   "Cloud3pm"      "Temp9am"       "Temp3pm"
> #merge numeric anf factor columns
> weather_data10=cbind(weather_data9,weather_data7)
> summary(weather_data10)
    MaxTemp         Sunshine     WindGustSpeed     Humidity9am      Pressure3pm        Cloud3pm        Temp9am         Temp3pm
 Min.   :-4.80   Min.   : 0.00   Min.   :  6.00   Min.   :  0.00   Min.   : 977.1   Min.   :0.00    Min.   :-7.20   Min.   :-5.40
 1st Qu.:17.90   1st Qu.: 4.80   1st Qu.: 31.00   1st Qu.: 57.00   1st Qu.:1010.4   1st Qu.:2.00    1st Qu.:12.30   1st Qu.:16.60
 Median :22.60   Median : 8.40   Median : 39.00   Median : 70.00   Median :1015.2   Median :5.00    Median :16.70   Median :21.10
 Mean   :23.22   Mean   : 7.61   Mean   : 40.03   Mean   : 68.88   Mean   :1015.3   Mean   :4.51    Mean   :16.99   Mean   :21.68
 3rd Qu.:28.20   3rd Qu.:10.60   3rd Qu.: 48.00   3rd Qu.: 83.00   3rd Qu.:1020.0   3rd Qu.:7.00    3rd Qu.:21.60   3rd Qu.:26.40
 Max.   :48.10   Max.   :14.50   Max.   :135.00   Max.   :100.00   Max.   :1039.6   Max.   :9.00    Max.   :40.20   Max.   :46.70
 NA's   :1261    NA's   :69835   NA's   :10263    NA's   :2654     NA's   :15028    NA's   :59358   NA's   :1767    NA's   :3609
  WindGustDir      WindDir9am      WindDir3pm    RainTomorrow
 W      : 9915   N      :11758   SE     :10838   No  :110316
 SE     : 9418   SE     : 9287   W      :10110   Yes : 31877
 N      : 9313   E      : 9176   S      : 9926   NA's:  3267
 SSE    : 9216   SSE    : 9112   WSW    : 9518
 E      : 9181   NW     : 8749   SSE    : 9399
 (Other):88091   (Other):86812   (Other):91441
 NA's   :10326   NA's   :10566   NA's   : 4228
```

4. Data normalisation/Cleaning

For all NA values we will replace with mean and mode.

```
> #Replace NA values with mean,mode
> library(dplyr)
> weather_data10=weather_data10 %>% mutate_if(is.numeric, funs(replace(.,is.na(.), mean(., na.rm = TRUE)))) %>%
+   mutate_if(is.factor, funs(replace(.,is.na(.), getmode(na.omit(.)))))
> summary(weather_data10)
    MaxTemp         Sunshine     WindGustSpeed     Humidity9am      Pressure3pm        Cloud3pm        Temp9am         Temp3pm
 Min.   :-4.80   Min.   : 0.000   Min.   :  6.00   Min.   :  0.00   Min.   : 977.1   Min.   :0.00    Min.   :-7.20   Min.   :-5.40
 1st Qu.:18.00   1st Qu.: 7.611   1st Qu.: 31.00   1st Qu.: 57.00   1st Qu.:1011.1   1st Qu.:4.00    1st Qu.:12.30   1st Qu.:16.70
 Median :22.70   Median : 7.611   Median : 39.00   Median : 69.00   Median :1015.3   Median :4.51    Median :16.80   Median :21.40
 Mean   :23.22   Mean   : 7.611   Mean   : 40.04   Mean   : 68.88   Mean   :1015.3   Mean   :4.51    Mean   :16.99   Mean   :21.68
 3rd Qu.:28.20   3rd Qu.: 8.700   3rd Qu.: 46.00   3rd Qu.: 83.00   3rd Qu.:1019.4   3rd Qu.:6.00    3rd Qu.:21.50   3rd Qu.:26.20
 Max.   :48.10   Max.   :14.500   Max.   :135.00   Max.   :100.00   Max.   :1039.6   Max.   :9.00    Max.   :40.20   Max.   :46.70

  WindGustDir      WindDir9am      WindDir3pm    RainTomorrow
 W      :20241   N      :22324   SE     :15066   No :113583
 SE     : 9418   SE     : 9287   W      :10110   Yes: 31877
 N      : 9313   E      : 9176   S      : 9926
 SSE    : 9216   SSE    : 9112   WSW    : 9518
 E      : 9181   NW     : 8749   SSE    : 9399
 S      : 9168   S      : 8659   SW     : 9354
 (Other):78923   (Other):78153   (Other):82087
>
```

> cat("No. of missing value",sum(is.na(weather_data10)))

No. of missing value 0

So no missing values in the dataset. So now available for modelling.

5. Data modelling

data was split into test and train data in the ratio 75:25.

```
>
> # 4.Data Modeling
> weather_data10$WindGustDir=as.numeric(weather_data10$WindGustDir)
> weather_data10$WindDir9am=as.numeric(weather_data10$WindDir9am)
> weather_data10$WindDir3pm=as.numeric(weather_data10$WindDir3pm)
> #Convert Raintomrrow data to numeric
> library(plyr)
> weather_data10$RainTomorrow <- revalue(weather_data10$RainTomorrow, c("Yes"=1))
> weather_data10$RainTomorrow <- revalue(weather_data10$RainTomorrow, c("No"=0))
> #Data us split to test and train data in the ratio 75:25
> #weather_data10
> library(caTools)
> set.seed(123)
> split = sample.split(weather_data10$RainTomorrow, SplitRatio = 0.75)
> split
   [1]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
  [23]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
  [45]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
  [67] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
  [89]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
 [111]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
 [133]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
 [155]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
 [177]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [199]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [221] FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
 [243] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
 [265]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE
 [287]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [309]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
 [331]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
 [353]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

For training set given all the true values of split and for test set all values of false. And scaled for other attributes

```
>
> training_set = subset(weather_data10, split == TRUE)
> test_set = subset(weather_data10, split == FALSE)
> #training_set$RainTomorrow
> # Feature Scaling
> training_set[-12] = scale(training_set[-12])
> test_set[-12] = scale(test_set[-12])
>
```

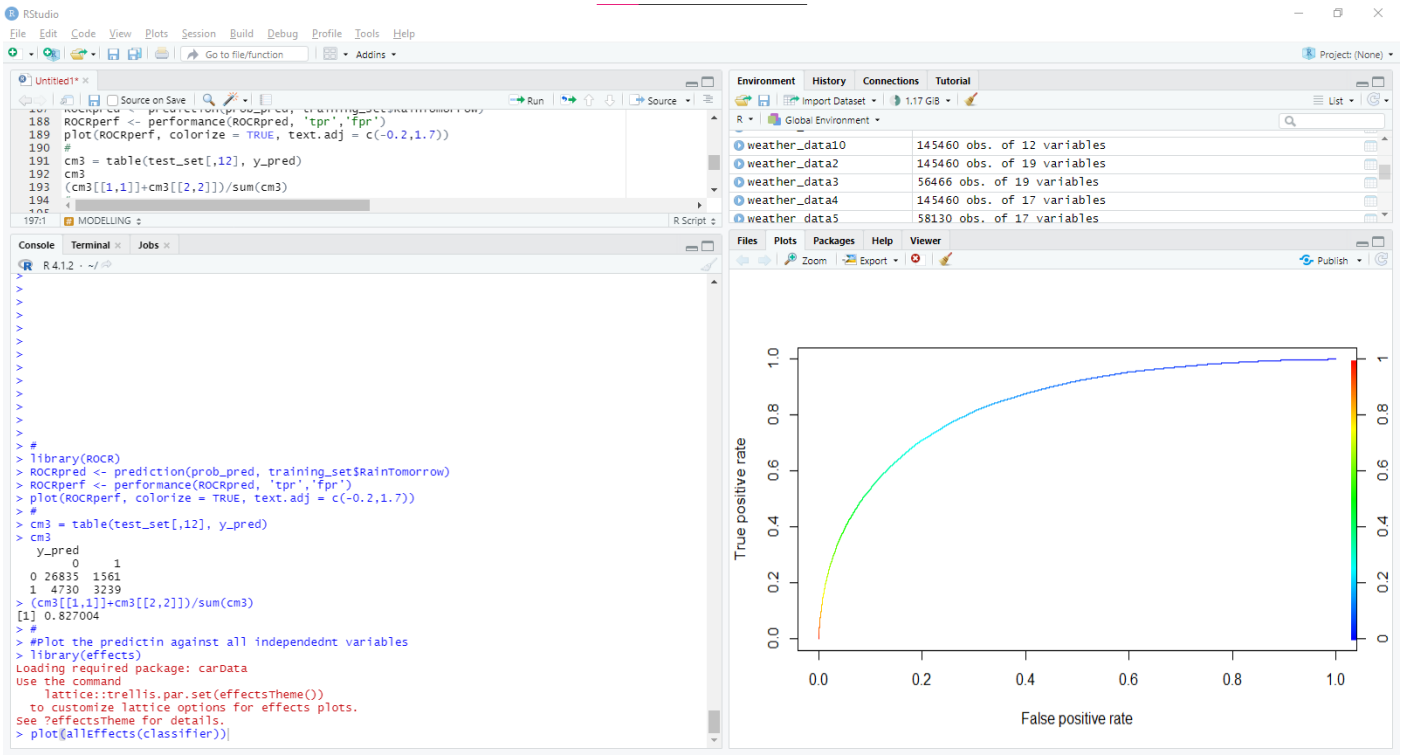
6. Applying algorithms for prediction on dataset

Logistic regression

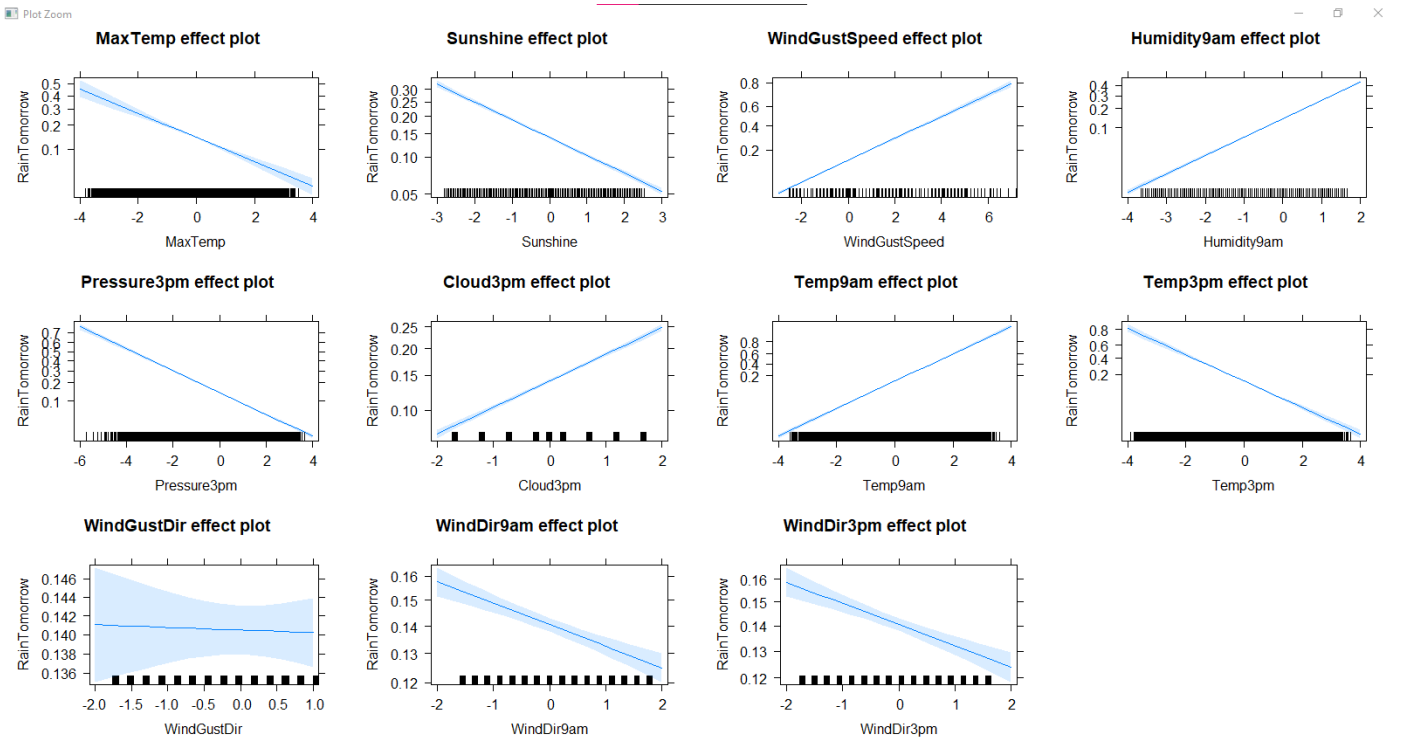
```
> # Fitting Logistic Regression to the Training set
> classifier = glm(formula = RainTomorrow ~ .,
+                  family = binomial,
+                  data = training_set)
> summary(classifier)

Call:
glm(formula = RainTomorrow ~ ., family = binomial, data = training_set)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6960  -0.6070  -0.3547  -0.1358   3.2772

Coefficients:
               Estimate Std. Error  z value Pr(>|z|)
(Intercept)   -1.811110   0.010849 -166.945  < 2e-16 ***
MaxTemp       -0.410266   0.034912  -11.751  < 2e-16 ***
Sunshine      -0.356540   0.010266  -34.732  < 2e-16 ***
WindGustSpeed  0.449281   0.009438   47.605  < 2e-16 ***
Humidity9am    0.781716   0.011938   65.483  < 2e-16 ***
Pressure3pm   -0.484713   0.010196  -47.537  < 2e-16 ***
Cloud3pm       0.348153   0.011372   30.615  < 2e-16 ***
Temp9am        1.112824   0.021211   52.465  < 2e-16 ***
Temp3pm       -0.814044   0.030969  -26.285  < 2e-16 ***
WindGustDir   -0.002227   0.011257   -0.198    0.843
WindDir9am    -0.066891   0.009789   -6.833 8.29e-12 ***
WindDir3pm    -0.071763   0.011250   -6.379 1.78e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 114730  on 109094  degrees of freedom
Residual deviance:  84024  on 109083  degrees of freedom
AIC: 84048

Number of Fisher Scoring iterations: 5

> #Predict using test set
> prob_pred = predict(classifier, type = 'response')
> prob_prd_glm=predict(classifier, type = 'response', newdata = test_set[-12])
> y_pred = ifelse(prob_prd_glm > 0.5, 1, 0)
>
```

After constructing confusion matrix, we get to know about the accuracy of our model.

And accuracy for Logistic regression is 82.7004%.



If plot a graph against all independent variables it looks as shown below.



KNN

We considered k value as k=7 here in this algorithm.

```
> #KNN---------------------------------------------------------------------------
> set.seed(200)
> library(class)
> y_predknn = knn(train = training_set[, -12],
+                 test = test_set[, -12],
+                 cl = training_set[, 12],
+                 k = 7,
+                 prob = TRUE)
> # Making the Confusion Matrix
> cm_knn = table(test_set[, 12], y_predknn)
> cm_knn
   y_predknn
        1     2
  0 26469  1927
  1  4493  3476
> (cm_knn[[1,1]]+cm_knn[[2,2]])/sum(cm_knn)
[1] 0.8234566
```

The KNN algorithm results in the accuracy of about 82.34566%.

SVM

#SVM----------

library(e1071)

svmfit = svm(formula = RainTomorrow ~ .,

data = training_set,

type = 'C-classification',

kernel = 'linear')

Predicting the Test set results

y_pred_svm = predict(svmfit, newdata = test_set[-12])

Making the Confusion Matrix

cm_svm = table(test_set[, 12], y_pred_svm)

cm_svm

(cm_svm[[1,1]]+cm_svm[[2,2]])/sum(cm_svm)

#plot the output of svm

plot(test_set[-12],svmfit)

here is the code to run SVM but it takes heavy time in the order of O(n^3).

So the #svm output for confusion matrix will be

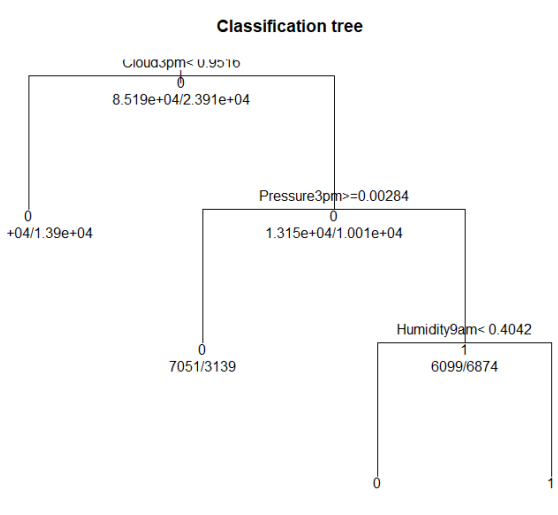
#26344 5043

#1235 2926

With accuracy of #Svm=82%

DECISION TREE

```
> library(rpart)
> model2 <- rpart(RainTomorrow ~.,training_set,method='class')
> plot(model2,uniform=TRUE,main="Classification tree")
> text(model2,use.n=TRUE,all=TRUE,cex=1.0)
> treePred<- predict(model2,test_set[-12],type='class')
> cm_tree=table(treePred,test_set[, 12])
> cm_tree

treePred     0     1
       0 27675  6589
       1   721  1380
> (cm_tree[[1,1]]+cm_tree[[2,2]])/sum(cm_tree)
[1] 0.7989825
```



The DECISION TREE results in the accuracy of about 79.89825%.

CONCLUSION

In this Activity we analysed dataset by knowing attributes in data, mean, median and class distribution of the dataset. We identified RainTomorrow as the target variable. We also made data visualization. And implemented algorithms for prediction**.**