PROJECT 4.1

BREAST CANCER – CLASSIFICATION MODEL

SCRIPT

```
setwd("C:/Users/tsraj/Desktop/Acadgild students projects/project4")
library(readr)
CancerData <- read_csv("CancerData.csv")
print(paste("rows:", nrow(df), "cols:", ncol(CancerData)))
View(CancerData)
summary(CancerData)
dim(CancerData)
names(CancerData)
#CancerData<- CancerData[-1]
CancerData$diagnosis <- factor(CancerData$diagnosis, levels = c("B", "M"),
                               labels = c("Benign", "Malignant"))
names(CancerData)
library(mice)
library(readr,dplyr)
library("ggplot2")
library("corrplot")
library("gridExtra")
library("pROC")
library("MASS")
library("caTools")
library("caret")
library(randomForest)
library(rpart)
library(rpart.plot)
library(rattle)
library(ggplot2)
library(Amelia)
```

```
library(class)
library(gmodels)
missmap(CancerData, main="Missing Data Map", col=c("#FF4081", "#3F51B5"),
      legend=FALSE)
data<-CancerData
data[,33]<-NULL

barplot(table(data$diagnosis), xlab = "Type of tumor", ylab="Numbers per type")

str(data)
any(is.na(data))
# visualize the missing values using the missing map from the Amelia package
missmap(data,col=c("yellow","red"))
data$diagnosis<-as.factor(data$diagnosis)
summary(data)
qplot(radius_mean, data=data, colour=diagnosis, geom="density",
    main="Radius mean for each tumor type")
qplot(smoothness_mean, data=data, colour=diagnosis, geom="density",
    main="Smoothness mean for each tumor type")
qplot(concavity_mean, data=data, colour=diagnosis, geom="density",
    main="Concavity mean for each tumor type")
qplot(area_worst , data=data, colour=diagnosis, geom="density",
    main="area worst for each tumor type")

# Looking at distribution for area.mean variable
plot.new()
hist(CancerData$area_mean,
   main = 'Distribution of Cell Area Means',
   xlab = 'Mean Area',
   col = 'green')
#we find that the data is imbalanced and also there is a lot of corelation between the attributes
```

```
## we find that there are no missing values

## we find that data is little unbalanced

prop.table(table(data$diagnosis))

## we then show some correlation

corr_mat<-cor(data[,3:ncol(data)])

corrplot(corr_mat)


plot.new()

plot(data$area_mean ~data$concavity_mean)

title('Basic Scatterplot')

ggplot(data, aes(x=data$area_worst)) + geom_histogram(binwidth = 1, fill = "yellow", color =
"black")

ggplot(data, aes(x=data$area_mean)) + geom_histogram(binwidth = 1, fill = "green", color = "red")


#Modelling

#We are going to get a training and a testing set to use when building some models:

set.seed(1234)

data_index<-createDataPartition(data$diagnosis,p=0.75,list = FALSE)

train_data<-data[data_index,-1]

test_data<-data[data_index,-1]


## Applying learning models

fitControl <- trainControl(method="cv",

                           number = 5,

                           preProcOptions = list(thresh = 0.99), # threshold for pca preprocess

                           classProbs = TRUE,

                           summaryFunction = twoClassSummary)

#Model1: Random Forest

#Building the model on the training data

## random forest

model_rf <- train(diagnosis~.,
```

```
            train_data,
            method="ranger",
            metric="ROC",
            #tuneLength=10,
            #tuneGrid = expand.grid(mtry = c(2, 3, 6)),
            preProcess = c('center', 'scale'),
            trControl=fitControl)

#Testing on the testing data
## testing for random forets
pred_rf <- predict(model_rf, test_data)
cm_rf <- confusionMatrix(pred_rf, test_data$diagnosis, positive = "M")
cm_rf
# We find the accuracy of the model is 100%
#Random forest model- takes decision trees and averages them
normalize<-function(x){return((x-min(x))/(max(x)-min(x)))}
data$diagnosis<-as.numeric(data$diagnosis)
data_n<-as.data.frame(lapply(data,normalize))
traindata_n<--data_n[1:426,]
testdata_n<-data_n[427:569,]
rf <- randomForest(diagnosis ~., data= traindata_n, ntree =300, mtry = 5, importance = TRUE)
print(rf)
plot.new()
varImpPlot(rf, type = 1, pch =8, col = 2, cex =0.8, main = "cancerdata")
abline(v= 45, col= "red")
library(party)
#cf1 <- cforest(diagnosis ~ . , data=traindata_n , control=fitControl(mtry=5,ntree=300)) # fit the
random forest


#varimp(cf1) # get variable importance, based on mean decrease in accuracy
```

```
#varimp(cf1, conditional=TRUE)  # conditional=True, adjusts for correlations between predictors


#varimpAUC(cf1)  # more robust towards class imbalance.

library(Boruta)

# Decide if a variable is important or not using Boruta


boruta_output <- Boruta( diagnosis~ ., data=na.omit(train_data), doTrace=2)  # perform Boruta search

boruta_signif <- names(boruta_output$finalDecision[boruta_output$finalDecision %in% c("Confirmed", "Tentative")])

boruta_signif

#Model2: Naive Bayes

#Building and testing the model

model_nb <- train(diagnosis~.,

                  train_data,

                  method="nb",

                  metric="ROC",

                  preProcess=c('center', 'scale'),

                  trace=FALSE,

                  trControl=fitControl)

## predicting for test data

pred_nb <- predict(model_nb, test_data)

cm_nb <- confusionMatrix(pred_nb, test_data$diagnosis, positive = "M")

cm_nb


#Accuracy of the model is 93.9%

#Model3: glm

#Building and testing the model

model_glm <- train(diagnosis~.,

                  train_data,

                  method="glm",

                  metric="ROC",
```

```
            preProcess=c('center', 'scale'),
            trace=FALSE,
            trControl=fitControl)

## predicting for test data
pred_glm <- predict(model_glm, test_data)
cm_glm <- confusionMatrix(pred_glm, test_data$diagnosis, positive = "M")
cm_glm
#Accuracy of the model is 98.3%
#algorithm for decision tree
library(C50)
data$diagnosis<-as.factor(data$diagnosis)
tree <- C5.0( diagnosis~., data = data)
summary(tree)
plot.new()
plot(tree)
results <- C5.0(diagnosis ~., data = data, rules = TRUE)
summary(results)
data<-as.data.frame(data)
library(rpart)
tree<-rpart(diagnosis~.,data =train_data,method="class")
plot(tree)
text(tree, pretty=0)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
plot.new()
fancyRpartPlot(tree)
plot.new()
printcp(tree)
plotcp(tree)
```

```
ptree<- prune(tree, cp= tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])
plot.new()
fancyRpartPlot(ptree, uniform=TRUE,main="Pruned Classification Tree")
library(rpart)
fit1 <- rpart(diagnosis~.,data=train_data)
fit1
summary(fit1)
#Kernlab Classification
require(kernlab)
installed.packages("kernlab")
library(kernlab)
data_classifier<-ksvm(diagnosis ~., data =train_data , kernel='vanilladot')
data_classifier
data_predictions<-predict(data_classifier,test_data)
head(data_predictions)
table(data_predictions, test_data$diagnosis)
agreement<-data_predictions == test_data$diagnosis
table(agreement)
prop.table(table(agreement))
agreement
set.seed(12345)
data_classifier_rbf<-ksvm(diagnosis ~., data = train_data, kernel='rbfdot')
data_predictions_rbf<-predict(data_classifier_rbf,test_data)
agreement_rbf<-data_predictions_rbf == test_data$diagnosis
table(agreement_rbf)
prop.table(table(agreement_rbf))


# logistic regression model:
fit <- glm(diagnosis~.,data = train_data,family = binomial(link='logit'))
summary(fit)
```

```
library(MASS)
step_fit <- stepAIC(fit,method='backward')
summary(step_fit)
confint(step_fit)
#ANOVA on base model
anova(fit,test = 'Chisq')
#ANOVA from reduced model after applying the Step AIC
anova(step_fit,test = 'Chisq')

#plot the fitted model
plot.new()
plot(fit$fitted.values)
pred_link <- predict(fit,newdata = test_data,type = 'link')
#check for multicollinearity
library(car)
vif(fit)
vif(step_fit)
pred <- predict(fit,newdata =test_data ,type ='response')
#check the AUC curve
library(pROC)
g <- roc(diagnosis ~ pred, data = test_data)
g
plot.new()
plot(g)
library(caret)
#with default prob cut 0.50
test_data$pred_diagnosis <- ifelse(pred<0.5,'yes','no')

table(test_data$pred_diagnosis,test_data$diagnosis)

#training split of diagnosis classes
```

```
round(table(train_data$diagnosis)/nrow(train_data),2)*100
# test split of diagnosis
round(table(test_data$diagnosis)/nrow(test_data),2)*100
#predicted split of diagnosis
round(table(test_data$pred_diagnosis)/nrow(test_data),2)*100
#create confusion matrix
#confusionMatrix(test_data$diagnosis,test_data$pred_diagnosis)
#how do we create a cross validation scheme
control <- trainControl(method = 'repeatedcv',
                        number = 10,
                        repeats = 3)
seed <-7
metric <- 'Accuracy'
set.seed(seed)
fit_default <- train(diagnosis~.,
                     data = train_data,
                     method = 'glm',
                     metric =metric ,
                     trControl = control)
print(fit_default)
library(caret)
varImp(step_fit)
varImp(fit_default)
library(woe)

library(riv)
train_data<-as.data.frame(train_data)
iv_df <- iv.mult(train_data, y="diagnosis", summary=TRUE, verbose=TRUE)
iv_df
iv <- iv.mult(train_data, y="diagnosis", summary=FALSE, verbose=TRUE)
# Plot information value summary
```

```
iv.plot.summary(iv_df)



#4. MARS (earth package)
#The earth package implements variable importance based on Generalized cross validation (GCV),
#number of subset models the variable occurs (nsubsets) and residual sum of squares (RSS).
library(earth)
marsModel<-earth(diagnosis~ ., data=data) # build model
ev <- evimp (marsModel) # estimate variable importance
ev
plot.new()
plot (ev)
```