ANN with visualization

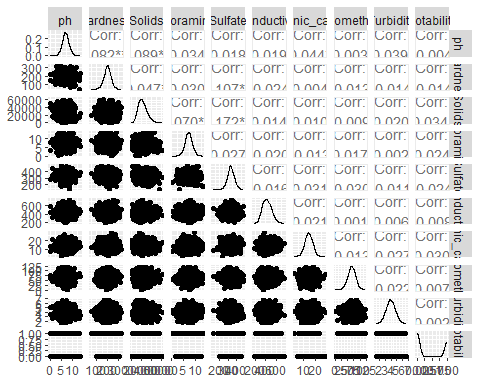
2022-12-28

setwd("C:\# R")

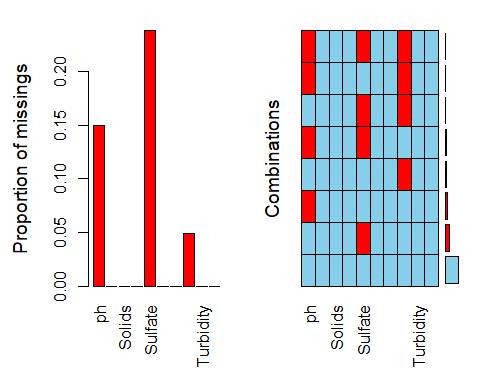
# read data  
  
df <- read.csv( "Water\_Quality.csv")  
  
# describe data  
library(psych)  
describe(df)

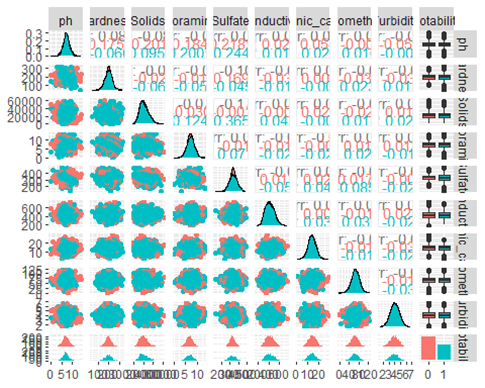
|  | vars | n | mean | sd | min | max | range | se |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ph | 1 | 2785 | 7.08 | 1.59 | 0.00 | 14.00 | 14.00 | 0.03 |
| Hardness | 2 | 3276 | 196.37 | 32.88 | 47.43 | 323.12 | 275.69 | 0.57 |
| Solids | 3 | 3276 | 22014.09 | 8768.57 | 320.94 | 61227.20 | 60906.25 | 153.20 |
| Chloramines | 4 | 3276 | 7.12 | 1.58 | 0.35 | 13.13 | 12.78 | 0.03 |
| Sulfate | 5 | 2495 | 333.78 | 41.42 | 129.00 | 481.03 | 352.03 | 0.83 |
| Conductivity | 6 | 3276 | 426.21 | 80.82 | 181.48 | 753.34 | 571.86 | 1.41 |
| Organic\_carbon | 7 | 3276 | 14.28 | 3.31 | 2.20 | 28.30 | 26.10 | 0.06 |
| Trihalomethanes | 8 | 3114 | 66.40 | 16.18 | 0.74 | 124.00 | 123.26 | 0.29 |
| Turbidity | 9 | 3276 | 3.97 | 0.78 | 1.45 | 6.74 | 5.29 | 0.01 |
| Potability | 10 | 3276 | 0.39 | 0.49 | 0.00 | 1.00 | 1.00 | 0.01 |

# explore data using plot panel  
library(GGally)  
ggpairs(df) # plots show no autocorrelation



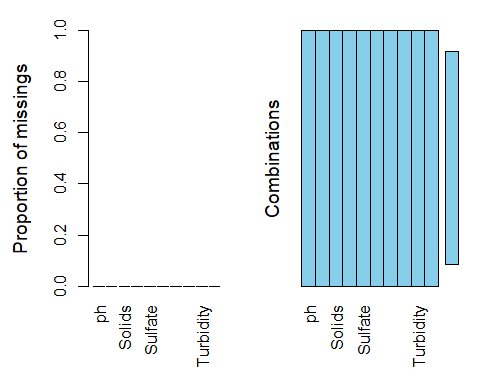
# find out missing data  
library(VIM)  
aggr(df)





# imputing data ( fill out the missing value) using Random forest ( rf) model  
  
library(mice)  
df\_imp <- mice(df,m=5, method="rf",maxit=5 )

df\_complete <- complete(df\_imp,1)  
  
# ensure that all missing values are imputed with the nearest match  
 aggr(df\_complete)



# convert potability to factor  
 df\_complete$Potability=factor(as.character(df\_complete$Potability))  
str(df\_complete$Potability)

## Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

# applying model svm  
 library(caret)  
  
 # split data to train and test  
   
 index <- createDataPartition(df\_complete$Potability,p=0.75,list=FALSE)  
  
df\_train <- df\_complete[index,]   
dim(df\_train) # to ensure proper split

## [1] 2458 10

str(df\_train)

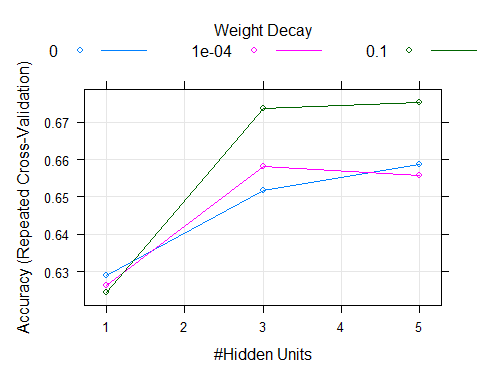
## 'data.frame': 2458 obs. of 10 variables:  
## $ ph : num 8.12 3.72 8.1 8.32 9.09 ...  
## $ Hardness : num 205 129 224 214 181 ...  
## $ Solids : num 20791 18630 19910 22018 17979 ...  
## $ Chloramines : num 7.3 6.64 9.28 8.06 6.55 ...  
## $ Sulfate : num 369 324 323 357 310 ...  
## $ Conductivity : num 564 593 419 363 398 ...  
## $ Organic\_carbon : num 10.4 15.2 16.9 18.4 11.6 ...  
## $ Trihalomethanes: num 87 56.3 66.4 100.3 32 ...  
## $ Turbidity : num 2.96 4.5 3.06 4.63 4.08 ...  
## $ Potability : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

df\_test <- df\_complete[-index,]   
dim(df\_test) # to ensure proper split

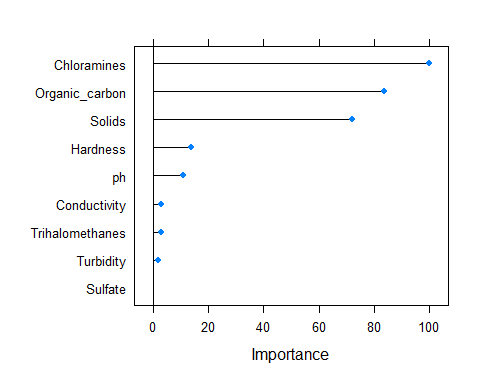
## [1] 818 10

# fit the model  
  
# prepare for cross validation  
  
tc <- trainControl(method="repeatedcv",number=10,repeats=5)  
  
# train model  
mod\_ann <- train(Potability~., df\_train, trControl =tc,method="pcaNNet")

plot(mod\_ann)



plot(varImp(mod\_ann)) #features can be selected based on variable importance.



Logistic regression with statistical

# set seed fo random number generator to keep results reproducible

set.seed(123455)

# Anomaly Detection in Water Quality

#setwd("C:\# R") # set working directory

df <- read.csv("Water\_Quality.csv") # read the input data

# Potability : water is safe for drinking

head(df)

df$Potability <- factor(df$Potability) # to covert numeric variable to factor variable for further analysis

# Describe data #

library(psych) # for function describe

describe(df)

describeBy(df,group=df$Potability) # describe the data sperately for the two levels of potability

# explore data using plot panel

#library(ggplot2)

library(GGally) # To explore the data visually , as scatter plots, histograms etc and also to see correlation

ggpairs(df) # plots show no autocorrelation, can be sued for modeling

ggpairs(df, aes(color=Potability)) # the plots will show potable and notpotable observations as separate colors

# data cleaning #

# Visualize missing data

library(VIM)

aggr(df) # visualize missing data form the data frame

# In sulphate , more than 20% data is missing. We can not omit the observations with the missing values.

# we will use 'rf- random forest model' to impute\* fill in) the missing data

# imputation with mice package

library(mice) # library for mice function for data imputation( Missing data Imputation using Chain Equations)

df\_imp <- mice(df, m=5,method="rf",maxit=5)

df\_imp$where # to see the places where impuattion has taken place. TRUE : imputed

# getting complete data

df\_complete <- complete(df\_imp)

# confirm imputation

aggr(df\_complete)

# confirm imputation quality

nrow(df)

# join original and imputed data

df\_both <- rbind(df,df\_complete)

nrow(df\_both)

# add column imputed: Ori/imp to distinguish original and imputed data and

df\_both$imputed <- rep(c("ori","imp"),each=nrow(df))

tail(df\_both$imputed )

#plot boxplots to compare original and imputed data for columns with missing data

par(mfrow=c(1,3)) # divide canvas into different sections

boxplot(ph~imputed,df\_both, main="pH",outcol="red")

boxplot(ph~imputed,df\_both, main="Sulfate",outcol="red")

boxplot(ph~imputed,df\_both, main="Turbidity",outcol="red")

# There is no difference in original and imputed data. We cab use imputed data set as original dataset for further analysis without any loss of reliability of results.

# fitting model to classify observations as potable or not.

library(caret) # library to implement many different statistical models.

# split data to train and test

index <- createDataPartition(df\_complete$Potability,p=0.75,list=FALSE)

df\_train <- df\_complete[index,]

dim(df\_train) # to ensure proper split

str(df\_train)

df\_test <- df\_complete[-index,]

dim(df\_test) # to ensure proper split

# fit the model #

# prepare for cross validation

tc <- trainControl(method="repeatedcv",number=10,repeats=5)

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

# train model " pcaNNET #

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

mod\_nnet <- train(Potability~., df\_train, trControl =tc,method="nnet")

# test model with test data

pred\_pot <- predict(mod\_nnet ,df\_test)

confusionMatrix(df\_test$Potability,pred\_pot)

plot(mod\_nnet )

plot(varImp(mod\_nnet)) #features can be selected based on variable importance.

varImp(mod\_nnet) # to get variable importance numeric values. Variable hardness has importance 100 and other variables iportabnce are as per the output.

# Accuracy of classification by nnet is low 0.6149. We will try another approach

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

# NNET with reduced features #

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

# Here we will use hardness, sulfate , solids, conductivity

mod\_nnet\_red <- train(Potability~., df\_train[,c(2,3,5,6,10) ], trControl =tc,method="nnet")

# test model with test data

pred\_pot <- predict(mod\_nnet ,df\_test)

confusionMatrix(df\_test$Potability,pred\_pot)

plot(mod\_nnet )

plot(varImp(mod\_nnet)) #features can be selected based on variable importance.

varImp(mod\_nnet) # to get variable importance numeric values. Variable hardness has importance 100 and other variables iportabnce are as per the output.

# Accuracy of classification by nnet after feature selection is low 0.6112. We will try another approach