Ensemble Techniques

Bishal Neupane, Saugat Gyawali

10/08/2022

Source:

https://www.kaggle.com/code/abhpasha/logistic-regression-predicting-rain-in-australia

Importing data and taking only first 15k because the data is two large more than 100k

```
df <- read.csv("weatherAUS.csv", header = TRUE)
```

```
head(df)
```

```
##        Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir
## 1 12/1/2008   Albury    13.4    22.9      0.6          NA       NA           W
## 2 12/2/2008   Albury     7.4    25.1      0.0          NA       NA         WNW
## 3 12/3/2008   Albury    12.9    25.7      0.0          NA       NA         WSW
## 4 12/4/2008   Albury     9.2    28.0      0.0          NA       NA          NE
## 5 12/5/2008   Albury    17.5    32.3      1.0          NA       NA           W
## 6 12/6/2008   Albury    14.6    29.7      0.2          NA       NA         WNW
##   WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am
## 1            44          W        WNW           20           24          71
## 2            44        NNW        WSW            4           22          44
## 3            46          W        WSW           19           26          38
## 4            24         SE          E           11            9          45
## 5            41        ENE         NW            7           20          82
## 6            56          W          W           19           24          55
##   Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm
## 1          22      1007.7      1007.1        8       NA    16.9    21.8
## 2          25      1010.6      1007.8       NA       NA    17.2    24.3
## 3          30      1007.6      1008.7       NA        2    21.0    23.2
## 4          16      1017.6      1012.8       NA       NA    18.1    26.5
## 5          33      1010.8      1006.0        7        8    17.8    29.7
## 6          23      1009.2      1005.4       NA       NA    20.6    28.9
##   RainToday RainTomorrow
## 1        No           No
## 2        No           No
## 3        No           No
## 4        No           No
## 5        No           No
## 6        No           No
```

#There are alot of column so removing columns with non numeric values.

```
df$Date<- NULL
df$WindGustDir<-NULL
df$WindGustDir <-NULL
df$WindDir3pm <- NULL
df$WindDir3pm <-NULL
df$Location <-NULL
df$Sunshine <-NULL
df$RainToday <- NULL
df$WindDir9am <-NULL
df$Evaporation <-NULL
```

Structure of Data Frame

```
str(df)
```

```
## 'data.frame':    145460 obs. of  15 variables:
##  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
##  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
##  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
##  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
##  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
##  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
##  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
##  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
##  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
##  $ RainTomorrow : chr  "No" "No" "No" "No" ...
```

Data Exploration

Names of Column

```
names(df)
```

```
##  [1] "MinTemp"       "MaxTemp"       "Rainfall"      "WindGustSpeed"
##  [5] "WindSpeed9am"  "WindSpeed3pm"  "Humidity9am"   "Humidity3pm"
##  [9] "Pressure9am"   "Pressure3pm"   "Cloud9am"      "Cloud3pm"
## [13] "Temp9am"       "Temp3pm"       "RainTomorrow"
```

Importing Package and using it to Change to factor

```
#install.packages("dplyr")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df <- mutate_if(df, is.character, as.factor)
```

Dimensions of df

```
dim(df)
```

```
## [1] 145460     15
```

```
str(df)
```

```
## 'data.frame':    145460 obs. of  15 variables:
##  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
##  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
##  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
##  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
##  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
##  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
##  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
##  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
##  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
##  $ RainTomorrow : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 2 1 ...
```

Statistics Summary of Each column

```
summary(df)
```

```
##     MinTemp         MaxTemp         Rainfall       WindGustSpeed
##  Min.   :-8.50   Min.   :-4.80   Min.   :  0.000   Min.   :  6.00
##  1st Qu.: 7.60   1st Qu.:17.90   1st Qu.:  0.000   1st Qu.: 31.00
##  Median :12.00   Median :22.60   Median :  0.000   Median : 39.00
##  Mean   :12.19   Mean   :23.22   Mean   :  2.361   Mean   : 40.03
##  3rd Qu.:16.90   3rd Qu.:28.20   3rd Qu.:  0.800   3rd Qu.: 48.00
##  Max.   :33.90   Max.   :48.10   Max.   :371.000   Max.   :135.00
##  NA’s   :1485    NA’s   :1261    NA’s   :3261      NA’s   :10263
##   WindSpeed9am     WindSpeed3pm    Humidity9am      Humidity3pm
##  Min.   :  0.00   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:  7.00   1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 37.00
##  Median : 13.00   Median :19.00   Median : 70.00   Median : 52.00
##  Mean   : 14.04   Mean   :18.66   Mean   : 68.88   Mean   : 51.54
##  3rd Qu.: 19.00   3rd Qu.:24.00   3rd Qu.: 83.00   3rd Qu.: 66.00
##  Max.   :130.00   Max.   :87.00   Max.   :100.00   Max.   :100.00
##  NA’s   :1767     NA’s   :3062    NA’s   :2654     NA’s   :4507
##   Pressure9am      Pressure3pm        Cloud9am        Cloud3pm
##  Min.   : 980.5   Min.   : 977.1   Min.   :0.00    Min.   :0.00
##  1st Qu.:1012.9   1st Qu.:1010.4   1st Qu.:1.00    1st Qu.:2.00
##  Median :1017.6   Median :1015.2   Median :5.00    Median :5.00
##  Mean   :1017.6   Mean   :1015.3   Mean   :4.45    Mean   :4.51
##  3rd Qu.:1022.4   3rd Qu.:1020.0   3rd Qu.:7.00    3rd Qu.:7.00
##  Max.   :1041.0   Max.   :1039.6   Max.   :9.00    Max.   :9.00
##  NA’s   :15065    NA’s   :15028    NA’s   :55888   NA’s   :59358
##     Temp9am         Temp3pm       RainTomorrow
##  Min.   :-7.20   Min.   :-5.40   No  :110316
##  1st Qu.:12.30   1st Qu.:16.60   Yes : 31877
##  Median :16.70   Median :21.10   NA’s:  3267
##  Mean   :16.99   Mean   :21.68
##  3rd Qu.:21.60   3rd Qu.:26.40
##  Max.   :40.20   Max.   :46.70
##  NA’s   :1767    NA’s   :3609
```

Exploring Missing values

```
sum(is.na(df))
```

```
## [1] 182242
```

Removing the row with target value NA

```
df <- subset(df,RainTomorrow  != "NA")
```

Dimension after removing rows with NA as Rain Tomorrow

```
dim(df)
```

```
## [1] 142193     15
```

```
str(df)
```

```
## 'data.frame':    142193 obs. of  15 variables:
##  $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
##  $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
##  $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
##  $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
##  $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
##  $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
##  $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
##  $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
##  $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
##  $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
##  $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
##  $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
##  $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
##  $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
##  $ RainTomorrow : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 2 1 ...
```

Replacing NA's with mean of a column

```
#install.packages('tidyr')
for(i in 1:ncol(df)){
  df[is.na(df[,i]), i] <- mean(df[,i], na.rm = TRUE)
}
```

```
## Warning in mean.default(df[, i], na.rm = TRUE): argument is not numeric or
## logical: returning NA
```

Summary after replacing NA's with mean

```
summary(df)
```

```
##     MinTemp         MaxTemp         Rainfall      WindGustSpeed
##  Min.   :-8.50   Min.   :-4.80   Min.   :  0.00   Min.   :  6.00
##  1st Qu.: 7.60   1st Qu.:17.90   1st Qu.:  0.00   1st Qu.: 31.00
##  Median :12.00   Median :22.70   Median :  0.00   Median : 39.00
##  Mean   :12.19   Mean   :23.23   Mean   :  2.35   Mean   : 39.98
##  3rd Qu.:16.80   3rd Qu.:28.20   3rd Qu.:  0.80   3rd Qu.: 46.00
##  Max.   :33.90   Max.   :48.10   Max.   :371.00   Max.   :135.00
##   WindSpeed9am  WindSpeed3pm    Humidity9am      Humidity3pm
##  Min.   :  0   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:  7   1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 37.00
##  Median : 13   Median :18.64   Median : 70.00   Median : 51.48
##  Mean   : 14   Mean   :18.64   Mean   : 68.84   Mean   : 51.48
##  3rd Qu.: 19   3rd Qu.:24.00   3rd Qu.: 83.00   3rd Qu.: 65.00
##  Max.   :130   Max.   :87.00   Max.   :100.00   Max.   :100.00
##   Pressure9am      Pressure3pm        Cloud9am        Cloud3pm
##  Min.   : 980.5   Min.   : 977.1   Min.   :0.000   Min.   :0.000
##  1st Qu.:1013.5   1st Qu.:1011.0   1st Qu.:3.000   1st Qu.:4.000
##  Median :1017.7   Median :1015.3   Median :4.437   Median :4.503
##  Mean   :1017.7   Mean   :1015.3   Mean   :4.437   Mean   :4.503
##  3rd Qu.:1021.8   3rd Qu.:1019.4   3rd Qu.:6.000   3rd Qu.:6.000
##  Max.   :1041.0   Max.   :1039.6   Max.   :9.000   Max.   :9.000
##     Temp9am         Temp3pm      RainTomorrow
##  Min.   :-7.20   Min.   :-5.40   No :110316
##  1st Qu.:12.30   1st Qu.:16.70   Yes: 31877
##  Median :16.80   Median :21.30
##  Mean   :16.99   Mean   :21.69
##  3rd Qu.:21.50   3rd Qu.:26.30
##  Max.   :40.20   Max.   :46.70
```

Data Visualization

```r
par(mfrow=c(1,6))
plot(df$RainTomorrow, df$MinTemp, data=df, main="MinTemp",
varwidth=TRUE)
plot(df$RainTomorrow, df$MaxTemp, data=df, main="MaxTemp", varwidth=TRUE)
plot(df$RainTomorrow, df$Rainfall, data=df, main="Rainfall", varwidth=TRUE)
plot(df$RainTomorrow, df$Evaporation, data=df, main="Evaporation", varwidth=TRUE)
```

```
## Warning in plot.window(...): "data" is not a graphical parameter

## Warning in plot.window(...): "varwidth" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "varwidth" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is not a
## graphical parameter

## Warning in box(...): "data" is not a graphical parameter

## Warning in box(...): "varwidth" is not a graphical parameter

## Warning in title(...): "data" is not a graphical parameter

## Warning in title(...): "varwidth" is not a graphical parameter
```

```
plot(df$RainTomorrow, df$Sunshine, data=df, main="Sunshine", varwidth=TRUE)
```

```
## Warning in plot.window(...): "data" is not a graphical parameter
```

```
## Warning in plot.window(...): "varwidth" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "varwidth" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is not a
## graphical parameter
```

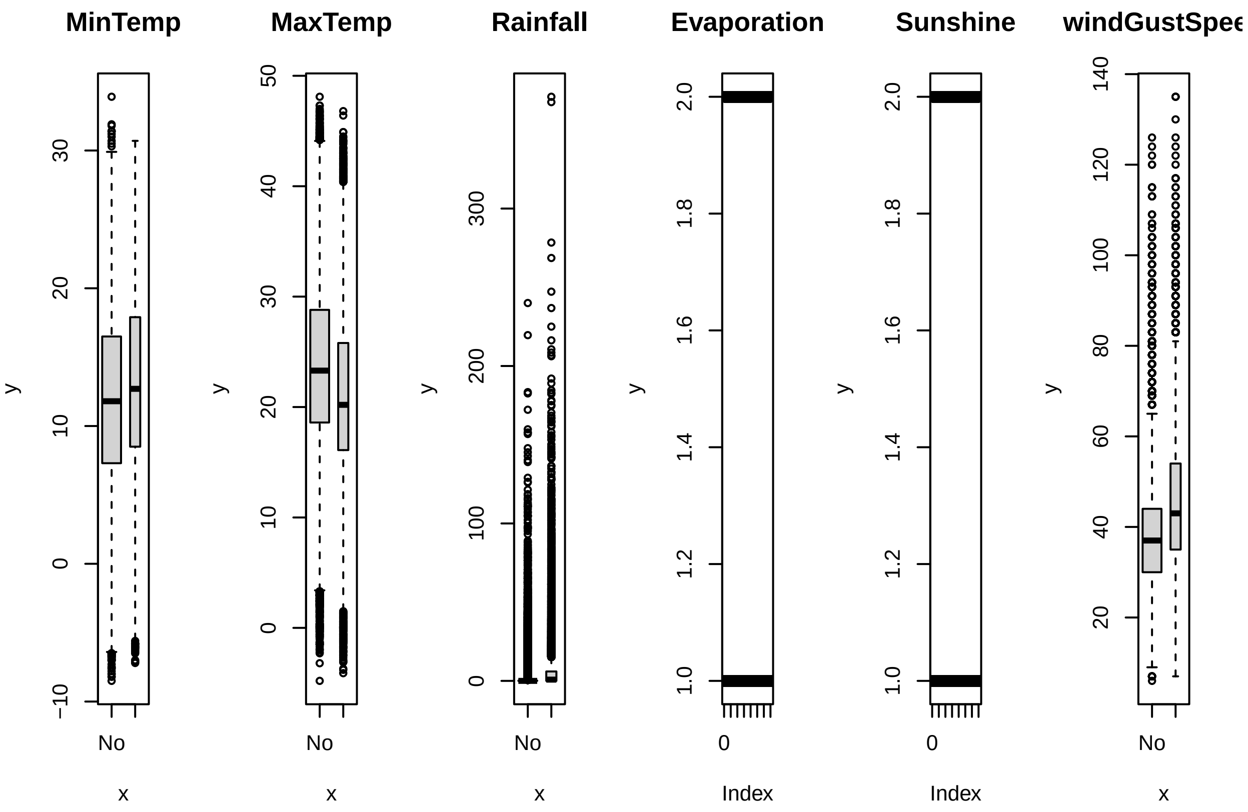
```
## Warning in box(...): "data" is not a graphical parameter
```

```
## Warning in box(...): "varwidth" is not a graphical parameter
```

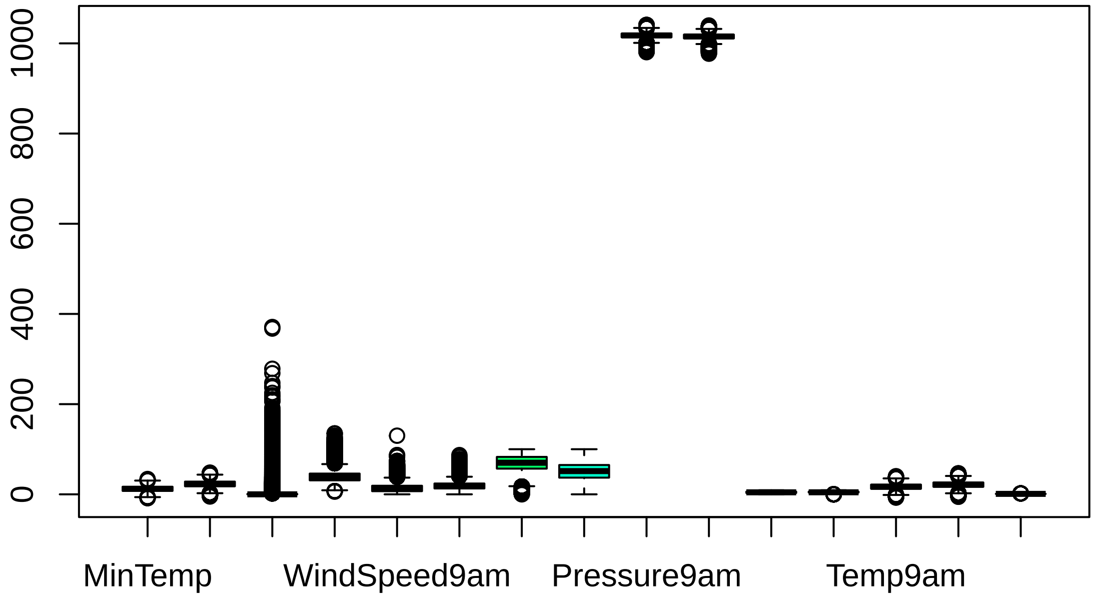
```
## Warning in title(...): "data" is not a graphical parameter
```

```
## Warning in title(...): "varwidth" is not a graphical parameter
```

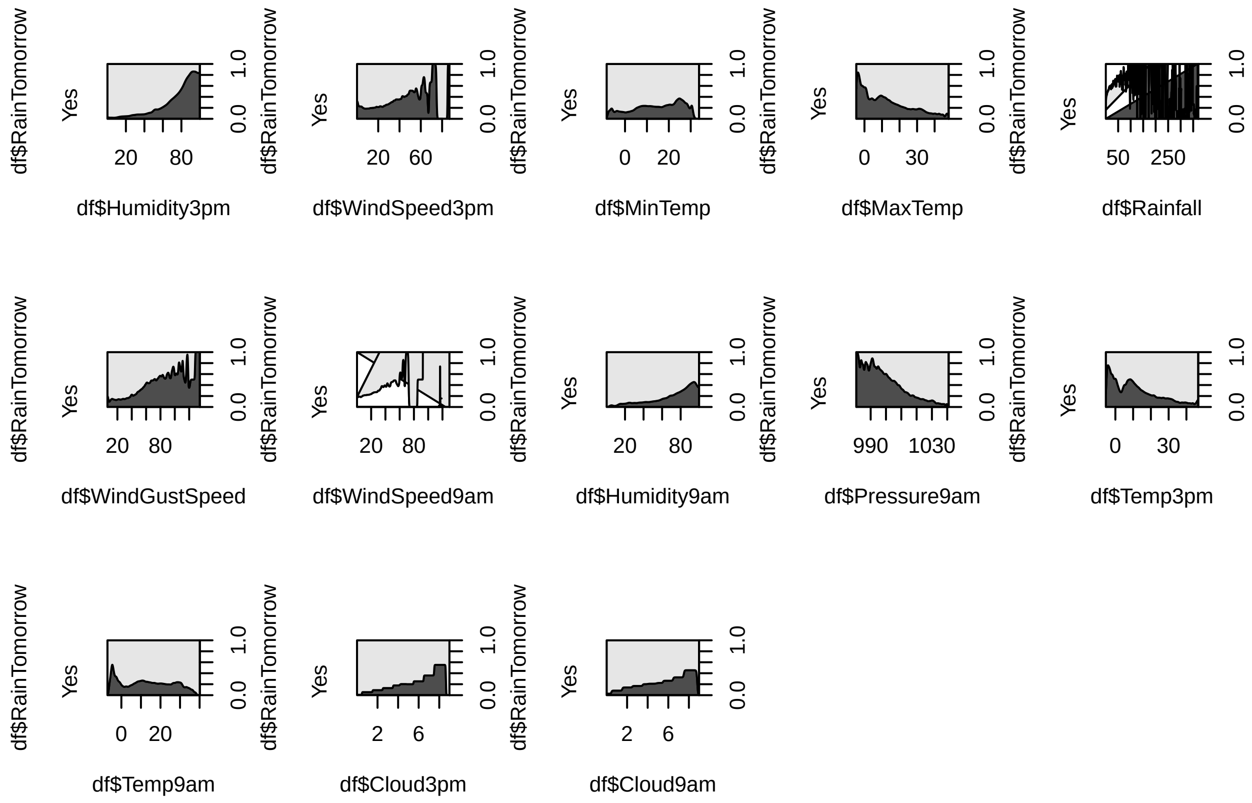
```
plot(df$RainTomorrow, df$WindGustSpeed, data=df, main="windGustSpeed",
varwidth=TRUE)
```



```
boxplot(df, col = rainbow(ncol(df)))
```



```
par(mfrow=c(3,5))
cdplot(df$RainTomorrow~df$Humidity3pm)
cdplot(df$RainTomorrow~df$WindSpeed3pm)
cdplot(df$RainTomorrow~df$MinTemp)
cdplot(df$RainTomorrow~df$MaxTemp)
cdplot(df$RainTomorrow~df$Rainfall)
cdplot(df$RainTomorrow~df$WindGustSpeed)
cdplot(df$RainTomorrow~df$WindSpeed9am)
cdplot(df$RainTomorrow~df$Humidity9am)
cdplot(df$RainTomorrow~df$Pressure9am)
cdplot(df$RainTomorrow~df$Temp3pm)
cdplot(df$RainTomorrow~df$Temp9am)
cdplot(df$RainTomorrow~df$Cloud3pm)
cdplot(df$RainTomorrow~df$Cloud9am)
```



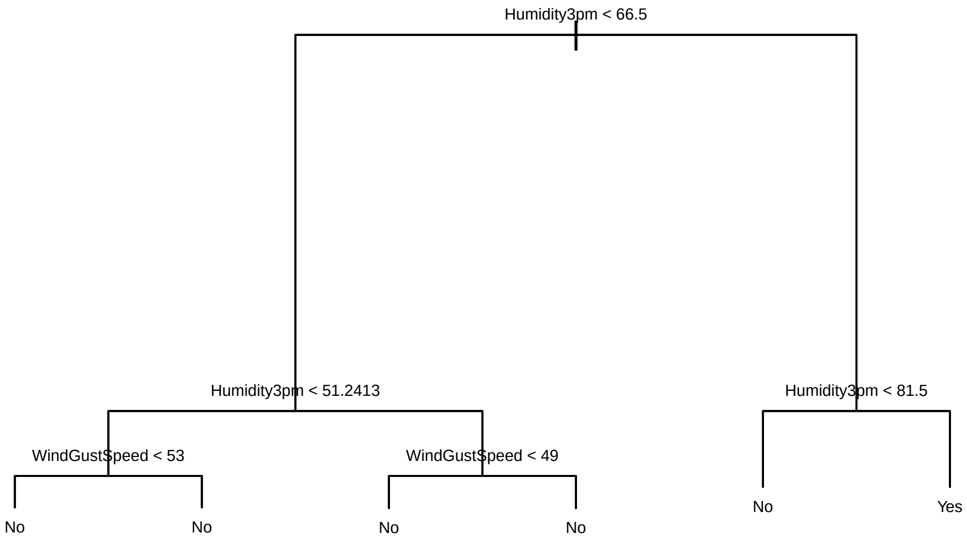
##Spliting into train and test set

```
set.seed(1234)
i <- sample(1:nrow(df), 0.80*nrow(df), replace=FALSE)
train <-df[i,]
test <- df[-i,]
testLabel <- test$RainTomorrow

trainLabel <- train$RainTomorrow
```

Decision Tree

```
library(tree)
tree_weather <- tree(RainTomorrow~.,data = train)
plot(tree_weather)
text(tree_weather, cex= 0.5, pretty=0)
```



### Prediction, Confusion Matrix and Statistics

```
prediction <- predict(tree_weather, newdata = test, type = "class")
table(prediction, testLabel)
```

```
##           testLabel
## prediction    No   Yes
##        No  21515  4592
##        Yes   490  1842
```

```
levels(prediction) <- list("1" = "No", "2" = "Yes")
levels(testLabel) <- list("1" = "No", "2" = "Yes")
library(caret)
```

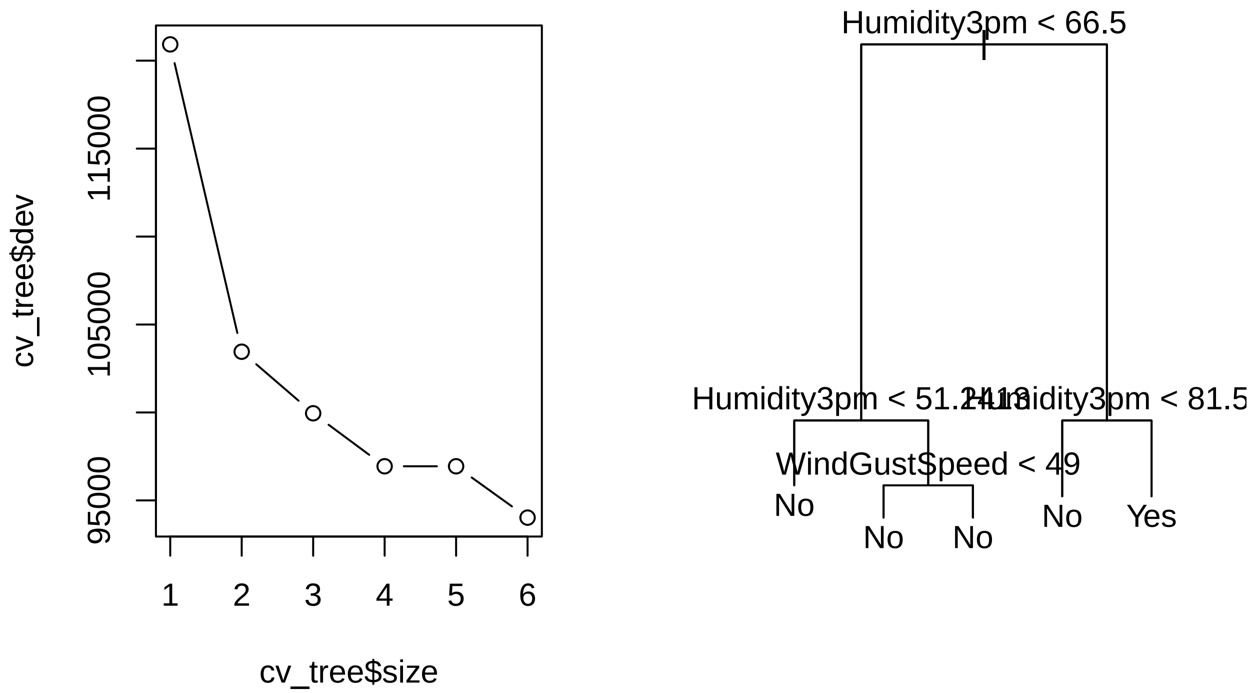
```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
confusionMatrix(as.factor(prediction),as.factor(testLabel))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 21515  4592
##          2   490  1842
##
##                Accuracy : 0.8213
##                  95% CI : (0.8168, 0.8257)
##     No Information Rate : 0.7738
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3409
##
##  Mcnemar’s Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9777
##             Specificity : 0.2863
##          Pos Pred Value : 0.8241
##          Neg Pred Value : 0.7899
##              Prevalence : 0.7738
##          Detection Rate : 0.7565
##    Detection Prevalence : 0.9180
##       Balanced Accuracy : 0.6320
##
##        ’Positive’ Class : 1
##
```

```
par(mfrow=c(1,2))
cv_tree <- cv.tree(tree_weather)
plot(cv_tree$size, cv_tree$dev, type="b")
tree_pruned <- prune.tree(tree_weather, best=5)
plot(tree_pruned)
text(tree_pruned, pretty=0)
```



```
prediction1 <- predict(tree_weather, newdata = test, type = "class")
table(prediction1, testLabel)
```

```
##            testLabel
## prediction1     1     2
##         No  21515  4592
##         Yes   490  1842
```

```
levels(prediction) <- list("1" = "No", "2" = "Yes")
levels(testLabel) <- list("1" = "No", "2" = "Yes")
library(caret)
confusionMatrix(as.factor(prediction),as.factor(testLabel))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 21515  4592
##          2   490  1842
##
##                Accuracy : 0.8213
##                  95% CI : (0.8168, 0.8257)
##     No Information Rate : 0.7738
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3409
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9777
##             Specificity : 0.2863
##          Pos Pred Value : 0.8241
##          Neg Pred Value : 0.7899
##              Prevalence : 0.7738
##          Detection Rate : 0.7565
##    Detection Prevalence : 0.9180
##       Balanced Accuracy : 0.6320
##
##        'Positive' Class : 1
##
```

Random Forest

```r
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
set.seed(1234)
randomf <- randomForest(train$RainTomorrow~., data = train, importance = TRUE)
```

Recution and confusion Matrix

```r
pred <- predict(randomf, newdata = test, type = "response")
levels(pred) <- list("1" = "No", "2" = "Yes")
levels(testLabel) <- list("1" = "No", "2" = "Yes")
library(caret)
confusionMatrix(as.factor(pred),as.factor(testLabel))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 20980  3200
##          2  1025  3234
##
##                Accuracy : 0.8514
##                  95% CI : (0.8472, 0.8556)
##     No Information Rate : 0.7738
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.518
##
##  Mcnemar’s Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9534
##             Specificity : 0.5026
##          Pos Pred Value : 0.8677
##          Neg Pred Value : 0.7593
##              Prevalence : 0.7738
##          Detection Rate : 0.7377
##    Detection Prevalence : 0.8502
##       Balanced Accuracy : 0.7280
##
##        ’Positive’ Class : 1
##
```

Boosting

```
#install.packages('adabag')
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: foreach
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
ada1 <- boosting(RainTomorrow~., data = train, boos = TRUE, mfinal =15, coeflearn = "Breiman")
summary(ada1)
```

```
##            Length Class   Mode
## formula         3 formula call
## trees          15 -none-  list
## weights        15 -none-  numeric
## votes      227508 -none-   numeric
## prob       227508 -none-   numeric
## class      113754 -none-   character
## importance     14 -none-   numeric
## terms           3 terms    call
## call            6 -none-   call
```

Result and Confusion Matrix

```
pred <- predict(ada1, newdata = test, type = "response")

accuracy <- mean(pred$class==test$RainTomorrow)
print(paste("accuracy is ", accuracy))
```

```
## [1] "accuracy is  0.838285453075003"
```

XGBOOST

```
#install.packages('xgboost')
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
#levels(trainLabel) <- list("0" = "No", "1" = "Yes")

model <- xgboost(data=data.matrix(train), label=trainLabel,nrounds=100)
```

```
## [1]  train-rmse:0.584553
## [2]  train-rmse:0.409193
## [3]  train-rmse:0.286439
## [4]  train-rmse:0.200510
## [5]  train-rmse:0.140359
## [6]  train-rmse:0.098253
## [7]  train-rmse:0.068778
## [8]  train-rmse:0.048145
## [9]  train-rmse:0.033702
## [10] train-rmse:0.023592
## [11] train-rmse:0.016514
## [12] train-rmse:0.011560
## [13] train-rmse:0.008092
## [14] train-rmse:0.005665
## [15] train-rmse:0.003965
## [16] train-rmse:0.002776
## [17] train-rmse:0.001943
## [18] train-rmse:0.001360
## [19] train-rmse:0.000952
## [20] train-rmse:0.000666
## [21] train-rmse:0.000467
## [22] train-rmse:0.000327
## [23] train-rmse:0.000229
## [24] train-rmse:0.000160
## [25] train-rmse:0.000112
## [26] train-rmse:0.000078
## [27] train-rmse:0.000055
## [28] train-rmse:0.000038
## [29] train-rmse:0.000027
## [30] train-rmse:0.000019
## [31] train-rmse:0.000013
## [32] train-rmse:0.000009
## [33] train-rmse:0.000006
## [34] train-rmse:0.000005
## [35] train-rmse:0.000004
## [36] train-rmse:0.000003
## [37] train-rmse:0.000003
## [38] train-rmse:0.000002
## [39] train-rmse:0.000002
## [40] train-rmse:0.000002
## [41] train-rmse:0.000002
## [42] train-rmse:0.000002
## [43] train-rmse:0.000002
## [44] train-rmse:0.000002
## [45] train-rmse:0.000002
## [46] train-rmse:0.000002
## [47] train-rmse:0.000002
## [48] train-rmse:0.000002
## [49] train-rmse:0.000002
## [50] train-rmse:0.000002
## [51] train-rmse:0.000002
## [52] train-rmse:0.000002
## [53] train-rmse:0.000002
## [54] train-rmse:0.000002
## [55] train-rmse:0.000002
## [56] train-rmse:0.000002
## [57] train-rmse:0.000002
## [58] train-rmse:0.000002
## [59] train-rmse:0.000002
## [60] train-rmse:0.000002
## [61] train-rmse:0.000002
## [62] train-rmse:0.000002
## [63] train-rmse:0.000002
## [64] train-rmse:0.000002
## [65] train-rmse:0.000002
## [66] train-rmse:0.000002
## [67] train-rmse:0.000002
## [68] train-rmse:0.000002
## [69] train-rmse:0.000002
## [70] train-rmse:0.000002
## [71] train-rmse:0.000002
## [72] train-rmse:0.000002
## [73] train-rmse:0.000002
## [74] train-rmse:0.000002
## [75] train-rmse:0.000002
## [76] train-rmse:0.000002
## [77] train-rmse:0.000002
## [78] train-rmse:0.000002
## [79] train-rmse:0.000002
## [80] train-rmse:0.000002
## [81] train-rmse:0.000002
## [82] train-rmse:0.000002
## [83] train-rmse:0.000002
## [84] train-rmse:0.000002
## [85] train-rmse:0.000002
## [86] train-rmse:0.000002
## [87] train-rmse:0.000002
## [88] train-rmse:0.000002
## [89] train-rmse:0.000002
## [90] train-rmse:0.000002
## [91] train-rmse:0.000002
## [92] train-rmse:0.000002
## [93] train-rmse:0.000002
## [94] train-rmse:0.000002
## [95] train-rmse:0.000002
## [96] train-rmse:0.000002
## [97] train-rmse:0.000002
## [98] train-rmse:0.000002
## [99] train-rmse:0.000002
## [100]    train-rmse:0.000002
```

```
summary(model)
```

```
##                 Length Class              Mode
## handle              1  xgb.Booster.handle externalptr
## raw             75295  -none-             raw
## niter               1  -none-             numeric
## evaluation_log      2  data.table         list
## call               13  -none-             call
## params              1  -none-             list
## callbacks           2  -none-             list
## feature_names      15  -none-             character
## nfeatures           1  -none-             numeric
```

```
levels(test$RainTomorrow) <- list("1" = "No", "2" = "Yes")
probs <- predict(model, data.matrix(test))
pred <- ifelse(probs>0.5, 1, 0)
levels(pred) <- list("1" = "No", "2" = "Yes")
levels(testLabel) <- list("1" = "No", "2" = "Yes")
library(caret)
confusionMatrix(as.factor(pred),as.factor(testLabel))
```

```
## Warning in confusionMatrix.default(as.factor(pred), as.factor(testLabel)):
```

```
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1     2
##          1 22005  6434
##          2     0     0
##
##                Accuracy : 0.7738
##                  95% CI : (0.7689, 0.7786)
##     No Information Rate : 0.7738
##     P-Value [Acc > NIR] : 0.5033
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7738
##          Neg Pred Value :    NaN
##              Prevalence : 0.7738
##          Detection Rate : 0.7738
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 1
##
```

bgboost visulation

```
#install.packages(DiagrammeR)
library(DiagrammeR)
xgb.plot.tree(model=model, trees =1:3)
```

Analysis Based on Run Time and Metrics:

I used Decision Tree, Random forest, XGBoost and boosting to perform the classification. Their analysis based on the run time and metrics can be done in the following ways:

The accuracy of decision tree was about 82 percent initially and later after pruning tree, there was not significant increase in accuracy. But the accuracy of the random forest was about 85 percent. According to the accuracy, random forest outperforms decision tree which is technically true. It is because decision tree uses the concept of feature importance and make prediction by making one tree but random forest selects the features randomly and make a forest of many decision tree. It will finally combine the result of all the decision trees in the forest and generialize the result more accuratly. But the run time of the random forest is more than the run time of decision tree because decision tree creates just a tree and random forest creates many trees and combine result. Accuracy of XGboost is about 77 percent while the accuracy of random forest and decision tree is 85 and 82 percent respectively. This is true because the data set I have is multiclass classification and a lot of data was missing initally which might have increased the noise in the data. But for the Xgboot, it will have great accuracy if the data was unbalanced. The run time of XGboost in my experiment was less than the run time of random forest. It is technically true because random forest created diffeent decision tree and combine the result later but XGboost creates the result and pass the result to another which make it efficient. The accuracy of adaboosting was about 83 percent which was little higher than decision tree and less than random forest. This is technically true because decision tree uses only on tree while ada use decision lump node and two children for decision. But random forest uses many decision trees and get the result from all at last. So, accuracy of random forest is definately high. But the time of ada is comperatively less than random forset. Random tree creates all the trees and get result while ada uses decision lumps for decision which is basically node and two children.