Installing and loading libraries

```
library(readr)
library(caret)
library(klaR)
```

Setting the working directory

```
setwd("D:/great learning/Capstone/6. Coronory Heart Risk Study")
mydata = read.csv("Coronary_heart_risk_study.csv", header = TRUE)
attach(mydata)
```

Missing Value treatement/Imputation and plot of missing value

```
colSums(is.na(mydata))
sum(is.na(mydata))
plot_missing(mydata)
mydata$education[is.na(mydata$education)] = mean(mydata$education, na.rm = T)
mydata$cigsPerDay[is.na(mydata$cigsPerDay)] = median(mydata$cigsPerDay, na.rm = T)
mydata$BPMeds[is.na(mydata$BPMeds)] = median(mydata$BPMeds, na.rm = T)
mydata$totChol[is.na(mydata$totChol)] = median(mydata$totChol, na.rm = T)
mydata$BMI[is.na(mydata$BMI)] = median(mydata$BMI, na.rm = T)
mydata$heartRate[is.na(mydata$heartRate)] = median(mydata$heartRate, na.rm = T)
mydata$glucose[is.na(mydata$glucose)] = median(mydata$glucose, na.rm = T)
plot_missing(mydata)
```

Outlier Treatement

```
a = mydata$BMI
```

```
qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)
caps = quantile(a, probs=c(0.05,0.95), na.rm=T)
h = 1.5*IQR(a, na.rm=T)
a[a<(qnt[1]-h)] = caps[1]
a[a>(qnt[2]+h)] = caps[2]
print(a)
boxplot(a)
BMI = a
boxplot(BMI, horizontal = TRUE, main = "Boxplot for BMI after outlier treatement")

a = mydata$totChol
qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)
caps = quantile(a, probs=c(0.05,0.95), na.rm=T)
h = 1.5*IQR(a, na.rm=T)
a[a<(qnt[1]-h)] = caps[1]
a[a>(qnt[2]+h)] = caps[2]
print(a)
boxplot(a)
totChol = a
boxplot(totChol, horizontal = TRUE, main = "Boxplot for totChol after outlier treatement")

a = mydata$sysBP
qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)
caps = quantile(a, probs=c(0.05,0.95), na.rm=T)
h = 1.5*IQR(a, na.rm=T)
a[a<(qnt[1]-h)] = caps[1]
a[a>(qnt[2]+h)] = caps[2]
print(a)
boxplot(a)
```

```
sysBP = a
boxplot(sysBP, horizontal = TRUE, main = "Boxplot for sysBP after outlier treatement")

a = mydata$diaBP
qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)
caps = quantile(a, probs=c(0.05,0.95), na.rm=T)
h = 1.5*IQR(a, na.rm=T)
a[a<(qnt[1]-h)] = caps[1]
a[a>(qnt[2]+h)] = caps[2]
print(a)
boxplot(a)
diaBP = a
boxplot(diaBP, horizontal = TRUE, main = "Boxplot for diaBP after outlier treatement")

a = mydata$heartRate
qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)
caps = quantile(a, probs=c(0.05,0.95), na.rm=T)
h = 1.5*IQR(a, na.rm=T)
a[a<(qnt[1]-h)] = caps[1]
a[a>(qnt[2]+h)] = caps[2]
print(a)
boxplot(a)
heartRate = a
boxplot(heartRate, horizontal = TRUE, main = "Boxplot for heartRate after outlier treatement")

a = mydata$glucose
qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)
caps = quantile(a, probs=c(0.05,0.95), na.rm=T)
h = 1.5*IQR(a, na.rm=T)
```

```
a[a<(qnt[1]-h)] = caps[1]

a[a>(qnt[2]+h)] = caps[2]

print(a)

boxplot(a)

glucose = a

boxplot(glucose, horizontal = TRUE, main = "Boxplot for glucose after outlier treatement")


a = mydata$cigsPerDay

qnt = quantile(a, probs=c(0.25,0.75), na.rm=T)

caps = quantile(a, probs=c(0.05,0.95), na.rm=T)

h = 1.5*IQR(a, na.rm=T)

a[a<(qnt[1]-h)] = caps[1]

a[a>(qnt[2]+h)] = caps[2]

print(a)

boxplot(a)

cigsPerDay = a

boxplot(cigsPerDay, horizontal = TRUE, main = "Boxplot for cigsPerDay after outlier treatement")
```

Variable Conversion

```
mydata$male <- as.factor(mydata$male)

mydata$education <- as.factor(mydata$education)

mydata$age <- as.numeric(mydata$age)

mydata$currentSmoker <- as.factor(mydata$currentSmoker)

mydata$cigsPerDay <- as.numeric(mydata$cigsPerDay)

mydata$BPMeds <- as.factor(mydata$BPMeds)

mydata$prevalentStroke <- as.factor(mydata$prevalentStroke)

mydata$prevalentHyp <- as.factor(mydata$prevalentHyp)

mydata$diabetes <- as.factor(mydata$diabetes)
```

```
mydata$heartRate <- as.numeric(mydata$heartRate)

mydata$TenYearCHD <- as.factor(mydata$TenYearCHD)
```

Splitting the data into train and test data set

```
library(caTools)

split = sample.split(mydata$TenYearCHD, SplitRatio = 0.7)

trainDS = subset(mydata, split==TRUE)

testDS = subset(mydata, split==FALSE)
```

Building logistic regression model

```
library(lmtest)

glm(as.factor(trainDS$TenYearCHD)~., data = trainDS, family = binomial)

summary(glm(as.factor(trainDS$TenYearCHD)~., data = trainDS, family = binomial))

model1 = glm(as.factor(trainDS$TenYearCHD)~male+age+cigsPerDay+prevalentStroke+sysBP+glucose, data
= trainDS, family = binomial)

summary(model1$fitted.values)

plot(as.factor(model1$y), model1$fitted.values)

model1.validate = predict(model1, data = testDS, type = 'response')

model1.pred = ifelse(model1.validate>0.5,1,0)

table(model1$y, model1.pred)
```

ROC for logistic regression model

```
library(pROC)

prob = predict(model1, type = c("response"))

trainDS$prob = prob

roc_curve = roc(TenYearCHD~prob, data = trainDS)
```

```
plot(roc_curve)
```

AUC for logistic regression model

```
library(ROCR)
predobj.trainDS = prediction(trainDS$prob, trainDS$TenYearCHD)
auc_1 = performance(predobj.trainDS,"auc")
auc_1 = as.numeric(auc_1@y.values)
print(auc.train)
```

Building logistic regression model with SMOTE

```
library(DMwR)
table(TenYearCHD)
mydata$TenYearCHD = as.factor(mydata$TenYearCHD)
mydata.smote.balanced = SMOTE(TenYearCHD~.,mydata, perc.over = 3000, k = 4, perc.under = 250)
split = sample.split(mydata.smote.balanced$TenYearCHD, SplitRatio = 0.7)
smote.train = subset(mydata.smote.balanced, split == TRUE)
smote.test = subset(mydata.smote.balanced, split == FALSE)

glm(as.factor(smote.train$TenYearCHD)~., data = smote.train, family = binomial)
summary(glm(as.factor(smote.train$TenYearCHD)~., data = smote.train, family = binomial))
model2 = glm(as.factor(smote.train$TenYearCHD)~., data = smote.train, family = binomial)
summary(model2$fitted.values)
plot(as.factor(model2$y), model2$fitted.values)
model2.validate = predict(model2, data = smote.test, type = 'response')
model2.pred = ifelse(model2.validate>0.5,1,0)
table(model2$y, model2.pred)
```

ROC for logistic regression model with SMOTE

```
library(pROC)
prob1 = predict(model2, type = c("response"))
smote.train$prob = prob1
roc_curve = roc(TenYearCHD~prob, data = smote.train)
plot(roc_curve)
```

AUC for logistic regression model with SMOTE

```
library(ROCR)
predobj.train = prediction(smote.train$prob, smote.train$TenYearCHD)
auc_2 = performance(predobj.train,"auc")
auc_2 = as.numeric(auc_2@y.values)
print(auc_2)
```

 # define training control

```
train_control <- trainControl(method = "cv", number = 10)
```

 # train the model on training set

```
model3 <- train(TenYearCHD ~ .,data = smote.train,trControl = train_control,method =
"glm",family=binomial())
```

 # print cv scores

```
summary(model3)
pred = predict(model3, newdata=smote.test)
```

```
confusionMatrix(data=pred, smote.test$TenYearCHD)
```

ROC for logistic regression model with SMOTE and cross validation

```
library(pROC)
prob1 = predict(model3, type = c("prob"))[,"1"]
smote.train$prob = prob1
View(prob1)
roc_curve = roc(TenYearCHD~prob, data = smote.train)
plot(roc_curve)
```

AUC for logistic regression model with SMOTE and cross validation

```
library(ROCR)
predobj.train = prediction(smote.train$prob, smote.train$TenYearCHD)
auc_3 = performance(predobj.train,"auc")
auc_3 = as.numeric(auc_3@y.values)
print(auc_3)
```

Building naive bayes model

```
library(e1071)
naive.NB = naiveBayes(x = trainDS[,1:15], y = trainDS[,16])
pred.nb = predict(naive.NB, newdata = testDS)
pred.nb
confusionMatrix(data = pred.nb, reference = as.factor(testDS$TenYearCHD))
```

ROC for Naive Bayes

```
library(pROC)
prob1 = predict(naive.NB, newdata = testDS, type = c("raw"))[,"1"]
testDS$prob = prob1
View(prob1)
roc_curve = roc(TenYearCHD~prob, data = testDS)
plot(roc_curve)
```

AUC for Naive Bayes model

```
library(ROCR)
predobj.train = prediction(testDS$prob, testDS$TenYearCHD)
auc_4 = performance(predobj.train,"auc")
auc_4 = as.numeric(auc_4@y.values)
print(auc_4)
```

Naive Bayes model with SMOTE

```
library(e1071)
naive.NB = naiveBayes(x = smote.train[,1:15], y = smote.train[,c(16)])
pred.nb = predict(naive.NB, newdata = smote.test)
pred.nb
confusionMatrix(data = pred.nb, reference = smote.test$TenYearCHD)
```

ROC for Naive Bayes with SMOTE

```
library(pROC)
prob1 = predict(naive.NB, newdata = smote.test, type = c("raw"))[,"1"]
smote.test$prob = prob1
View(prob1)
```

```
roc_curve = roc(TenYearCHD~prob, data = smote.test)
plot(roc_curve)
```

AUC for Naive Bayes model with SMOTE

```
library(ROCR)
predobj.test = prediction(smote.test$prob, smote.test$TenYearCHD)
auc_5 = performance(predobj.test,"auc")
auc_5 = as.numeric(auc_5@y.values)
print(auc_5)
```

Naive bayes model with smote and cross validation

```
# define training control
train_control <- trainControl(method = "cv", number = 10)

# train the model on training set
model4 <- train(TenYearCHD ~ .,data = smote.train,trControl = train_control,method = "nb")

# print cv scores
summary(model4)


pred = predict(model4, newdata=smote.test)
confusionMatrix(data=pred, smote.test$TenYearCHD)
```

ROC for Naive Bayes with SMOTE and cross validation

```
library(pROC)
```

```
prob1 = predict(model4, type = c("prob"))[,"1"]

smote.train$prob = prob1

View(prob1)

roc_curve = roc(TenYearCHD~prob, data = smote.train)

plot(roc_curve)
```

AUC for Naive Bayes model with SMOTE and cross validation

```
library(ROCR)

predobj.train = prediction(smote.train$prob, smote.train$TenYearCHD)

auc_6 = performance(predobj.train,"auc")

auc_6 = as.numeric(auc_6@y.values)

print(auc_6)
```

Random Forest analysis with train data set

```
print(sum(trainDS$TenYearCHD=="1")/nrow(trainDS))


library(randomForest)

seed=1000

set.seed(seed)

rndforest = randomForest(trainDS$TenYearCHD~., data = trainDS, ntree = 501, mtry = 3, nodesize = 10,
importance = TRUE)

print(rndforest)

print(rndforest$err.rate)

plot(rndforest)

importance(rndforest)

trainDS$predict.class = predict(rndforest, trainDS, type = "class")

trainDS$prob = predict(rndforest, trainDS, type = "prob")[,"1"]
```

```
head(trainDS)

confusionMatrix(trainDS$TenYearCHD, trainDS$predict.class)
```

Tuning

```
set.seed(seed)

trndforest = tuneRF(x=trainDS, y=trainDS$TenYearCHD, mtryStart = 3, stepFactor = 1.5, ntreeTry = 25, improve = 0.001, trace = TRUE, plot = TRUE, doBest = TRUE, importance = TRUE)

trainDS$predict.class = predict(trndforest, trainDS, type = "class")

trainDS$prob = predict(trndforest, trainDS, type = "prob")[,"1"]

head(trainDS)

confusionMatrix(trainDS$TenYearCHD, trainDS$predict.class)
```

Testing the built Random Forest Model with test data set

```
testDS$predict.class = predict(rndforest, testDS, type = "class")

testDS$prob = predict(rndforest, testDS, type = "prob")[,"1"]

head(testDS)

confusionMatrix(testDS$TenYearCHD, testDS$predict.class)
```

ROC

```
library(ROCR)

library(ineq)

predobj.trainDS = prediction(trainDS$prob, trainDS$TenYearCHD)

perf.train_rf = performance(predobj.trainDS, "tpr", "fpr")

predobj.testDS = prediction(testDS$prob, testDS$TenYearCHD)

perf.test_rf = performance(predobj.testDS, "tpr", "fpr")

plot(perf.train_rf)
```

```
plot(perf.test_rf)
```

auc

```
auc.train2 = performance(predobj.trainDS,"auc")
auc.train2 = as.numeric(auc.train2@y.values)
print(auc.train2)

auc.test2 = performance(predobj.testDS,"auc")
auc.test2 = as.numeric(auc.test2@y.values)
print(auc.test2)
```

Random Forest analysis with SMOTE

```
library(caTools)
library(DMwR)
table(TenYearCHD)
mydata$TenYearCHD = as.factor(mydata$TenYearCHD)
mydata.smote.balanced = SMOTE(TenYearCHD~.,mydata, perc.over = 3000, k = 4, perc.under = 250)
split = sample.split(mydata.smote.balanced$TenYearCHD, SplitRatio = 0.7)
smote.train = subset(mydata.smote.balanced, split == TRUE)
smote.test = subset(mydata.smote.balanced, split == FALSE)

library(randomForest)
seed=1000
set.seed(seed)
rndforest = randomForest(smote.train$TenYearCHD~., data = smote.train, ntree = 501, mtry = 3, nodesize = 10,
importance = TRUE)
print(rndforest)
```

```
print(rndforest$err.rate)

plot(rndforest)

importance(rndforest)

smote.train$predict.class = predict(rndforest, smote.train, type = "class")

smote.train$prob = predict(rndforest, smote.train, type = "prob")[,"1"]

head(smote.train)

confusionMatrix(smote.train$TenYearCHD, smote.train$predict.class)
```

Tuning

```
seed = 1000

set.seed(seed)

trndforest = tuneRF(x=smote.train, y=smote.train$TenYearCHD, mtryStart = 3, stepFactor = 1.25, ntreeTry = 25, improve = 0.0001, trace = TRUE, plot = TRUE, doBest = TRUE, importance = TRUE)

smote.train$predict.class = predict(trndforest, newdata = smote.train, type = "class")

trainDS$prob = predict(trndforest, trainDS, type = "prob")[,"1"]

head(smote.train)

confusionMatrix(smote.train$TenYearCHD, smote.train$predict.class)
```

Testing the built Random Forest Model with test data set

```
smote.test$predict.class = predict(rndforest, smote.test, type = "class")

smote.test$prob = predict(rndforest, smote.test, type = "prob")[,"1"]

head(smote.test)

confusionMatrix(smote.test$TenYearCHD, smote.test$predict.class)
```

ROC

```
library(ROCR)
```

```
predobj.testDS = prediction(smote.test$prob, smote.test$TenYearCHD)

perf.test_rf = performance(predobj.testDS, "tpr", "fpr")

plot(perf.test_rf)
```

auc

```
auc.test2 = performance(predobj.testDS,"auc")

auc.test2 = as.numeric(auc.test2@y.values)

print(auc.test2)
```