######

list.of.packages <- c("caret", "lattice", "visNetwork","data.table")

library(caret)

install.packages("caret")

install.packages("ggplot2")

library(ggplot2)

library(lattice)

library(visNetwork)

install.packages("dummies")

library(data.table)

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

if(length(new.packages)) install.packages(new.packages)

#######

library(caret)

library(lattice)

library(visNetwork)

library(data.table)

#Read the data

data <- read.csv("weatherAUS.csv")

############################################

# data prep

colnames(data)

table(data$WindDir9am)

nrow(data)

sum(is.na(data$Evaporation))

sum(is.na(data$Sunshine))

#since"Evaporation" and "Sunshine" variable contains more NUll values i.e 140390 and 139559

#it will not make any sense to keep this attribute

data1<-data[,c(-1,-2,-5,-6,-7,-8,-10,-11,-22,-23,-24)]

data1<-data1[complete.cases(data), ]

sum(is.na(data1))

############################################

#let's calculate correlation

corr<-cor(data1)

#inspect matrix

corr

#load visualization libraries

library(ggplot2)

#install.packages("Rcpp")

#install.packages("rlang")

library(Rcpp)

library(rlang)

library(ggcorrplot)

#plot the correlation matrix visual

ggcorrplot(corr)

#add correlation coefficients & reorder matrix using hierarchical clustering

ggcorrplot(corr, hc.order = TRUE, type = "lower",

lab = TRUE)

#you can also plot the matrix with circles

ggcorrplot(corr, lab = TRUE, type = "lower", method="circle")

############################################

data.pca <- prcomp(data1, center = TRUE,scale. = TRUE)

summary(data.pca)

str(data.pca)

library(dummies)

pca.train <- data1[1:nrow(data1),]

pca.train<-pca.train[complete.cases(pca.train), ]

prin\_comp <- prcomp(pca.train, scale. = T)

names(prin\_comp)

#outputs the mean of variables

prin\_comp$center

#outputs the standard deviation of variables

prin\_comp$scale

prin\_comp$rotation

prin\_comp$rotation[1:5,1:4]

biplot(prin\_comp, scale = 0)

#compute standard deviation of each principal component

std\_dev <- prin\_comp$sdev

#compute variance

pr\_var <- std\_dev^2

#check variance of first 10 components

pr\_var[1:10]

#proportion of variance explained

prop\_varex <- pr\_var/sum(pr\_var)

prop\_varex[1:12]

#scree plot

plot(prop\_varex, xlab = "Principal Component",

ylab = "Proportion of Variance Explained",

type = "b",main = "PCA - standardised variables")

#cumulative scree plot

plot(cumsum(prop\_varex), xlab = "Principal Component",

ylab = "Cumulative Proportion of Variance Explained",

type = "b")

######################################################

ncol(data)

data<-data[,c(-1,-2,-5,-6,-7,-8,-10,-11,-22,-23)]

colnames(data)

str(data)

nrow(data)

sum(is.na(data$WindGustSpeed))

sum(is.na(data$Pressure9am))

sum(is.na(data$Pressure3pm))

sum(is.na(data$Cloud9am))

sum(is.na(data$Cloud3pm))

data$WindGustSpeed[is.na(data$WindGustSpeed)] <- mean(data$WindGustSpeed, na.rm = TRUE)

data$Pressure9am[is.na(data$Pressure9am)] <- mean(data$Pressure9am, na.rm = TRUE)

data$Pressure3pm[is.na(data$Pressure3pm)] <- mean(data$Pressure3pm, na.rm = TRUE)

data$Cloud9am[is.na(data$Cloud9am)] <- mean(data$Cloud9am, na.rm = TRUE)

data$Cloud3pm[is.na(data$Cloud3pm)] <- mean(data$Cloud3pm, na.rm = TRUE)

library(mice)

data<-data[complete.cases(data), ]

sum(is.na(data))

install.packages("mice")

library("mice")

c<-md.pattern(data)

data$RainTomorrow<-as.factor(data$RainTomorrow)

data$RainTomorrow<-factor(data$RainTomorrow,levels = c( 'No', 'Yes'),labels=c(0,1))

table(data$RainTomorrow)

##########################################################################################################################################

library(caTools)

#We use Catools mainly to split the data

set.seed(123)

#we use set.seed to give randomization to data

split = sample.split(data$RainTomorrow, SplitRatio = 0.8)

#split is performed and data is split into two sets

train = subset(data, split == TRUE)

test = subset(data, split == FALSE)

#train set contains 80% of the entire data and test contains rest 20%

###################################################Modelling##########################################

########################################################################

##Logistic

#Create plot

plot(data$MinTemp,data$MaxTemp,col=as.factor(data$RainTomorrow),xlab="MinTemp",ylab="MaxTemp")

mylogit <- glm(RainTomorrow ~ ., data = train, family = 'binomial', maxit = 100)

summary(mylogit)

colnames(train)

res=predict(mylogit,test, type="response") # prediction

predictedvalues=rep(0,nrow(test))

predictedvalues[res>0.5]=1 # probability of status being 1, if p<0.5 then status=0

a<-t(test[,14])

cm\_log=table( predictedvalues, actualvalues=a)

#accuracy

n\_log = sum(cm\_log)

diag\_log = diag(cm\_log)

accuracy\_log = sum(diag\_log) / n\_log

accuracy\_log<-accuracy\_log\*100

accuracy\_log

########################################################################

##Decision Tree

library(rpart)

ncol(data)

classifier\_dt = rpart(formula =RainTomorrow ~ .,

data = train)

rpart.plot(classifier\_dt, extra= 106)

summary(classifier\_dt)

str(data)

ncol(train)

# Predicting the Test set results

y\_pred\_dt= predict(classifier\_dt, newdata = test[-14], type = 'class')

length(a)

# Making the Confusion Matrix

cm\_dt = table(a, y\_pred\_dt)

#accuracy

n\_dt = sum(cm\_dt)

diag\_dt = diag(cm\_dt)

accuracy\_dt = sum(diag\_dt) / n\_dt

accuracy\_dt<-accuracy\_dt\*100

accuracy\_dt

########################################################################

####rf

library(randomForest)

set.seed(123)

classifier\_rf = randomForest(x = train[-14],

y = train$RainTomorrow,

ntree = 500)

# Predicting the Test set results

y\_pred\_rf = predict(classifier\_rf, newdata = test[-14])

summary(classifier\_rf)

# Making the Confusion Matrix

cm\_rf = table(a, y\_pred\_rf)

#calculating accuracy

n\_rf = sum(cm\_rf)

diag\_rf = diag(cm\_rf)

accuracy\_rf = sum(diag\_rf) / n\_rf

accuracy\_rf<-accuracy\_rf\*100

accuracy\_rf

########################################################################

##knn

library(class)

str(data)

##the normalization function is created

nor <-function(x) { (x -min(x))/(max(x)-min(x)) }

train\_nor <- as.data.frame(lapply(train[,c(1:13)], nor))

test\_nor <- as.data.frame(lapply(test[,c(1:13)], nor))

knn <- knn(train\_nor,test\_nor,cl=train$RainTomorrow,k=20)

##create the confucion matrix

cm\_knn<- table(knn,a)

#accuracy

n\_knn = sum(cm\_knn)

diag\_knn = diag(cm\_knn)

accuracy\_knn = sum(diag\_knn) / n\_knn

accuracy\_knn<-accuracy\_knn\*100

accuracy\_knn

##########################################################################################################################################

#K- Fold Cross validation to improve the accuracy . we take 10 folds and use it on ann model

nrFolds <- 10

# generate array containing fold-number for each sample (row)

folds <- rep\_len(1:nrFolds, nrow(data))

# actual cross validation

for(k in 1:nrFolds) {

# actual split of the data

fold <- which(folds == k)

data.train <- data[-fold,]

data.test <- data[fold,]

# train and test your model with data.train and data.test

}

########################################################################

##Result

Result<- data.frame("Algorithm" = c("Logistic","D\_tree","Random\_forest","knn"),

"Accuracy"=c(accuracy\_log,accuracy\_dt,accuracy\_rf,accuracy\_knn),

stringsAsFactors = FALSE)

Result