CIA LAB EXAM

MANOJ KUMAR - 2048015

4/30/2021

## Problem Description

The DublinTest is the provided Binary Classification dataset, where we have to classify the Target label 'Outcome' using other 7 variables namely BloodPressure, RBS, FBS, Serum.Insulin, BMI, BUN, and Age. Initailly we have to do neccessary Exploratory data analysis work to sort messy data and furture proceeding with feature seletions to get appropriate features to classify our dataset.

## Dataset Understanding

The DublinTest dataset has 8 variables with 680 observations, considered to be an Binary Classification dataset sicne it's consists of 1 Target column and 7 Regresses.

*Blood Pressure*: This records the systolic blood pressure in the arteries when the heartbeats.

*RBS*: Random Blood Sugar testing measures the levels of glucose in the blood at any given point in the day.

*FBS*: Fasting Blood sugar measures your blood sugar after an overnight fast.

*Serum Insulin*: Insulin test, used to measure the amount of insulin in the body.

*BMI*: Body Mass Index is a reliable indicator of body fatness.

*BUN*: Blood Urea Nitrogen level of the person.

*Age*: Age of the patient.

*Outcome*: class variable

## 1. Import the DublinTest Dataset and load the necessary packages.

library(RColorBrewer) # color  
library(funModeling) # histogram

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':  
##   
## format.pval, units

## funModeling v.1.9.4 :)  
## Examples and tutorials at livebook.datascienceheroes.com  
## / Now in Spanish: librovivodecienciadedatos.ai

library(Amelia) # missmap

## Loading required package: Rcpp

## ##   
## ## Amelia II: Multiple Imputation  
## ## (Version 1.7.6, built: 2019-11-24)  
## ## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell  
## ## Refer to http://gking.harvard.edu/amelia/ for more information  
## ##

library(tidyverse) # duplicate values

## ── Attaching packages ───────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ tibble 3.0.3 ✓ dplyr 1.0.4  
## ✓ tidyr 1.1.1 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.1  
## ✓ purrr 0.3.4

## ── Conflicts ──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::src() masks Hmisc::src()  
## x dplyr::summarize() masks Hmisc::summarize()

library(mice) # impute missing values

##   
## Attaching package: 'mice'

## The following object is masked from 'package:stats':  
##   
## filter

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(caret) # Classification and regression

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

## The following object is masked from 'package:survival':  
##   
## cluster

library(randomForest) # Random forest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(dplyr) # Remove duplicate rows  
library(e1071)

##   
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':  
##   
## impute

dataset <- read.csv("DublinTest dataset.csv")  
head(dataset)

## BloodPressure RBS FBS Serum.Insulin BMI BUN Age Outcome  
## 1 117 92 0 0 34.1 0.337 38 0  
## 2 109 75 26 0 36.0 0.546 60 0  
## 3 158 76 36 245 31.6 0.851 28 1  
## 4 88 58 11 54 24.8 0.267 22 0  
## 5 92 92 0 0 19.9 0.188 28 0  
## 6 122 78 31 0 27.6 0.512 45 0

## 2.Explore the Descriptive Analysis and visualize the data using plots

# Dimension of dataset  
  
dim(dataset)

## [1] 681 8

#### Insights

- The DublinTest dataset has 8 variables with 680 observations  
 - All the 8 variables are numberical in nature.

# Check data structure  
  
str(dataset)

## 'data.frame': 681 obs. of 8 variables:  
## $ BloodPressure: int 117 109 158 88 92 122 103 138 102 90 ...  
## $ RBS : int 92 75 76 58 92 78 60 76 76 68 ...  
## $ FBS : int 0 26 36 11 0 31 33 0 37 42 ...  
## $ Serum.Insulin: int 0 0 245 54 0 0 192 0 0 0 ...  
## $ BMI : num 34.1 36 31.6 24.8 19.9 27.6 24 33.2 32.9 38.2 ...  
## $ BUN : num 0.337 0.546 0.851 0.267 0.188 0.512 0.966 0.42 0.665 0.503 ...  
## $ Age : int 38 60 28 22 28 45 33 35 46 27 ...  
## $ Outcome : int 0 0 1 0 0 0 0 0 1 1 ...

#### Insights

- BloodPressure, RBS, FBS, Serum, Insulin, Age, and Outcome are Numberic and Discrite in nature.  
 - BMI and BUN are Numberic and countinuous in nature.  
 - Clearnly, Outcome is our Target classification variable

# Check data structure  
  
summary(dataset)

## BloodPressure RBS FBS Serum.Insulin   
## Min. : 0.0 Min. : 0.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 99.0 1st Qu.: 64.0 1st Qu.: 0.00 1st Qu.: 0.00   
## Median :117.0 Median : 72.0 Median :23.00 Median : 30.50   
## Mean :120.9 Mean : 69.1 Mean :20.64 Mean : 80.05   
## 3rd Qu.:141.0 3rd Qu.: 80.0 3rd Qu.:32.25 3rd Qu.:125.00   
## Max. :198.0 Max. :122.0 Max. :99.00 Max. :846.00   
## NA's :1 NA's :1 NA's :1 NA's :1   
## BMI BUN Age Outcome   
## Min. : 0.00 Min. :0.0780 Min. :21.00 Min. :0.0000   
## 1st Qu.:27.30 1st Qu.:0.2487 1st Qu.:24.00 1st Qu.:0.0000   
## Median :32.00 Median :0.3815 Median :30.00 Median :0.0000   
## Mean :32.05 Mean :0.4782 Mean :33.59 Mean :0.3632   
## 3rd Qu.:36.60 3rd Qu.:0.6275 3rd Qu.:41.00 3rd Qu.:1.0000   
## Max. :67.10 Max. :2.4200 Max. :81.00 Max. :1.0000   
## NA's :1 NA's :1 NA's :1 NA's :1

#### Insights

- Summary is used to get basic stastistical report for all the 8 variables.  
 - BloodPressure has the 0.0 value as minimum, and 198 value as maximum.  
 - RBS has the 0.0 value as minimum, and 122 value as maximum.  
 - FBS has the 0.0 value as minimum, and 99 value as maximum.  
 - Serum.Insulin has the 0.0 value as minimum, and 846 value as maximum.  
 - BMI has the 0.0 value as minimum, and 67 value as maximum.  
 - BUN has the 0.0780 value as minimum, and 2.4200 value as maximum.  
 - Age has the 21.0 value as lowest, and 81 value as maximum.  
 - Outcome has 0 and 1 as class label value.

# Total missing count  
  
colSums(is.na(dataset))

## BloodPressure RBS FBS Serum.Insulin BMI   
## 1 1 1 1 1   
## BUN Age Outcome   
## 1 1 1

#### Insights

- All the variables having 1 NULL values in the DublinTest dataset.

dataset = na.omit(dataset)

# Total missing count  
  
colSums(is.na(dataset))

## BloodPressure RBS FBS Serum.Insulin BMI   
## 0 0 0 0 0   
## BUN Age Outcome   
## 0 0 0

#Checking for Empty Values  
  
colSums(dataset=='')

## BloodPressure RBS FBS Serum.Insulin BMI   
## 0 0 0 0 0   
## BUN Age Outcome   
## 0 0 0

#### Insights

- Zero Empty Values values in the DublinTest dataset.

#Checking for Duplicate values  
  
duplicated(dataset)

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [433] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [445] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [457] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [469] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [481] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [493] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [505] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [517] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [577] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [589] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [601] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [613] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [625] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [637] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [649] FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [661] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [673] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

#### Insights

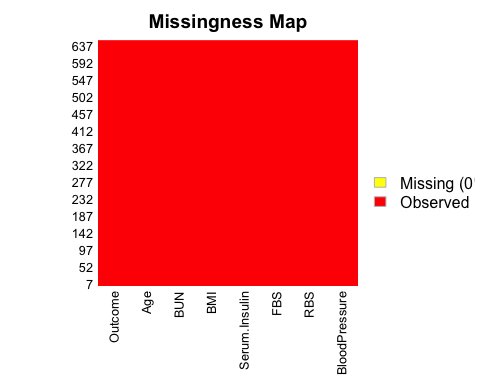
- Clearly, we found 28 duplicated instance in the DublinTest dataset.

dataset = unique(dataset)

#### Insights

- dplyr library is used to remove duplicate in the dataset.  
 - To extract unique elements from the data frames: unique(datadet) is used.

# Missing value imputation  
  
missmap(dataset,col=c("yellow","red"))



#### Insights

- From the above chart it's clear that we don't have any missing values in the DublinTest dataset

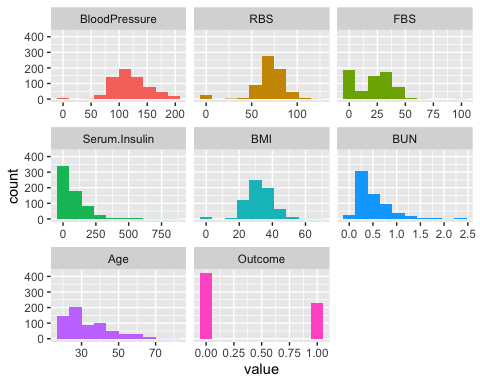
# To find the unique values in each column.  
  
apply(dataset,2,function(x) length(unique(x)))

## BloodPressure RBS FBS Serum.Insulin BMI   
## 133 44 51 171 237   
## BUN Age Outcome   
## 461 50 2

#### Insights

- From the above stastistical table we can find the unique values in each column.   
 - We can convert the Outcome class label column to factor since it's having only 2 distinct value

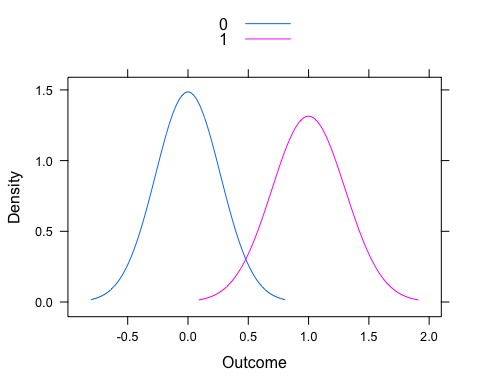
# Distribution of all the variables   
  
plot\_num(dataset)



#### Insights

- Blood Pressure is normally distributed.  
 - Serum Insulin, BUN, and Age is rightly skewed and showing possitive intent.  
 - BMI, RBS, FBS are not normally distributed.

densityplot(~ Outcome, data =dataset,  
 plot.points = FALSE,  
 groups = Outcome,  
 auto.key = TRUE)



#### Insights

- Both the outomes are normally distributed.

# Compute correlation matrix  
library(corrplot)

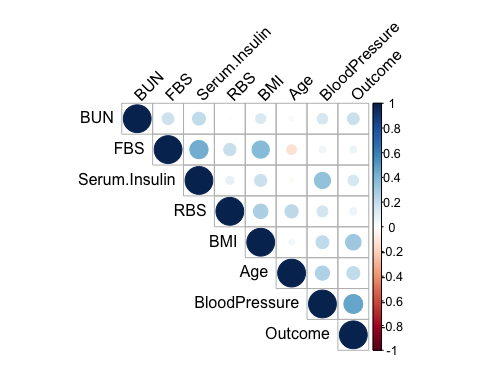
## corrplot 0.84 loaded

library(ggplot2)  
library(reshape2)

##   
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':  
##   
## smiths

res <- cor(dataset)  
corrplot(res, type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)



#### Insights

- BloodPressure and BMI having strong correlation value towards Outcome.  
 - FBS and RBS having less correlation strenght while comparing with other values.

library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

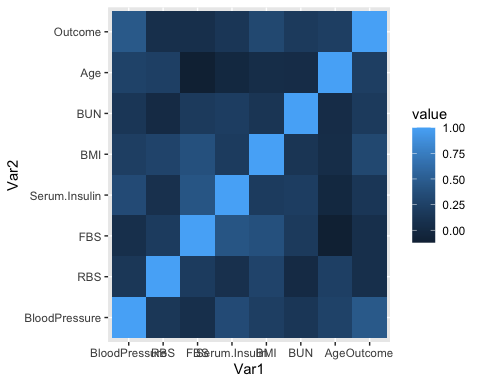
##   
## Attaching package: 'GGally'

## The following object is masked from 'package:funModeling':  
##   
## range01

ggcorr(dataset,label=T)



#Correlation Heat Map  
  
cormat <- round(cor(dataset),2)  
melted\_cormat <- melt(cormat)  
ggplot(data = melted\_cormat, aes(x=Var1, y=Var2, fill=value)) + geom\_tile()



#### Insights

- Most of the columns were seems to be similar in shardings, so multi-collinearity needs to validated.

### Feature Selection

# Checking for multi-collinearity using variance inflation factor

library(car)

## Loading required package: carData  
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
## recode

## The following object is masked from 'package:purrr':  
## some

lrm <- lm(dataset$Outcome~.,data=dataset)  
vif(lrm)

## BloodPressure RBS FBS Serum.Insulin BMI   
## 1.299704 1.167541 1.496838 1.435367 1.285522   
## BUN Age   
## 1.072691 1.161110

#### Insights

- Since all the variance inflation factor are less than 5, we can conclude that there exist no Multi-Collinearity

dataset = subset(dataset, select = -c( RBS, FBS) )  
head(dataset)

## BloodPressure Serum.Insulin BMI BUN Age Outcome  
## 1 117 0 34.1 0.337 38 0  
## 2 109 0 36.0 0.546 60 0  
## 3 158 245 31.6 0.851 28 1  
## 4 88 54 24.8 0.267 22 0  
## 5 92 0 19.9 0.188 28 0  
## 6 122 0 27.6 0.512 45 0

# Converting Class label to Factor  
  
dataset$Outcome=as.factor(dataset$Outcome)

## 3. Split your data set into training and test data

# Data Splitting  
  
set.seed(52)  
  
index <- createDataPartition(dataset$Outcome, p=0.80, list=FALSE)   
train <- dataset[index,] #Train data : 80%  
test<- dataset[-index,] #Test data : 20%

#### Insights

- Here, our dataset is splitted into 8:2 ratio.  
 - 80 % of Training data  
 - 20 % of Testing data

#### 4. Use any two machine learning algorithms to build your model

*Decision tree*  
Decision tree works with both regression and classification problem. Since in dataset target variable has two values 0 and 1 that is its binary classification problem. A Decision tree is nothing but the graphical representation of solutions based on the certain conditions.One disadvantage of using this model is that its generally overfilling.

# Implementing Decision Tree  
  
model\_rpart<-train(Outcome~.,  
 test,  
 method='rpart')  
prediction=predict.train(model\_rpart,  
 test,  
 type="raw")  
confusionMatrix(table(predict(model\_rpart,test),  
 test$Outcome))

## Confusion Matrix and Statistics  
##   
## 0 1  
## 0 81 31  
## 1 3 14  
##   
## Accuracy : 0.7364   
## 95% CI : (0.6516, 0.8101)  
## No Information Rate : 0.6512   
## P-Value [Acc > NIR] : 0.02437   
##   
## Kappa : 0.3219   
##   
## Mcnemar's Test P-Value : 3.649e-06   
##   
## Sensitivity : 0.9643   
## Specificity : 0.3111   
## Pos Pred Value : 0.7232   
## Neg Pred Value : 0.8235   
## Prevalence : 0.6512   
## Detection Rate : 0.6279   
## Detection Prevalence : 0.8682   
## Balanced Accuracy : 0.6377   
##   
## 'Positive' Class : 0   
##

#### Insights

- Accuracy : 0.7364\*

# Using another evaluation error metrics ROC curve.  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

for\_auc=predict(model\_rpart,  
 test,  
 type="prob")  
  
plot(roc(test$Outcome,for\_auc[,2]),  
 main="Decision Tree",  
 col="green")

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases



*SVM*

Support vector machine(SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space (where n is number of features), with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that best differentiates the two classes.

train[["Outcome"]] = factor(train[["Outcome"]])  
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
  
#Building the model  
svm\_Linear <- train(Outcome ~., data = train, method = "svmLinear",  
trControl=trctrl,  
preProcess = c("center", "scale"),  
tuneLength = 10)  
  
#Model Summary  
svm\_Linear

## Support Vector Machines with Linear Kernel   
##   
## 523 samples  
## 5 predictor  
## 2 classes: '0', '1'   
##   
## Pre-processing: centered (5), scaled (5)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 471, 471, 470, 471, 471, 470, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7859223 0.4989604  
##   
## Tuning parameter 'C' was held constant at a value of 1

#### Insights

- Accuracy : 0.7859223

*Random Forest Model*

Random Forest also works with both Regression and Classification problems. It builds a number of the Decision tree and adds them together to get a more effective result. It can also be used for variable importance estimation. Random Forest is a good choice if the model is suffering from the High Variance problem.

# Implementing random Forest.  
  
set.seed(52)  
  
model\_rf<-train(Outcome~.,  
 train,  
 method='rf')  
prediction=predict.train(model\_rf,  
 train,  
 type="raw")  
confusionMatrix(table(predict(model\_rf,train),  
 train$Outcome))

## Confusion Matrix and Statistics  
##   
##   
## 0 1  
## 0 339 0  
## 1 0 184  
##   
## Accuracy : 1   
## 95% CI : (0.993, 1)  
## No Information Rate : 0.6482   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.6482   
## Detection Rate : 0.6482   
## Detection Prevalence : 0.6482   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : 0   
##

#### Insights

- Accuracy : 1

# CONCLUSION

- After doing the neccessary pre-processing work and modeling. Results clearly show that Random Forest performs better than a Decision Tree. Also pre-processing help us in avoiding errors. There are other packages also those help R users for data wrangling and statistical analysis without doing complex coding, but by studying caret package came to know that this one package is enough to building machine learning system.  
 - We got, 73 % of accuracy score from Decision Tree, where as Random Forest provided almost 99 % of accuracy score.  
 - Even through Random forest giving good result, its overfitting.