HarvardX: PH125.9x Data Science
Breast Cancer Prediction Project

Jananya Sivakumar

I. Overview

HarvardX: Data Science course deals with the different ways to explore the data given, visualize the data using R packages and various modelling approaches employing Machine Learning techniques. The objective of the Cancer prediction project is to build a system that predicts the likeliness of an indivudual to have breast cancer by considering the biological attributes that are significant in the identification.

1. Introduction

One of the most common causes od cancer deaths in women are caused by Breast Cancer. One of the warning symptoms of breast cancer is the development of a tumor in the breast. A tumor, however, could be either benign or malignant. Breast cancer is the most common type of cancer and has always been a threat to women's lives. Early diagnosis requires an effective method to predict cancer to allow physicians to distinguish benign and malicious cancer. By determining which cytological attributes are significant in identifying benign and malignant, this project aims to predict whether an individual has breast cancer tumors.

2. Aim of the Project

The aim of this project is to train a Machine Learning algorithm to predict the occurance of breast cancer in women by estimating the accuracy of all the models and selecting the best model for cancer prediction.

3. Dataset

The experimental study is based on the Wisconsin Breast Cancer database from the UC Irvine Machine Learning Repository. https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data

```
#Loading the dataset
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-
cancer-wisconsin/breast-cancer-wisconsin.data"
data <- read.csv(file = url, header = FALSE,
                 col.names = c("ID","clump_thickness", "uniformity_size",
"uniformity_shape", "marginal_adhesion", "single_epithelial_cell_size",
"bare_nuclei", "bland_chromatin", "normal_nucleoli","mitoses", "diagnosis"))

#Attaching the required R packages
library(gmodels)
library(ggplot2)
library(reshape)
```

```
library(reshape2)
library(corrplot)
library(rpart.plot)
library(randomForest)
library(class)
library(dplyr)
library(C50)
library(caTools)
library(kableExtra)

#Data Cleansing
data <- select(data, -1)
data <- data[data$bare_nuclei != "?",] %>% mutate(bare_nuclei =
as.integer(as.character((bare_nuclei))))
data <- data %>% mutate(diagnosis = ifelse(diagnosis == 2, 0, 1),
                        diagnosis = as.factor(diagnosis))
```

II. Methods

1. Data Preparation

Preparation of data is carried out by generating the validation and trainig datasets to design and test the algorithms. Training, developing and selection of the modelling algorithms is done using the training dataset.

```
set.seed(3011)
train_index <- sample(nrow(data), size = round(0.75 * nrow(data)), replace =
FALSE)
train <- data[train_index,]
test <- data[-train_index,]
```

2. Data Exploration

Analysis of the data is done by exploring the given data.

Structure

The internal structure of the R object is displayed along with the dimention of the dataframe. The names of the column heads in the dataframe are also printed.

```
str(data)
dim(data)
names(data)
```

Viewing the data

Displaying the first 6 rows present in the input data frame and the summary of the data. The standard deviation of the epithelial cell size and the summary of the column including the minimum, maximum values and the quartile medians of the data is displayed.

```
head(data)
summary(data)
```

```
sd(data$single_epithelial_cell_size)
summary(data$single_epithelial_cell_size)
```

Computation

Computation measure (mean, median, min, max, etc..) or a function for each factor variable in a vector. It creates a subset and apply functions to each of the subset. The total observation of the data is calculated. An implementation of a cross-tabulation function with output similar to S-Plus crosstabs() and SAS Proc Freq (or SPSS format) with Chi-square, Fisher and McNemar tests of the independence of all table factors. Correlation of the differant attributes with each other is computed.

```
#Sum and Mean
tapply(data$clump_thickness, data$diagnosis, sum)
tapply(data$clump_thickness, data$diagnosis, mean)

#CrossTable
CrossTable(data$uniformity_size,data$uniformity_shape)

#Correlation
correlation <- cor(data[,-10])
corrplot(correlation, type = "upper", col = c("#fcbba1", "#b2d2e8"),
addCoef.col = "black", tl.col = "black")
```

3. Data Visualization

```
#Segmentation of the diagnosis
qplot(diagnosis, data = data, fill = I("darkblue"))

#Plotting the thickness of the Clump
qplot(clump_thickness, data = data, facets = diagnosis ~ .,bins=30)

#Jitter plot of uniformity size attribute grouped by diagnosis
qplot(diagnosis, uniformity_size, data = data, geom = "jitter")

#Box and Jitter plot of clump thickness attribute grouped by diagnosis
qplot(diagnosis, clump_thickness, data = data, geom = c("boxplot", "jitter"),
alpha = I(1/5))

#Density of bland chromatin filling in by mitosis attribute
qplot(bland_chromatin, data = data, fill = mitoses, geom = "density",alpha =
I(1/2))
```

III. Modelling Approaches

Decision Tree

Decision tree is a type of supervised learning algorithm that can be used in both regression and classification problems. It works for both categorical and continuous input and output variables. The decision tree model is run through a grid search of minsplit (the minimum number of observations in each split) and maxdepth (the maximum depth of the tree) in order to find the optimized combination of hyperparameters.

```
train_tree <- 0
test_tree <- 0
Dtree <- data.frame(train_tree = numeric(), test_tree = numeric())

set.seed(3011)
tree_parameters <- data.frame(minsplit_para = floor(runif(8, 10, 60)),
                              maxdepth_para = floor(runif(8, 10, 30)))

for(para_comb in 1:nrow(tree_parameters)){
  decision_tree <- rpart(diagnosis ~ .,  data = train,
                         control = rpart.control(minsplit =
tree_parameters[para_comb, "minsplit_para"],
                                                 maxdepth =
tree_parameters[para_comb, "maxdepth_para"]))

  pred_train_tree <- as.data.frame(predict(decision_tree, train,
type='prob'))
  train_tree <- roc(train$diagnosis, pred_train_tree$`1`, percent = TRUE,
plot = TRUE)

  pred_test_tree <- as.data.frame(predict(decision_tree, test, type='prob'))
  test_tree <- roc(test$diagnosis, pred_test_tree$`1`, percent = TRUE, plot =
TRUE)

  Dtree[para_comb, ] <- c(round(train_tree$auc, 2), round(test_tree$auc, 2))
  train_tree = ifelse(train_tree > train_tree$auc, train_tree,
train_tree$auc)
  test_tree = ifelse(test_tree > test_tree$auc, test_tree, test_tree$auc)
}

# Minsplit of 11 and Maxdepth of 10.
best_decision_tree <- rpart(diagnosis ~., data = train,
                            control = rpart.control(minsplit = 11,
                                                    maxdepth = 10))
rpart.plot(x = best_decision_tree, box.palette="RdBu", shadow.col="gray",
nn=TRUE, yesno = 2)
```

Random Forest

In the random forest approach, a large number of decision trees are created. Every observation is fed into every decision tree. The most common outcome for each observation is used as the final output. A new observation is fed into all the trees and taking a majority vote for each classification model. The hyperparameters used are nodesize (the minimum number of observations in the terminal nodes), sampsize (the sample size of each tree), mtry (the number of variables to be considered for each tree), and ntree (the number of decision trees that constitute the forest).

```
train_bestrf <- 0
test_bestrf <- 0
rf <- data.frame(train_rf = numeric(), test_rf = numeric())

set.seed(160)
rf_parameters <- data.frame(nodesize = round(runif(10,5,20)),
                            sampsize= round(runif(10,1,400)),
                            mtry = round(runif(10,1,10)),
                            ntree = round(runif(10,1,400)))

for(paracomb_rf in 1:nrow(rf_parameters)){
  random_forest <- randomForest(diagnosis ~ ., data = train,
                                nodesize = rf_parameters[paracomb_rf,
"nodesize"],
                                sampsize = rf_parameters[paracomb_rf,
"sampsize"],
                                mtry = rf_parameters[paracomb_rf, "mtry"],
                                ntree = rf_parameters[paracomb_rf, "ntree"])

  pred_train_rf <- as.data.frame(predict(random_forest, train, type='prob'))
  train_rf <- roc(train$diagnosis, pred_train_rf$`1`, percent = TRUE, plot =
TRUE)

  pred_test_rf <- as.data.frame(predict(random_forest, test, type='prob'))
  test_rf <- roc(test$diagnosis, pred_test_rf$`1`, percent = TRUE, plot =
TRUE)

  rf[paracomb_rf, ] <- c(round(AUC_train_rf$au, 2), round(AUC_test_rf$auc,
2))
  train_bestrf = ifelse(train_bestrf > train_rf$auc, train_bestrf,
train_rf$auc)
  test_bestrf = ifelse(test_bestrf > test_rf$auc, test_bestrf, test_rf$auc)
}

# nodesize of 9, sampsize of 329, mtry of 7, and ntree of 210.
best_random_forest <- randomForest(diagnosis ~ ., data = train,
                                   nodesize = 9,
                                   sampsize = 329,
                                   mtry = 7,
```

```
                                    ntree = 210)
best_random_forest
varImpPlot(best_random_forest)
```

KNN Classifier

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

```
data$Clump_Thickness=as.numeric(data$clump_thickness)
data$Uniformity_CellSize=as.numeric(data$uniformity_size)
data$Uniformity_CellShape=as.numeric(data$uniformity_shape)
data$Marginal_Adhesion=as.numeric(data$marginal_adhesion)
data$Single_Epithelial_cellSize=as.numeric(data$single_epithelial_cell_size)
data$BareNuclei=as.numeric(data$bare_nuclei)
data$Bland_Chromatin=as.numeric(data$bland_chromatin)
data$Normal_Nucleoli=as.numeric(data$normal_nucleoli)
data$Mitoses=as.numeric(data$mitoses)
data$Diagnosis=as.factor(data$diagnosis)

sub <- sample(nrow(data), floor(nrow(data) * 0.75))

train.class=data[sub,11]
test.class<- data[-sub,11]

predict<-knn(train, test,train.class,k = 10)
table(test.class,predict)
```

##K means Clustering and C4.5 decision tree classifier Classification accuracy of C4.5 is improved with K means Clustering and adding the tested data dynamically to the training set.

```
set.seed(100)
malignantdata=subset(data,Diagnosis==0)
benigndata=subset(data,Diagnosis==1)
str(malignantdata)

k=2;
KMC = kmeans(malignantdata[ ,3:11], centers = k, iter.max = 1000)
KMC$cluster

malignantdata$Class=KMC$cluster
benigndata$Class=3

data<-rbind(malignantdata,benigndata)
```

```
data
str(data)
data$Class=as.factor(data$Class)
str(data)
treeModel<-C5.0(x=data[,3:11],y=data$Class)
summary(treeModel)
```

IV. Results

```
models_list <- list(Decision_Tree=Dtree,
                    Random_Forest=rf,
                    KNN=predict,
                    kmc=KMC)
models_results <- resamples(models_list)
summary(models_results)

confusionmatrix_list <- list(
  Decision_Tree=Dtree,
  Random_Forest=rf,
  KNN=predict,
  kmc=KMC)
confusionmatrix_list_results <- sapply(confusionmatrix_list, function(x)
x$byClass)
confusionmatrix_list_results %>% knitr::kable()

confusionmatrix_results_max <- apply(confusionmatrix_list_results, 1,
which.is.max)
output_report <- data.frame(metric=names(confusionmatrix_results_max),

best_model=colnames(confusionmatrix_list_results)[confusionmatrix_results_max
],
                            value=mapply(function(x,y)
{confusionmatrix_list_results[x,y]},
                                         names(confusionmatrix_results_max),
                                         confusionmatrix_results_max))
rownames(output_report) <- NULL
output_report
```

V. Conclusion

This paper treats the Wisconsin Madison Breast Cancer diagnosis problem as a pattern classification problem. Investigation of several machine learning languages has given the accuracy report of KNN Classifier as: 95.4% and K means Clustering and C4.5 decision tree classifier as: 95.1%