Artificial Neural Networks (ANN) and Support Vector Machines (SVM)

Appiah Prince pappiah@miners.utep.edu

Due: 12/09/2022

Contents

1 Project Description

We consider the HCV data0 available at UCI machine learning repository: https://archive.ics.uci.edu/ml/data0sets/HCV+data0 Excluding the first column, this 615x13 data0 set contains laboratory values of blood donors and Hepatitis C patients and demographic values like age. The target attribute for classification is Category(blood donors vs. Hepatitis C) (including its progress)('just' Hepatitis C, Fibrosis, Cirrhosis). The main goal of this project is to make medical diagnosis of Hepatitis C based on the results of lab blood work.

2 Data Preparation

2.1 Read in the data

```
dat <- read.csv("hcvdat0.csv", header=TRUE, colClasses=c("NULL", rep(NA, 13)))
dim(dat)
```

```
## [1] 615  13
```

```
head(dat)
```

```
##        Category Age Sex  ALB  ALP  ALT  AST  BIL   CHE CHOL CREA  GGT PROT
## 1 0=Blood Donor  32   m 38.5 52.5  7.7 22.1  7.5  6.93 3.23  106 12.1 69.0
## 2 0=Blood Donor  32   m 38.5 70.3 18.0 24.7  3.9 11.17 4.80   74 15.6 76.5
## 3 0=Blood Donor  32   m 46.9 74.7 36.2 52.6  6.1  8.84 5.20   86 33.2 79.3
## 4 0=Blood Donor  32   m 43.2 52.0 30.6 22.6 18.9  7.33 4.74   80 33.8 75.7
## 5 0=Blood Donor  32   m 39.2 74.1 32.6 24.8  9.6  9.15 4.32   76 29.9 68.7
## 6 0=Blood Donor  32   m 41.6 43.3 18.5 19.7 12.3  9.92 6.05  111 91.0 74.0
```

```
anyNA(dat)
```

```
## [1] TRUE
```

We have 615 observations with 13 variables. Also, we have some missing values in the data.

2.2 Modify the target variable Category

```
dat$Category<-ifelse(dat$Category=="0=Blood Donor" | dat$Category=="0s=suspect Blood Don
```

We modified the target variable Category into binary so that Category = 0 if it falls into either "0=Blood Donor" or "0s=suspect Blood Donor" and 1 if it falls into any other category except being missing, in which case we keep it as is.

2.3 Frequency distribution of Category and checking missing values.

```r
#frequency distribution of Category
table(dat$Category)
```

```
##
##   0   1
## 540  75
```

```r
n <- dim(dat)[1]
table(dat$Category)/n*100
```

```
##
##        0        1
## 87.80488 12.19512
```

```r
# Check if there are missing values in the target variable
anyNA(dat$Category)
```

```
## [1] FALSE
```

We checked the the frequency distribution of the target variable. We found the 540 observation for class labelled *0* and 75 for class labelled *1*. We also found out the percentage of class labelled *1* is 87.8% while 12.2% for the class labelled *0*. Thus, we see that we do have an imbalance classification since the sample size is relatively samll and there is a wide difference in the percentages of the class labelled *1* and *0*. Also, there are no missing observation in the target variable.

2.4 Inspecting for missing values in the predictors

```r
missing_rate <- data.frame()
nr <- NROW(dat)
nc <- NCOL(dat)
Var_name <- variable.names(dat)
for (i in 1:nc) {
  na <- sum(is.na(dat[,i]))
  na_rate <- (na/nr)*100
  result <- list(Variable = Var_name[i],Number_Missing = na, Missing_Rate = na_rate)
  missing_rate <- rbind(missing_rate, result, stringsAsFactors = F)
}
(missing_rate)
```

```
##    Variable Number_Missing Missing_Rate
## 1  Category              0    0.0000000
## 2       Age              0    0.0000000
## 3       Sex              0    0.0000000
## 4       ALB              1    0.1626016
## 5       ALP             18    2.9268293
## 6       ALT              1    0.1626016
## 7       AST              0    0.0000000
## 8       BIL              0    0.0000000
## 9       CHE              0    0.0000000
## 10     CHOL             10    1.6260163
## 11     CREA              0    0.0000000
## 12      GGT              0    0.0000000
## 13     PROT              1    0.1626016
```

There are missing values in the data. For instance, from the table above, the predictor variable ALP has 18 missing observations of about 2.927% of missing rate. We there inpute the missing values.

```
# Imputation of the missing values in the predictors
set.seed(123)
suppressPackageStartupMessages(library(mice))
data_imputed <- mice(dat[,-c(1)], printFlag = F)
```

```
## Warning: Number of logged events: 1
```

```
data <- complete(data_imputed, 1)
data1 <- as.data.frame(data)
data<-cbind("Category"=dat$Category,data1)
rm(data_imputed)

anyNA(data)
```

```
## [1] FALSE
```

We used the package *mice* for imputation of the missing values and checked later after imputation for missing values. We then found out that there are no missing values in the data after imputation.

2.5 Model Matrix

```
Model_matrix <- model.matrix(Category~.,data=data)
head(Model_matrix)
```

```
##   (Intercept) Age Sexm  ALB  ALP  ALT  AST  BIL   CHE CHOL CREA  GGT PROT
## 1           1  32    1 38.5 52.5  7.7 22.1  7.5  6.93 3.23  106 12.1 69.0
## 2           1  32    1 38.5 70.3 18.0 24.7  3.9 11.17 4.80   74 15.6 76.5
## 3           1  32    1 46.9 74.7 36.2 52.6  6.1  8.84 5.20   86 33.2 79.3
## 4           1  32    1 43.2 52.0 30.6 22.6 18.9  7.33 4.74   80 33.8 75.7
## 5           1  32    1 39.2 74.1 32.6 24.8  9.6  9.15 4.32   76 29.9 68.7
## 6           1  32    1 41.6 43.3 18.5 19.7 12.3  9.92 6.05  111 91.0 74.0
```

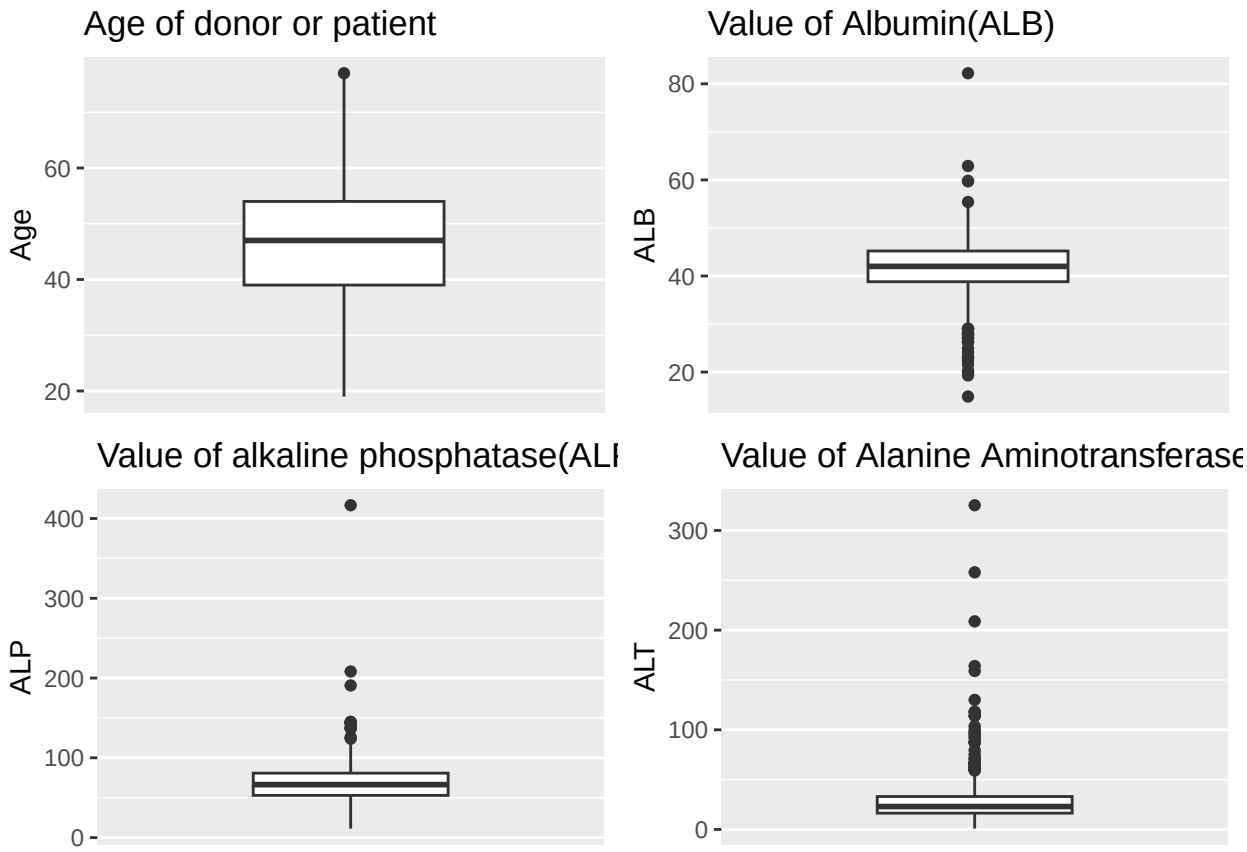
We used model.matrix() to change the data matrix into numeric and found out that dummy variables were automatically created for each categorical predictor.

3 Exploratory data Analysis(EDA)

3.1 View the range and variations of the predictors

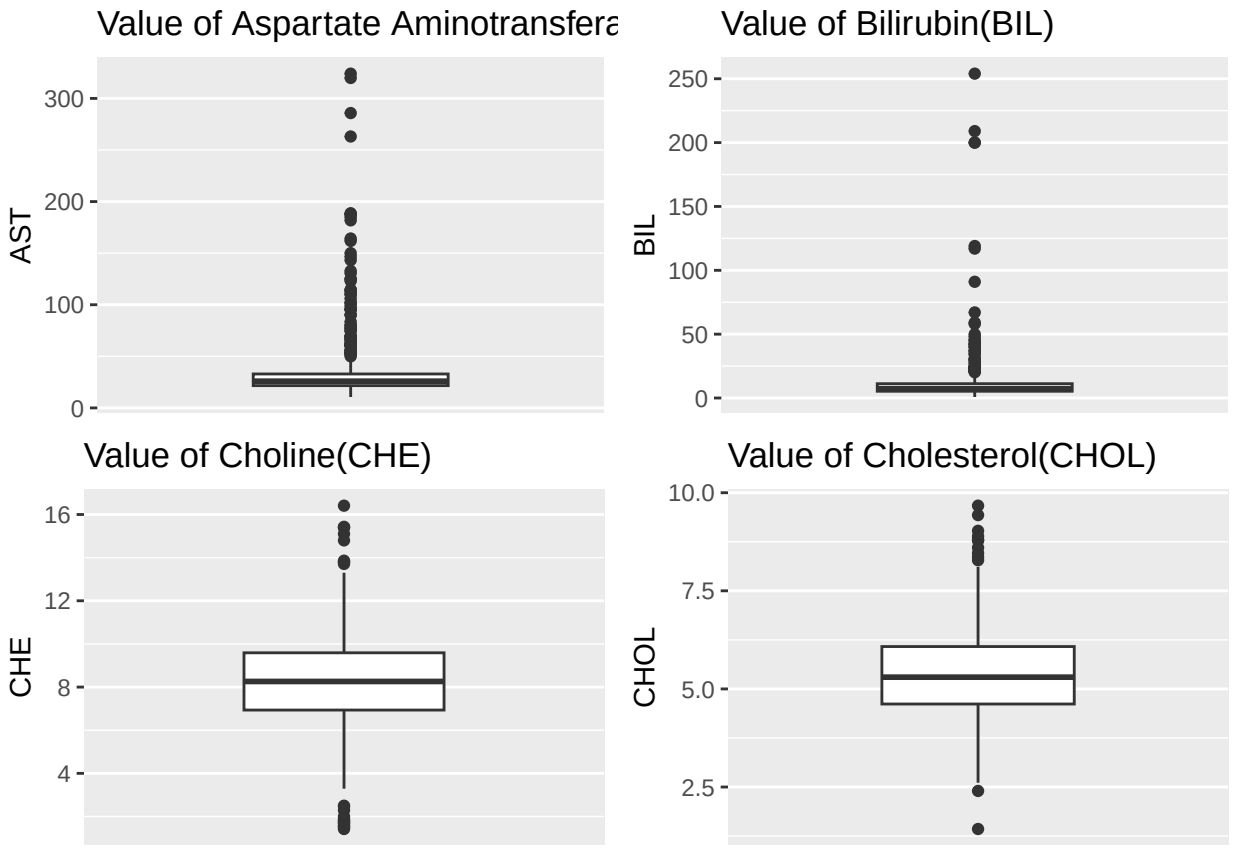
```
library(gridExtra)
library(ggplot2)


b1 <- ggplot() +
  geom_boxplot(aes(y = data$Age)) +
  scale_x_discrete( ) +
  labs(title = "Age of donor or patient",
       y = "Age")

b2 <- ggplot() +
  geom_boxplot(aes(y = data$ALB)) +
  scale_x_discrete( ) +
  labs(title = "Value of Albumin(ALB)",
       y = "ALB")

b3 <- ggplot() +
  geom_boxplot(aes(y = data$ALP)) +
  scale_x_discrete( ) +
  labs(title = "Value of alkaline phosphatase(ALP)",
       y = "ALP")
```

```
b4 <- ggplot() +
  geom_boxplot(aes(y = data$ALT)) +
  scale_x_discrete( ) +
  labs(title = "Value of Alanine Aminotransferase(ALT)",
       y = "ALT")
grid.arrange(b1, b2, b3, b4, nrow=2)
```



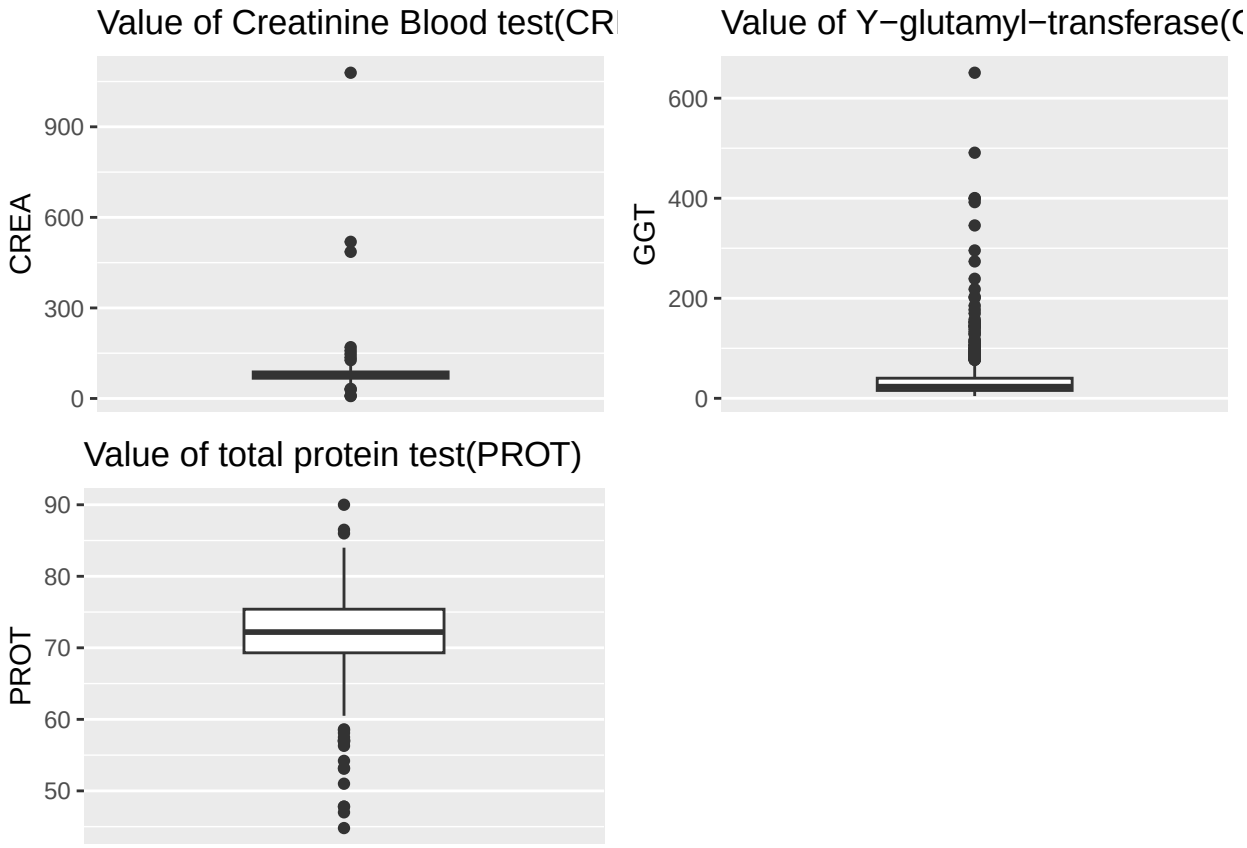

```
b5 <- ggplot() +
  geom_boxplot(aes(y = data$AST)) +
  scale_x_discrete( ) +
  labs(title = "Value of Aspartate Aminotransferase(AST)",
       y = "AST")

b6 <- ggplot() +
  geom_boxplot(aes(y = data$BIL)) +
  scale_x_discrete( ) +
  labs(title = "Value of Bilirubin(BIL)",
       y = "BIL")


b7 <- ggplot() +
```

```
  geom_boxplot(aes(y = data$CHE)) +
  scale_x_discrete( ) +
  labs(title = "Value of Choline(CHE)",
       y = "CHE")

b8 <- ggplot() +
  geom_boxplot(aes(y = data$CHOL)) +
  scale_x_discrete( ) +
  labs(title = "Value of Cholesterol(CHOL)",
       y = "CHOL")
grid.arrange(b5, b6, b7, b8, nrow=2)
```




```
b9 <- ggplot() +
  geom_boxplot(aes(y = data$CREA)) +
  scale_x_discrete( ) +
  labs(title = "Value of Creatinine Blood test(CREA)",
       y = "CREA")

b10 <- ggplot() +
  geom_boxplot(aes(y = data$GGT)) +
  scale_x_discrete( ) +
```

```
  labs(title = "Value of Y-glutamyl-transferase(GGT)",
       y = "GGT")

b11 <- ggplot() +
  geom_boxplot(aes(y = data$PROT)) +
  scale_x_discrete( ) +
  labs(title = "Value of total protein test(PROT)",
       y = "PROT")
grid.arrange(b9, b10, b11,nrow=2)
```



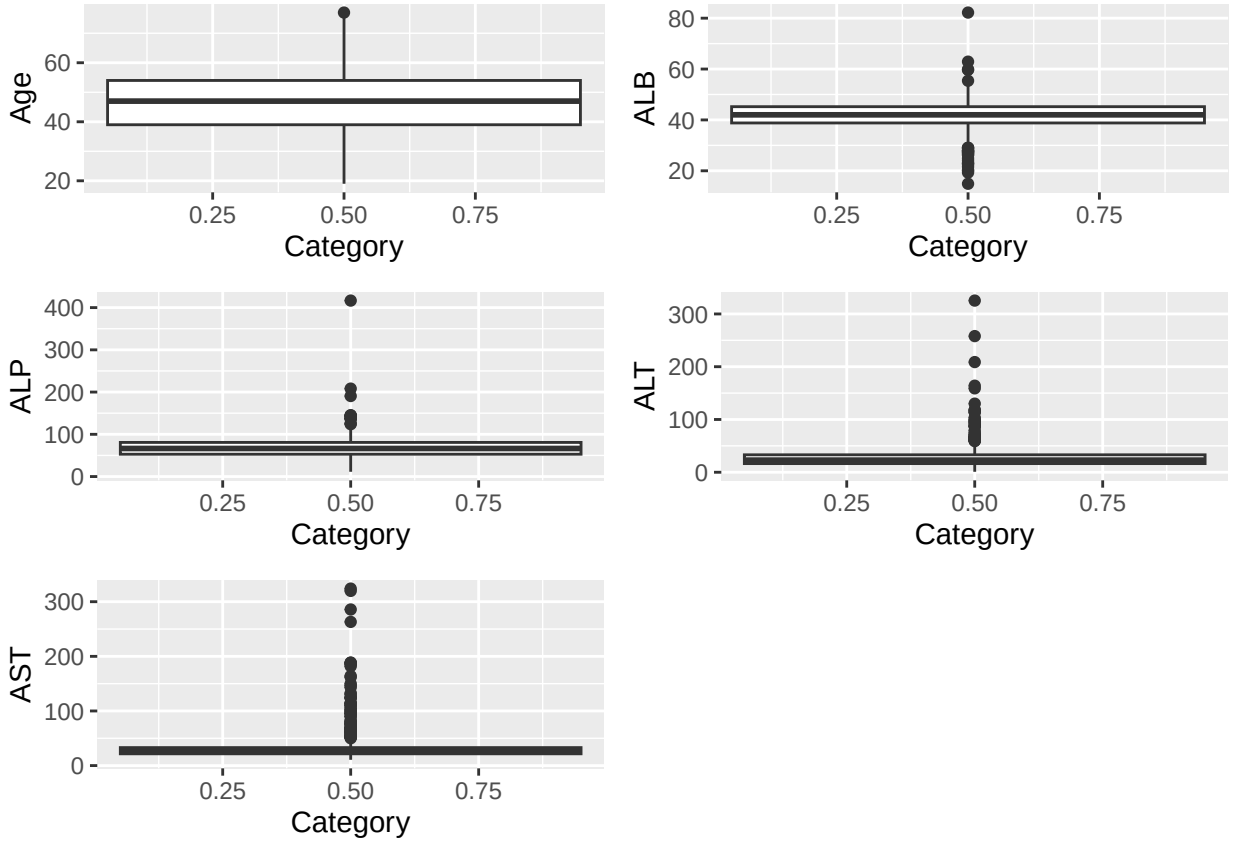
We used boxplot to check for the range and variation of the predictors. We see from the boxplot that there are variations in the predictors. Hence, normalization or scaling is needed for some models.

3.2 Association between the target and predictor

Proportion of target with respect to predictors

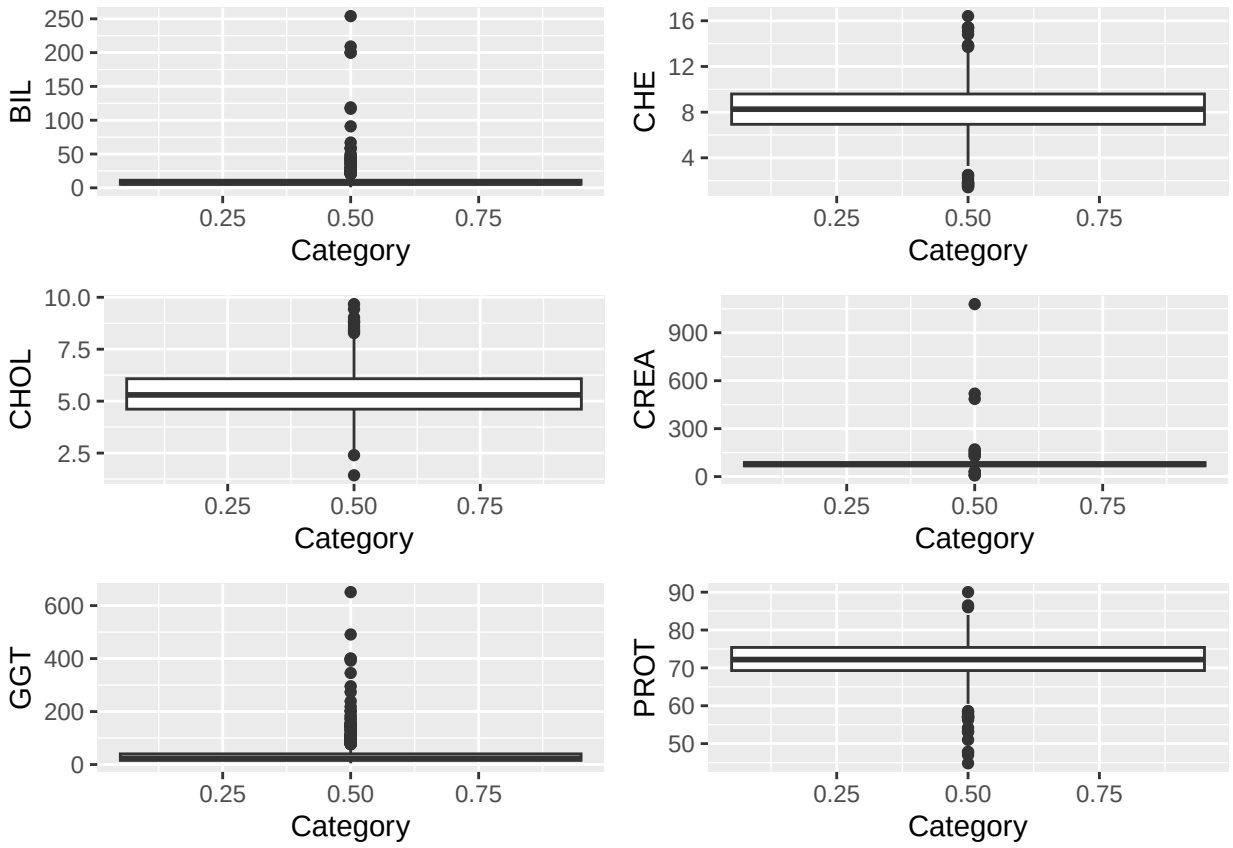
```
ct1 <- ggplot(data, aes(x =Category, y = Age, fill = Category)) +
  geom_boxplot()
```

```
ct2 <- ggplot(data, aes(x =Category, y = ALB, fill = Category)) +
  geom_boxplot()

ct3 <- ggplot(data, aes(x =Category, y = ALP, fill = Category)) +
  geom_boxplot()

ct4 <- ggplot(data, aes(x =Category, y = ALT, fill = Category)) +
  geom_boxplot()

ct5 <- ggplot(data, aes(x =Category, y = AST, fill = Category)) +
  geom_boxplot()

grid.arrange(ct1,ct2,ct3,ct4,ct5,nrow = 3)
```




```
ct6 <- ggplot(data, aes(x =Category, y = BIL, fill = Category)) +
  geom_boxplot()

ct7 <- ggplot(data, aes(x =Category, y = CHE, fill = Category)) +
  geom_boxplot()

ct8 <- ggplot(data, aes(x =Category, y = CHOL, fill = Category)) +
```

```
  geom_boxplot()

ct9 <- ggplot(data, aes(x =Category, y = CREA, fill = Category)) +
  geom_boxplot()

ct10 <- ggplot(data, aes(x =Category, y = GGT, fill = Category)) +
  geom_boxplot()

ct11 <- ggplot(data, aes(x =Category, y = PROT, fill = Category)) +
  geom_boxplot()
grid.arrange(ct6,ct7,ct8,ct9,ct10,ct11,nrow = 3)
```

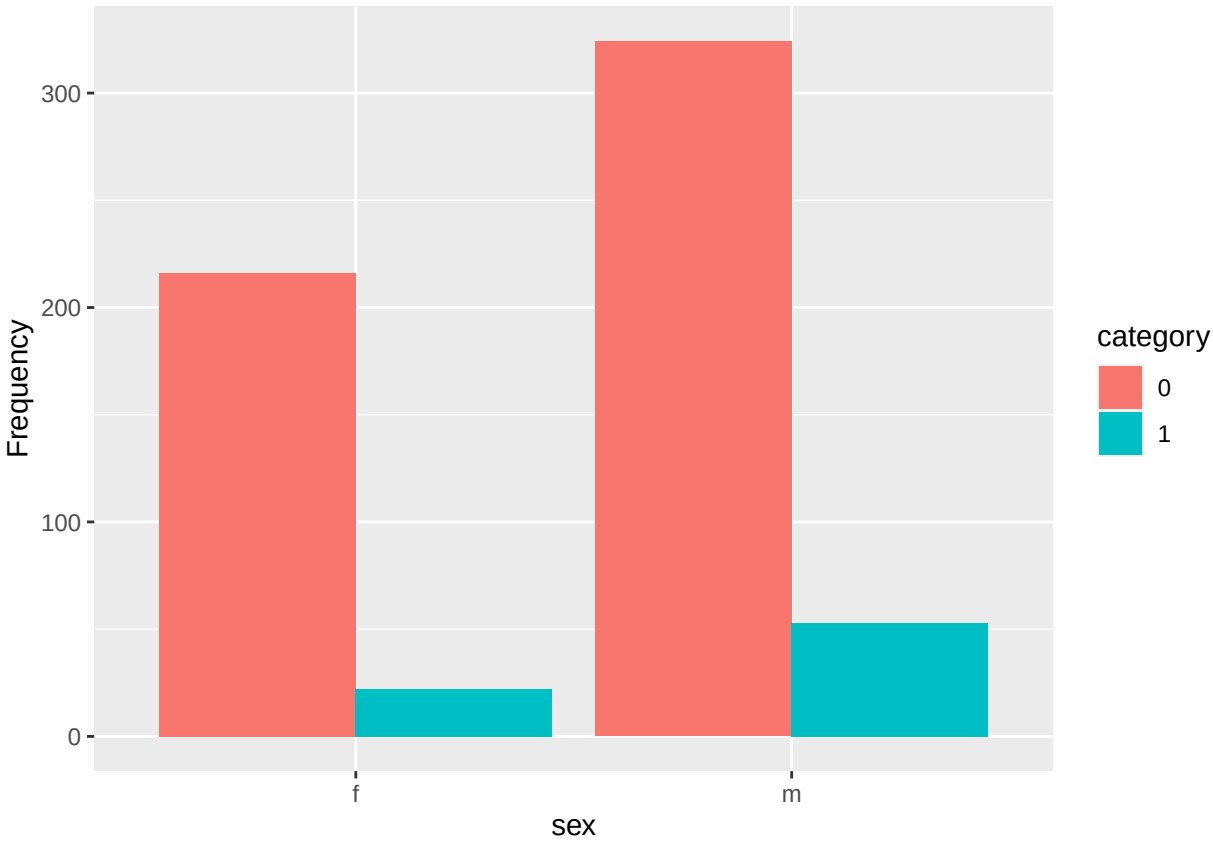


We also checked the proportion/variation of the target with respect to each predictor. From, the each plot we see that there are some variations and wide unequal proportion of the target classes.

Association of the target variable and the categorical predictor variable sex

```
library(ggplot2)
tab <- table(data$Sex, data$Category)
df <- as.data.frame(tab)
colnames(df) <- c("sex", "category", "Frequency")
```

```
ggplot(df, aes(x = sex, y = Frequency, fill = category)) +
  geom_bar(position = "dodge", stat = 'identity')
```

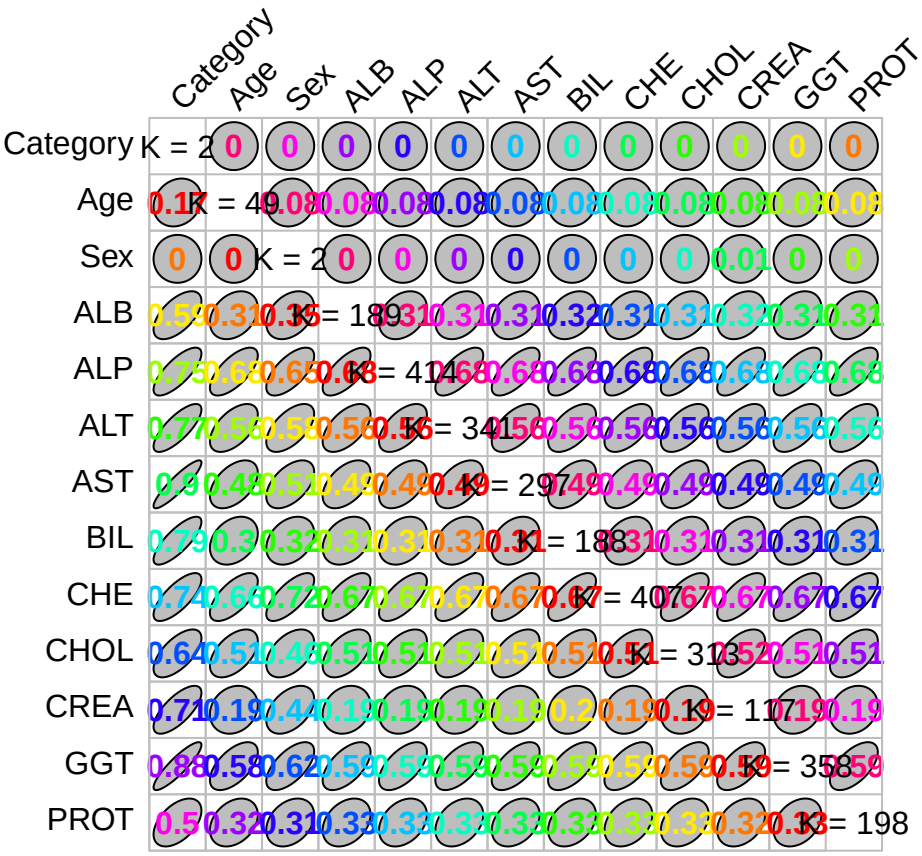


From the plot above, we see that females with target variable labelled *0* has very great percentage/proportion than those labelled *1* Also, males with target variable labelled *0* has greater proportion/percentage than those labelled *1*.

Computing and Visualizing correlation matrix among the variables

```
# Correlation matrix
library(GoodmanKruskal)
data0 <- GKtauDataframe(data)
```

```
# Visualization of the correlation matrix
plot(data0, corColors = "magenta")
```



We see from the values within the the plot of the correlation matrix that there are high association between the variables.

- AST and Category has a correlation coefficient of *0.9*
- GGT and Category has a correlation coefficient of *0.88*
- CHE has relatively high association with all the other variables.

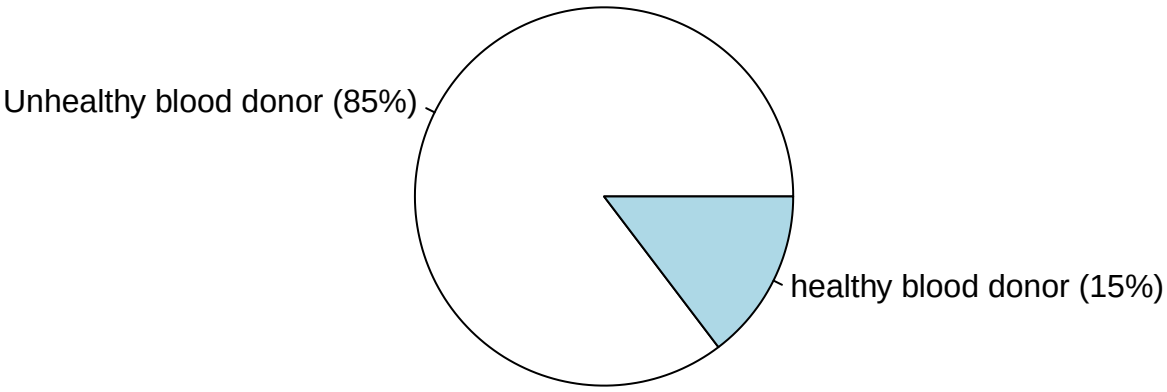
4 Outlier Detection

```
library("e1071")
# Convert Model Matrix to dataframe
df <- data.frame(Model_matrix[,-1])
Category <- data$Category
new_data <- cbind(Category,df)

# Partition dataset all 0(healthy blood donors) to train_set_hbd and
# all 1(Hepatitis C patient) to test_set_hcp
```

```
train_set_hbd <- new_data[new_data$Category==0,]
test_set_hcp <- new_data[new_data$Category==1,]

x <- train_set_hbd[,-1]
p <- NCOL(x)
fit.OneClassSVM <- svm(x, y=NULL, type="one-classification", nu=0.02,
    kernel="radial", gamma=1/p)
summary(fit.OneClassSVM)
```

```
##
## Call:
## svm.default(x = x, y = NULL, type = "one-classification", kernel = "radial",
##     gamma = 1/p, nu = 0.02)
##
##
## Parameters:
##    SVM-Type:  one-classification
##  SVM-Kernel:  radial
##       gamma:  0.08333333
##          nu:  0.02
##
## Number of Support Vectors:  63
##
##
##
##
## Number of Classes: 1
```

```
pred <- predict(fit.OneClassSVM, test_set_hcp[,-1])
tab <- table(pred)
perc_FALSE <- 64*100/75 # FALSE indicates outliers
perc_TRUE <- 11*100/75
TAB <- c(perc_FALSE,perc_TRUE)
percentage_rounded <- round(TAB)
target_des <- c("Unhealthy blood donor", "healthy blood donor")
labels <- paste(target_des, " (", percentage_rounded, "%)", sep = "")
pie(tab, labels)
```



We trained the model on the healthy blood donor dataset and test with the unhealhty blood donor dataset. We found out that 85% proportion of unhealthy blood donors was detected as outliers. Hence, the one-class SVM method was able to detect the Hepatitis C patient as outliers based on what we learned as healthy blood donors.

5 Data Partition

```
set.seed(125)
V <- 10
n <- NROW(data); n0 <- sum(data$Category==0); n1 <- n-n0;
id.fold <- 1:n
id.fold[data$Category==0] <- sample(x=1:V, size=n0, replace=TRUE)
id.fold[data$Category==1] <- sample(x=1:V, size=n1, replace=TRUE)
dim_train <- c()
dim_test <- c()

for (v in 1:V) {
train.v <- data[id.fold!=v, ]
test.v <- data[id.fold==v, ]
dim_train <- c(dim_train, dim(train.v))
dim_test <- c(dim_test, dim(test.v))
}

dim_train
```

```
##  [1] 555  13 553  13 563  13 563  13 554  13 552  13 547  13 548  13 547  13 553
## [20]  13
```

```
dim_test
```

```
##  [1] 60 13 62 13 52 13 52 13 61 13 63 13 68 13 67 13 68 13 62 13
```

6 Predictive Modeling

6.1 Logistic Regression (LASSO) Via V-folds

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
library(verification)
```

```
## Loading required package: fields
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

```
## Loading required package: grid
```

```
## Spam version 2.7-0 (2021-06-25) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following object is masked from 'package:Matrix':
##
##     det
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
## Loading required package: viridis
```

```
## Loading required package: viridisLite
```

```
##
## Try help(fields) to get started.

## Loading required package: boot

## Loading required package: CircStats

## Loading required package: MASS

## Loading required package: dtw

## Loading required package: proxy

##
## Attaching package: 'proxy'

## The following object is masked from 'package:spam':
##
##     as.matrix

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Loaded dtw v1.22-3. See ?dtw for help, citation("dtw") for use in publication.
```

```
set.seed(125)
V <- 10
n <- NROW(data)
n0 <- sum(data$Category==0)
n1 <- n-n0;

missclass.rate = c()
AUC_val=c()
```

```
for (v in 1:V) {
train.v <- data[id.fold!=v, ]; test.v <- data[id.fold==v, ];

formula0 = Category~.
X = model.matrix (as.formula(formula0), data = train.v)
y = factor(train.v$Category)
fit.lasso = glmnet(x=X, y=y, family="binomial", alpha=1,
                   lambda.min = 1e-4, nlambda = 100, standardize=T, thresh =
                     1e-07, maxit=1000)


CV = cv.glmnet(x=X, y=y, family="binomial", alpha = 1,
               lambda.min = 1e-4, nlambda = 200, standardize = T,
               thresh = 1e-07, maxit=1000)

best.lambda = CV$lambda.1se; #best.lambda
fit.best = glmnet(x=X, y=y, family="binomial", alpha = 1,
                  lambda=best.lambda, standardize = T,
                  thresh = 1e-07, maxit=1000)

formula0 = Category ~.
fit.final = glm(formula0, family = "binomial", data = train.v)
summ <- summary(fit.final)
Coefficient_summ <- summ$coefficients

yobs = test.v$Category
X.test = test.v[, -1]
pred.glm = predict(fit.final, newdata = X.test, type="response")

AUC = roc.area(yobs, pred.glm)$A
AUC_val[v] = AUC

pred.rate = ifelse(pred.glm > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate

}
print(paste("Average of AUC:", mean(AUC_val)))
print(paste("Average of Miss:", mean(missclass.rate)))
print(Coefficient_summ)
print(fit.best$beta)

lasso.miss <- mean(missclass.rate)
lasso.AUC <- mean(AUC_val)
```

Here, I used LASSO as penalty function for fitting the logistic regression. We found out that the average AUC is 0.967503644424891 and average mis-classification rate is 0.0450670732872178.

We also found the predictor variables AST,BIL,CHOL,CREA,GGT as statistically significant variables.

Fitting the final best Logistic regression model

```
set.seed(125)
# Here I used the last (when v = 10)result for the train data set
fit.pen.lasso <- glm(factor(Category) ~ AST + BIL + CHOL + CREA + GGT,
                     family = binomial, data=train.v)
sm <- summary(fit.pen.lasso)
sm_coefficients <- sm$coefficients
sm_coefficients
```

```
##                 Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept) -3.579971182 1.073002830 -3.336404 8.486966e-04
## AST          0.054251438 0.009287218  5.841516 5.172781e-09
## BIL          0.072213278 0.020700484  3.488483 4.857706e-04
## CHOL        -0.542065114 0.178415938 -3.038210 2.379879e-03
## CREA         0.007540750 0.003943418  1.912237 5.584580e-02
## GGT          0.008206935 0.004166032  1.969964 4.884244e-02
```

```
confint(fit.pen.lasso, level=0.95)
```

```
##                    2.5 %      97.5 %
## (Intercept) -5.7862682662 -1.50769663
## AST          0.0373251956  0.07390675
## BIL          0.0323873257  0.11421392
## CHOL        -0.9049608678 -0.19672198
## CREA         0.0019933622  0.01882232
## GGT          0.0007841475  0.01717457
```

Again, using the p-values,we found out that the predictor variables AST,BIL,CHOL,CREA,GGT are statistically significant variables at 5% significant level.

The associated odds ratio

```
exp(coef(fit.pen.lasso))
```

```
## (Intercept)         AST         BIL        CHOL        CREA         GGT
##   0.0278765   1.0557500   1.0748846   0.5815461   1.0075693   1.0082407
```

The 95% confidence intervals for the odds ratio

```
exp(confint(fit.pen.lasso, level=0.95))
```

```
##                    2.5 %    97.5 %
## (Intercept) 0.003069415 0.2214194
## AST         1.038030529 1.0767064
## BIL         1.032917503 1.1209919
## CHOL        0.404557716 0.8214190
## CREA        1.001995350 1.0190006
## GGT         1.000784455 1.0173229
```

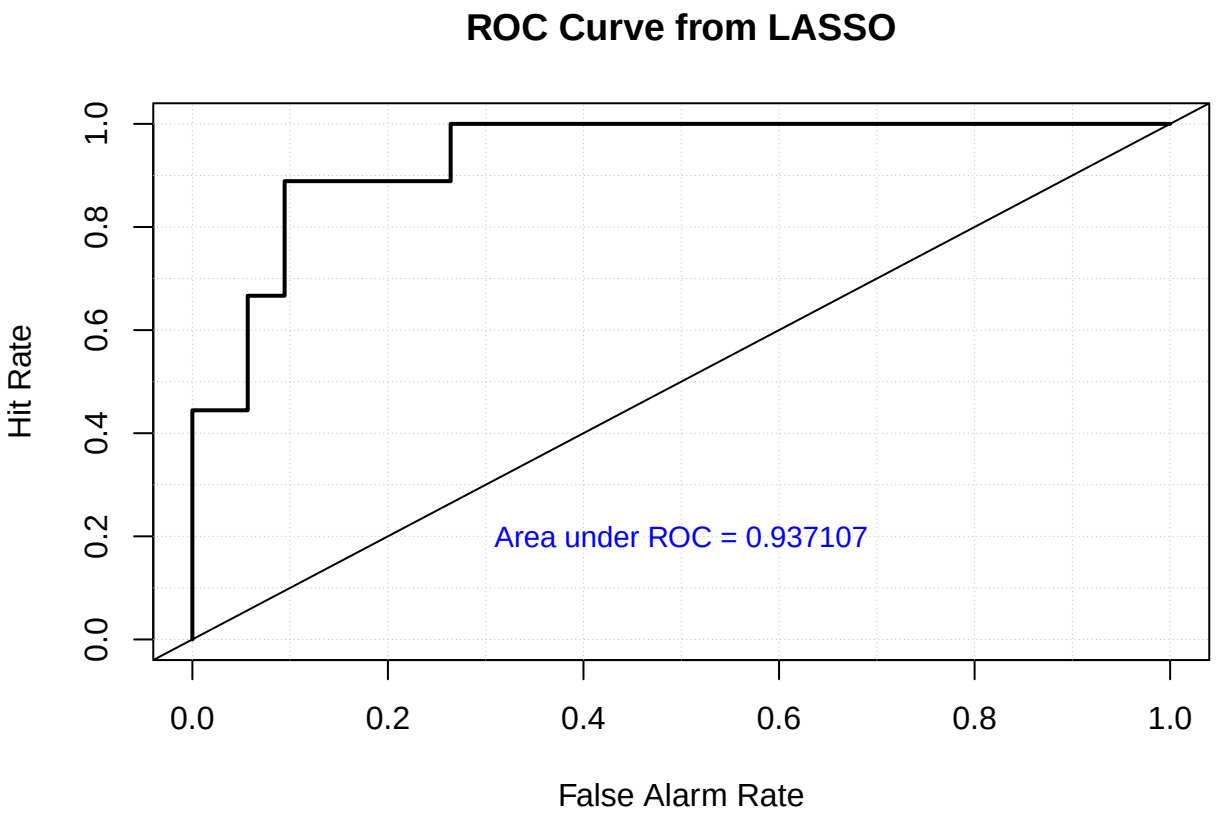
We see that all the selected predictors are signifcant since none of the CI for the odd ratio is contain 1.

ROC Curve

```
set.seed(125)
library(cvAUC)
library(verification)
n <- NROW(test.v)
yobs <- test.v$Category
yhat.lasso <- predict(fit.pen.lasso, newdata=test.v, type="response")
AUC.lasso  <- ci.cvAUC(predictions=yhat.lasso, labels=yobs, folds=1:n, confidence=0.95);
AUC.lasso1 <- AUC.lasso$cvAUC
area.glm <- verify(obs=yobs, pred=yhat.lasso)
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs
```

```
roc.plot(area.glm, plot.thres = NULL, main="ROC Curve from LASSO")
text(x=0.5, y=0.2, paste("Area under ROC =", round(AUC.lasso$cvAUC, digits=6),
    sep=" "), col="blue", cex=0.9)
```



We have 0.937107 as the AUC for the final best model.

```
# Mis-classification rate
missRate.lasso <- mean(yobs != (yhat.lasso > 0.5))
missRate.lasso
```

```
## [1] 0.1290323
```

With a threshold of 0.5, LASSO classifier has a mis-classification rate of 0.129

6.2 Random Forest on V-Folds

```
library(randomForest)
set.seed(125)
for (v in 1:V) {
train.v <- data[id.fold!=v, ]; test.v <- data[id.fold==v, ];

## Fitting model
fit.rf = randomForest(factor(Category) ~.,
                      data=train.v, importance=TRUE, proximity=TRUE,
                      ntree=500)
```

```
yobs = test.v$Category
pred.rf = predict(fit.rf, newdata=test.v[, -c(1)], type="prob")[, 2]
AUC = roc.area(yobs, pred.rf)$A
AUC_val[v] = AUC

pred.rate = ifelse(pred.rf > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate


}
print(paste("Average of AUC:", mean(AUC_val)))
```

```
## [1] "Average of AUC: 0.979888163821854"
```

```
print(paste("Average of Miss:", mean(missclass.rate)))
```

```
## [1] "Average of Miss: 0.0304817049360237"
```

```
rf.miss <- mean(missclass.rate)
rf.AUC <- mean(AUC_val)
```

We found out that the average AUC is 0.979888163821854 and average mis-classification rate is 0.0304817049360237.

Reporting one random forest model with training data through V-folds

```
set.seed(125)
fit.rf
```

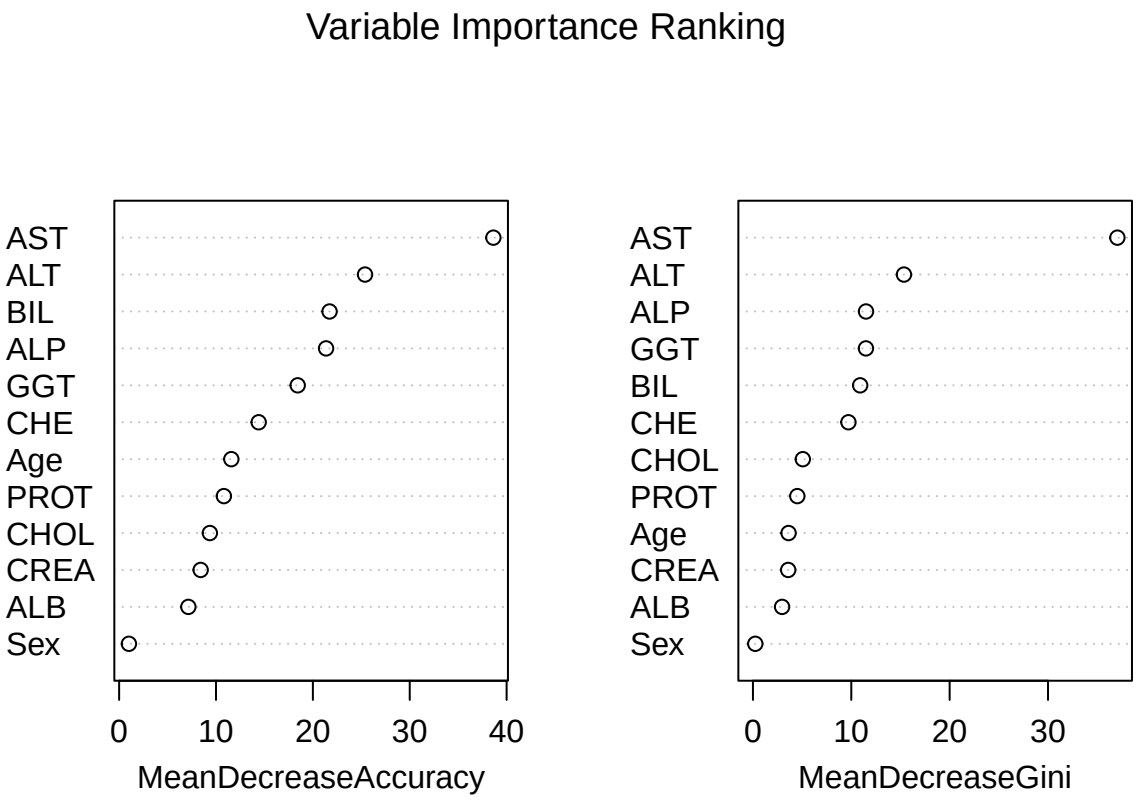
```
##
## Call:
##  randomForest(formula = factor(Category) ~ ., data = train.v,      importance = TRUE,
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 2.35%
## Confusion matrix:
##     0  1 class.error
## 0 484  3 0.006160164
## 1  10 56 0.151515152
```

```
# Here, I used the last fold
pred.rf = predict(fit.rf, newdata=test.v[, -1], type="prob")[, 2]
```

```
# VARIABLE IMPORTANCE RANKING
set.seed(125)
round(importance(fit.rf), 2)
```

```
##           0     1 MeanDecreaseAccuracy MeanDecreaseGini
## Age   11.46  6.15                11.58             3.62
## Sex    1.49 -0.60                 1.01             0.24
## ALB    6.10  3.52                 7.15             2.96
## ALP   17.17 17.81                21.36            11.50
## ALT   23.08 17.76                25.38            15.35
## AST   28.54 38.41                38.63            37.05
## BIL    8.09 22.24                21.72            10.90
## CHE   12.42  9.47                14.41             9.71
## CHOL   7.49  5.52                 9.37             5.07
## CREA   7.13  6.00                 8.42             3.58
## GGT   12.90 14.56                18.43            11.49
## PROT  11.69  0.19                10.81             4.51
```

```
varImpPlot(fit.rf, main="Variable Importance Ranking")
```



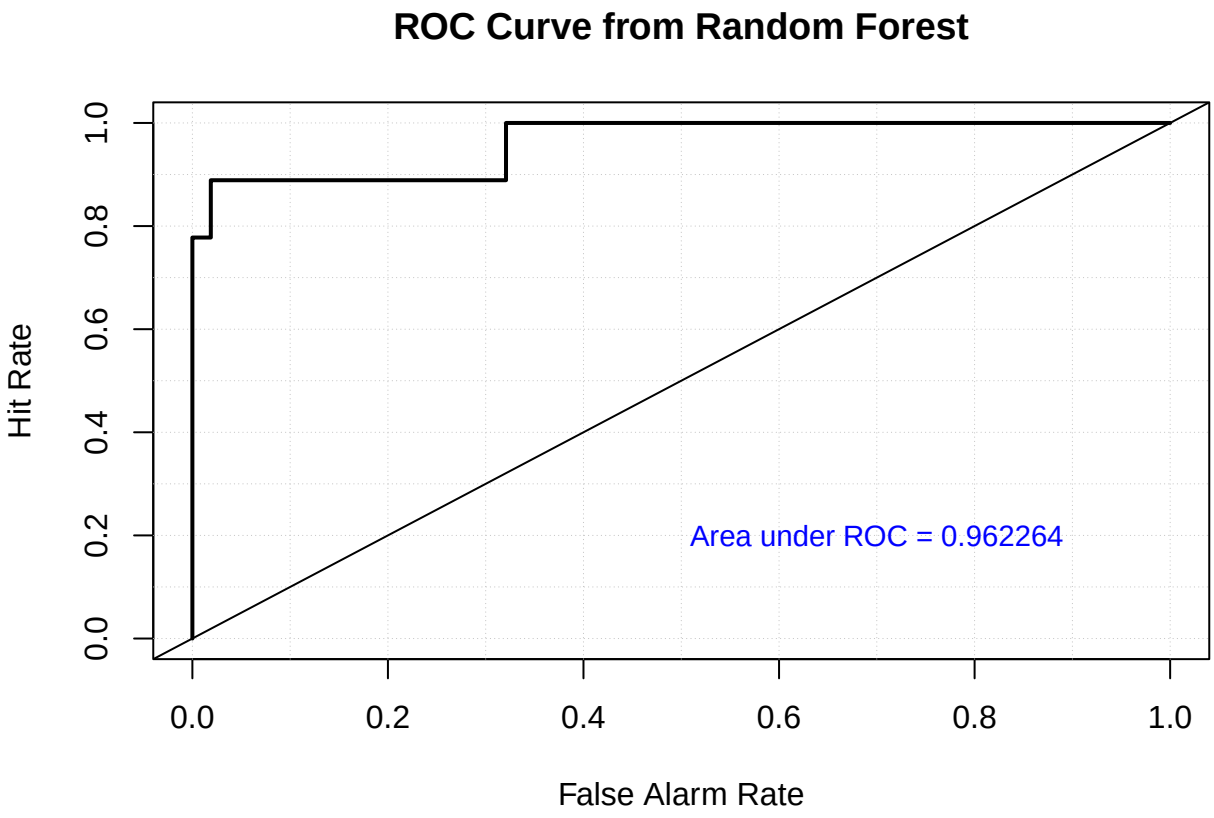
Using the MeanDecreaseAccuracy, the top three variables according to the variable importance ranking for random forest are AST, ALT and BIL.The least significant variable is sex.

AUC Curve for Random forest

```
AUC.RF <- roc.area(obs=yobs, pred=pred.rf)$A
area.rf <- verify(obs=yobs, pred=pred.rf)
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs
```

```
roc.plot(area.rf, plot.thres = NULL, col="red", main="ROC Curve from Random Forest")
text(x=0.7, y=0.2, paste("Area under ROC =", round(AUC.RF, digits=6),
    sep=" "), col="blue", cex=0.9)
```



The AUC is 0.962264

6.3 Multivariate Adaptive Regression Splines through V-folds

```
set.seed(125)
library("earth")
```

```
library(ggplot2)    # plotting
library(caret)      # automating the tuning process
library(vip)        # variable importance
library(pdp)        # variable relationships

for (v in 1:V) {
train.v <- data[id.fold!=v, ]
test.v <- data[id.fold==v, ]

fit.mars <- earth(Category ~ .,  data = train.v, degree=3,
    glm=list(family=binomial(link = "logit")))

yobs = test.v$Category
yhat.mars <- predict(fit.mars, newdata=test.v[, -1], type="response")
AUC = roc.area(yobs, yhat.mars)$A
AUC_val[v] = AUC


pred.rate = ifelse(pred.rf > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate



}
print(paste("Average of AUC:", mean(AUC_val)))
```

```
## [1] "Average of AUC: 0.926639944299163"
```

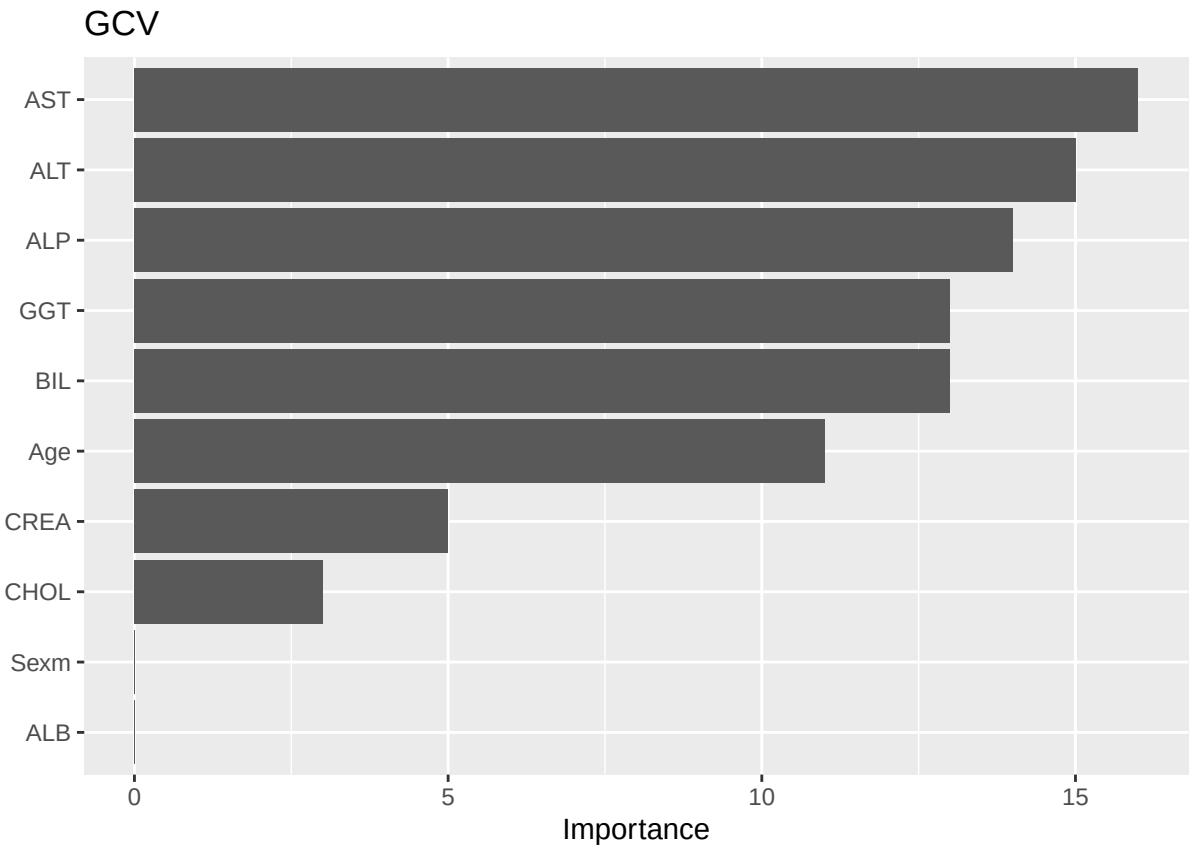
```
print(paste("Average of Miss:", mean(missclass.rate)))
```

```
## [1] "Average of Miss: 0.122719678466797"
```

```
MARS.miss <- mean(missclass.rate)
MARS.AUC <- mean(AUC_val)
```

We found out that the average AUC is 0.926639944299163 and average mis-classification rate is 0.122719678466797.

```
library(vip)
# VARIABLE IMPORTANCE PLOT
vip(fit.mars, num_features = 10) + ggtitle("GCV")
```



We see from the graph that the top three important variables are AST,ALT and ALP.

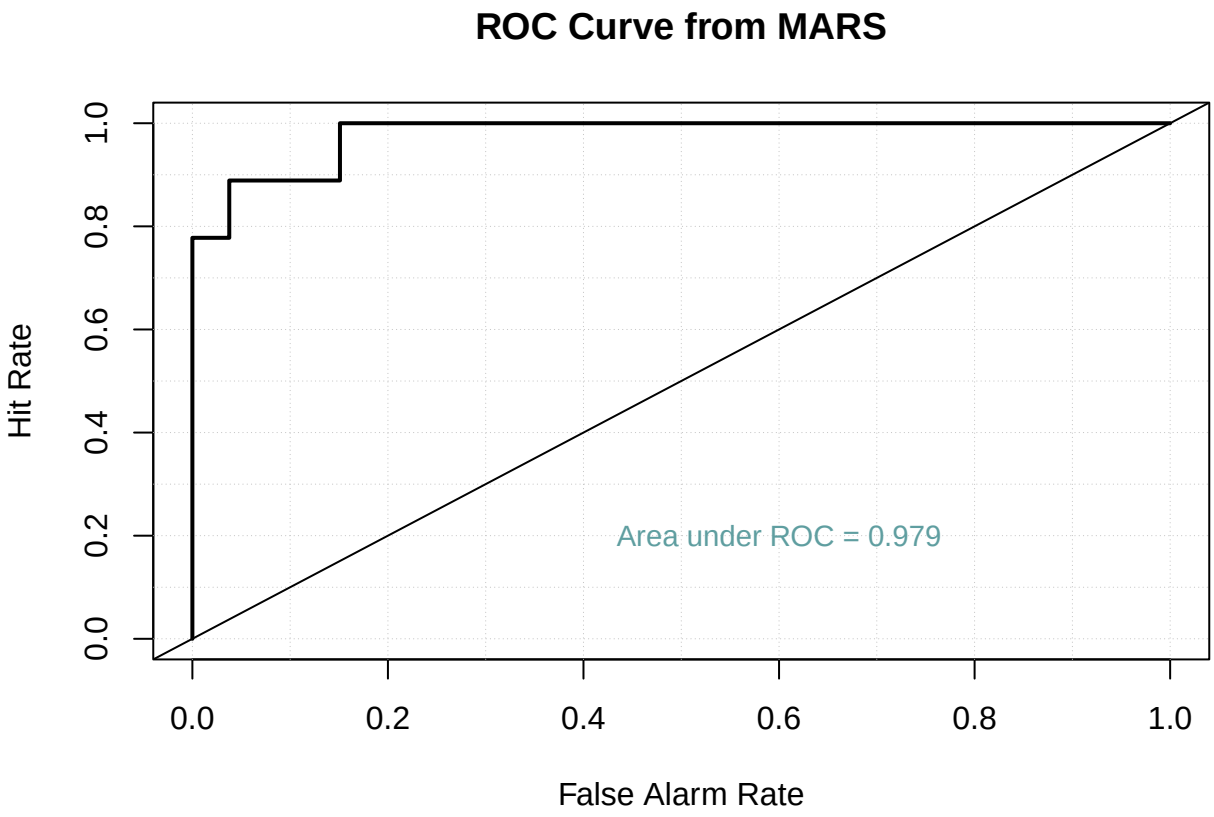
```
# PREDICTION
AUC.MARS <- ci.cvAUC(predictions=yhat.mars, labels=yobs,
                     folds=1:length(yhat.mars), confidence=0.95); AUC.MARS
```

```
## $cvAUC
## [1] 0.9790356
##
## $se
## [1] 0.01741457
##
## $ci
## [1] 0.9449037 1.0000000
##
## $confidence
## [1] 0.95
```

```
AUC.Mar<-AUC.MARS$cvAUC
auc.ci <- round(AUC.MARS$ci, digits=6)
library(verification)
area.mars <- verify(obs=yobs, pred=yhat.mars)
```

```
## If baseline is not included, baseline values  will be calculated from the  sample obs
```

```
roc.plot(area.mars, plot.thres = NULL, main="ROC Curve from MARS")
text(x=0.6, y=0.2, paste("Area under ROC =",
                         round(AUC.MARS$cvAUC, digits=4),
    sep=" "), col="cadetblue", cex=0.9)
```



The AUC is 0.979

```
# Missclassification rate
missRate.Mars <- mean(yobs != (yhat.mars>0.5))
missRate.Mars
```

```
## [1] 0.06451613
```

MARS classifier has a missclassification rate of 0.0645 with a threshold of 0.5.

6.4 Support Vector Machines (SVM)

6.4.1 Linear SVM

```r
# Using  C value in Linear Kernel Classifier
library(caret)
set.seed(125)
data11 = as.data.frame(model.matrix(Category~. -1, data = data))
newdata = data.frame(Category = data$Category, data11)
trctrl <- trainControl(method = "repeatedcv", number=10, repeats=3)
V <- 10
index.cv <- sample(1:V, size=NROW(newdata), replace=TRUE)

for (v in 1:V){
    train_d.v <- newdata[index.cv!=v, ]
    valid_d.v <- newdata[index.cv==v, ]
    yobs <- valid_d.v[, 1]
    X.train <- train_d.v[, -1]; X.test <- valid_d.v[, -1]
    scale.train <- scale(X.train, center=TRUE, scale = TRUE)
    train_scaled.v <- data.frame(Category=train_d.v[, 1] , scale.train)
    valid_d.v[, 2:14] <- as.data.frame(scale(X.test,
                          center=attributes(scale.train)$`scaled:center`,
                          scale=attributes(scale.train)$`scaled:scale`))

grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))
svm_Linear_Grid <- train(factor(Category) ~., data = train_scaled.v,
                          method = "svmLinear",
                     trControl=trctrl,
                     tuneGrid = grid,
                     tuneLength = 10)

test_pred.svm2 <- predict(svm_Linear_Grid, newdata =valid_d.v[,-1],type="raw")
test_pred.svm2 <- as.numeric(as.character(test_pred.svm2))

AUC = roc.area(yobs, test_pred.svm2)$A
AUC_val[v] = AUC

pred.rate = ifelse(test_pred.svm2 > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate

}
print(c("Average AUC:", mean(AUC_val)))
```

```
## [1] "Average AUC:"      "0.855577810927894"
```

```
print(c("Average Misclassification rate:", mean(missclass.rate)))
```

```
## [1] "Average Misclassification rate:" "0.0483341598089986"
```

```
SVM2.miss <- mean(missclass.rate)
SVM2.AUC <- mean(AUC_val)
```

We found out that the average AUC is 0.855577810927894 and average mis-classification rate is 0.0483341598089986.

6.4.2 Non Linear SVM

```
# Using Radial kernel
set.seed(125)
library(caret)

data11 = as.data.frame(model.matrix(Category~. -1, data = data))
newdata = data.frame(Category = data$Category, data11)

V <- 10
index.cv <- sample(1:V, size=NROW(newdata), replace=TRUE)

for (v in 1:V){
    train_d.v <- newdata[index.cv!=v, ]
    valid_d.v <- newdata[index.cv==v, ]
    yobs <- valid_d.v[, 1]
    X.train <- train_d.v[, -1]; X.test <- valid_d.v[, -1]
    scale.train <- scale(X.train, center=TRUE, scale = TRUE)
    train_scaled.v <- data.frame(Category=train_d.v[, 1] , scale.train)
    valid_d.v[, 2:14] <- as.data.frame(scale(X.test,
                          center=attributes(scale.train)$`scaled:center`,
                          scale=attributes(scale.train)$`scaled:scale`))

trctrl <- trainControl(method = "repeatedcv", number=10, repeats = 3)

svm_Radial <- train(factor(Category) ~., data = train_scaled.v,
                    method = "svmRadial",
                  trControl=trctrl,
                  tuneLength = 10)

# PREDICTION
```

```
test_pred.svm3 <- predict(svm_Radial, newdata = valid_d.v[,-1],type="raw")
test_pred.svm3 <- as.numeric(as.character(test_pred.svm3))


AUC = roc.area(yobs, test_pred.svm3)$A
AUC_val[v] = AUC


pred.rate = ifelse(test_pred.svm3 > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate

}
print(c("Average AUC:", mean(AUC_val)))
```

```
## [1] "Average AUC:"      "0.865223467659276"
```

```
print(c("Average Misclassification rate:", mean(missclass.rate)))
```

```
## [1] "Average Misclassification rate:" "0.0485994168280949"
```

```
SVM3.miss <- mean(missclass.rate)
SVM3.AUC <- mean(AUC_val)
```

We found out that the average AUC is 865223467659276 and average mis-classification rate is 0.0485994168280949.

6.5 Artificial Neural Networks (ANN)

```
# DATA PREPARATION - NEED TO DEAL WITH CATEGORICAL PREDICTORS
X.dat <- as.data.frame(model.matrix(Category~.-1, data=data))
dat1 <- data.frame(cbind( Category=data$Category,X.dat))
```

6.5.1 Model 1: 1 hidden layer with 3 hidden units

```
#Partitioning of Data
set.seed(125)
library(neuralnet);
```

```
V <- 10
index.cv <- sample(1:V, size=NROW(data), replace=TRUE)

for (v in 1:V){
    train_d.v <- dat1[index.cv!=v, ]
    valid_d.v <- dat1[index.cv==v, ]

    X.train <- train_d.v[, -1]; X.test <- valid_d.v[, -1]
    scale.train <- scale(X.train, center=TRUE, scale = TRUE)
    train_scaled.v <- data.frame(Category=train_d.v[, 1] , scale.train)
    valid_d.v[, 2:14] <- as.data.frame(scale(X.test,
                          center=attributes(scale.train)$`scaled:center`,
                          scale=attributes(scale.train)$`scaled:scale`))

    yobs <- valid_d.v[, 1]

# Using  1 hidden layer, 3 units
options(digits=3)
net1 = neuralnet(Category~., data=train_scaled.v, hidden=3, rep = 1,
        threshold = 0.05, stepmax = 1e+05, algorithm = "rprop+",
        err.fct = "ce", act.fct = "logistic",
        linear.output=FALSE, likelihood=TRUE)

# PLOT THE MODEL
pred.11 = as.vector(neuralnet::compute(net1,
                                       covariate=valid_d.v[,-1])$net.result)
AUC = roc.area(yobs, pred.11)$A
AUC_val[v] = AUC


pred.rate = ifelse(pred.11 > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate
}

print(paste("Average of AUC:", mean(AUC_val)))
```

```
## [1] "Average of AUC: 0.960630654131686"
```

```
print(paste("Average of Miss:", mean(missclass.rate)))
```

```
## [1] "Average of Miss: 0.0485530267164936"
```

```
ANN1.miss <- mean(missclass.rate)
ANN1.AUC <- mean(AUC_val)
```

We found out that the average AUC is 960630654131686 and average mis-classification rate is 0.0485530267164936

6.5.2 Model 2: 2 hidden layers with 2 units in layer 1 and 3 units in layer 2

```
set.seed(125)
library(neuralnet);

V <- 10
index.cv <- sample(1:V, size=NROW(dat1), replace=TRUE)

for (v in 1:V){
    train_d.v <- dat1[index.cv!=v, ]
    valid_d.v <- dat1[index.cv==v, ]

    X.train <- train_d.v[, -1]; X.test <- valid_d.v[, -1]
    scale.train <- scale(X.train, center=TRUE, scale = TRUE)
    train_scaled.v <- data.frame(Category=train_d.v[, 1] , scale.train)
    valid_d.v[, 2:14] <- as.data.frame(scale(X.test,
                          center=attributes(scale.train)$`scaled:center`,
                          scale=attributes(scale.train)$`scaled:scale`))

yobs <- valid_d.v[, 1]

options(digits=3)
net2 = neuralnet(Category~., data=train_scaled.v, hidden=c(2,3), rep = 1,
        threshold = 0.05, stepmax = 1e+05, algorithm = "rprop+",
        err.fct = "ce", act.fct = "logistic",
        linear.output=FALSE, likelihood=TRUE)



pred.11 = as.vector(neuralnet::compute(net2,
                                       covariate=valid_d.v[,-1])$net.result)
AUC = roc.area(yobs, pred.11)$A
AUC_val[v] = AUC


pred.rate = ifelse(pred.11 > 0.5, 1, 0)
```

```
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate

}

print(paste("Average of AUC:", mean(AUC_val)))
```

```
## [1] "Average of AUC: 0.930436745042663"
```

```
print(paste("Average of Miss:", mean(missclass.rate)))
```

```
## [1] "Average of Miss: 0.0592647842113311"
```

```
ANN2.miss <- mean(missclass.rate)
ANN2.AUC <- mean(AUC_val)
```

We found out that the average AUC is 0.930436745042663 and average mis-classification rate is 0.0592647842113311.

7 Model Comparison

```
Missclassification_Rate <-c(lasso.miss,rf.miss,MARS.miss,
                            SVM2.miss,SVM3.miss,ANN1.miss,
                            ANN2.miss)
AUC <- c(lasso.AUC,rf.AUC,MARS.AUC,SVM2.AUC,SVM3.AUC,
         ANN1.AUC,ANN2.AUC)
Measures <- data.frame("Classifier Method"= c("LASSO Regularized Regression",
                                              "ANN-c(2:2,3)"),
             "MissClassification Rate"= Missclassification_Rate, "AUC"=AUC)
knitr::kable(Measures, align = "lcc")
```

Classifier.Method	MissClassification.Rate	AUC
LASSO Regularized Regression	0.045	0.968
RF	0.030	0.980
MARS	0.123	0.927
SVM-Linear	0.048	0.856
SVM-Nonlinear	0.049	0.865
ANN-c(1,3)	0.049	0.961
ANN-c(2:2,3)	0.059	0.930

Final Remarks

We observe that Random Forest has the highest AUC and smallest missclassification rate among the other classifier techniques.Hence, we conclude that the Random Forest technique is the best technique for this particular dataset and problem. Random forest has high performance and accurate performance in modelling and no feature scaling is required. However,it has high prediction time and it is computationally costly.

The second best technique is the Logisitc regression(Lasso classifier). In this technique, results are easy to interpret and the response variable need not be normal. However, it requires more data to attain stability and its mostly effective on linearly separable.

Among the non baseline classifer techniques, ANN is the best model followed by main MARS. ANN has the ability to learn and model non-linear and complex relationships.However, in ANN there is no specific rule for determining the structure of artificial neural networks.The appropriate network structure is achieved through experience and trial and error. It is also difficult to learn and understand. SVM performed poorer as compared to the other techniques.SVM does not perform very well when the data set has more noise and in cases where the number of features for each data point exceeds the number of training data samples. MARS is robust to outliers, works well with large number of predictor variables and able to detect interactions between variables. The cons of MARS are more difficult to understand and interpret and susceptible to overfitting.

It is important to note that on average all the classifier techniques did well. The least AUC was about 86% which is reasonably good.

8 References

1. https://medium.com/@gokul.elumalai05/pros-and-cons-of-common-machine-learning-algorithms-45e05423264f

2. https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107