Final Project

Isaiah Thompson Ocansey

2022-12-09

DATA PREPARATION

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

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

```
##        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
```

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

The data set has 615 observations and 13 variables with some missing values

(1a) Modify 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.

```
dat$Category<-ifelse(dat$Category=="0=Blood Donor" | dat$Category=="0s=suspect Blood Donor", 0,1)
head(dat)
```

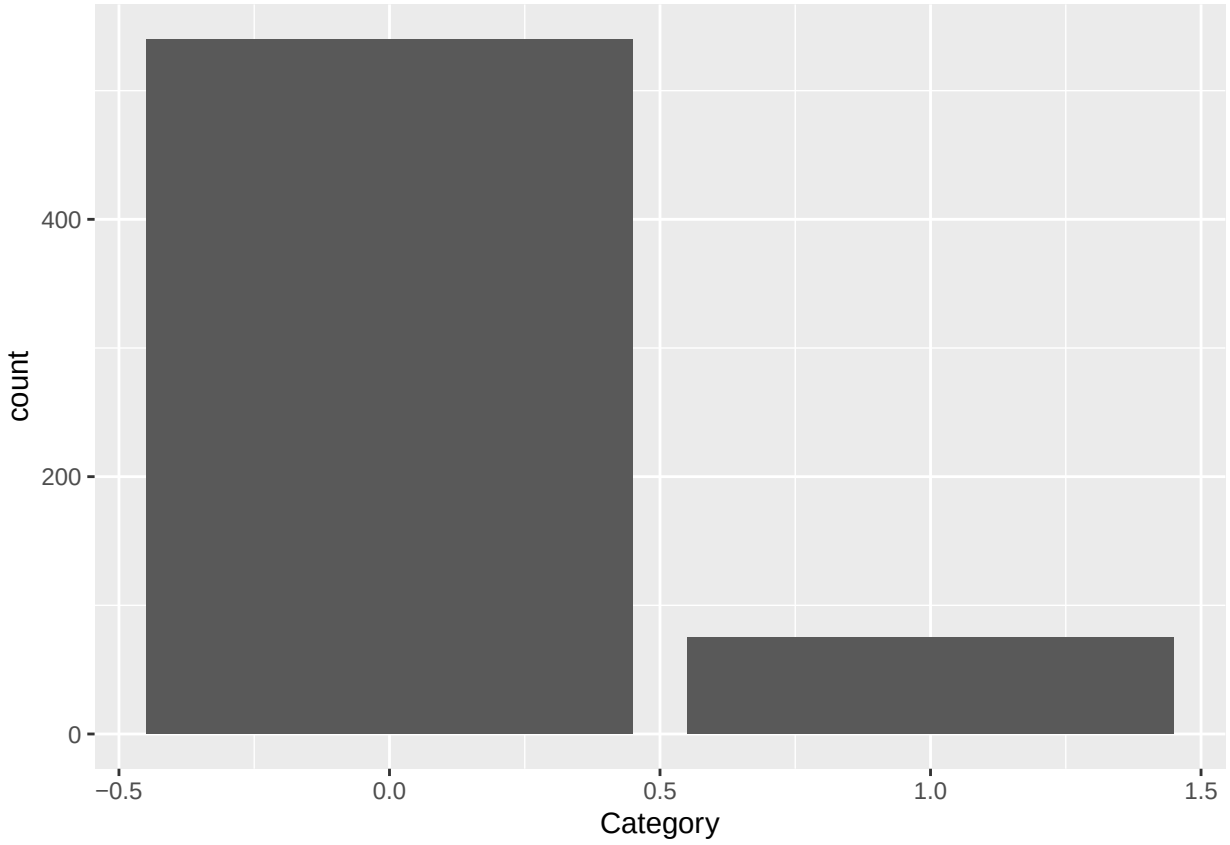
```
##   Category Age Sex  ALB  ALP  ALT  AST  BIL   CHE CHOL CREA  GGT PROT
## 1        0  32   m 38.5 52.5  7.7 22.1  7.5  6.93 3.23  106 12.1 69.0
## 2        0  32   m 38.5 70.3 18.0 24.7  3.9 11.17 4.80   74 15.6 76.5
## 3        0  32   m 46.9 74.7 36.2 52.6  6.1  8.84 5.20   86 33.2 79.3
## 4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

(1b) Obtain the frequency distribution of Category. Do we have an imbalanced classification problem? Any missing value in Category? If so, let's remove these observations or rows

```
tab<-table(dat$Category)
as.data.frame(tab)
```

```
##   Var1 Freq
## 1    0  540
## 2    1   75
```

```
library(ggplot2)
dist<-ggplot(dat,aes(Category)) + geom_bar()
dist
```




```
library(questionr)
freq(dat$Category, total=T)
```

```
##         n     %  val%
## 0     540  87.8  87.8
## 1      75  12.2  12.2
## Total 615 100.0 100.0
```

```
## n % val%
## 0 540 87.8 87.8
## 1 75 12.2 12.2
## Total 615 100.0 100.0
```

From the plot above, the classification problem seems imbalanced and needs scaling for the purpose of modeling. Also, it can be observed that about 87.8% of the total observations falls under the healthy donors and about 12.2% are unhealthy donors.

There are no missing values in the variable category

(1c) Inspect for the missing values in the predictors. Impute the missing values if any. Note that you are not supposed to use information in the target variable when performing the imputation.

‘

```
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
```

The above table shows the number of missing values with corresponding missing rates, It can be observed from the table that ALP and CHOL has the highest number of missing values

```
#Missing  value imputation
set.seed(123)
suppressPackageStartupMessages(library(mice))
imputed_dat <- mice(dat[,-1], printFlag = F)
data <- complete(imputed_dat, 1)
data0 <- as.data.frame(data)
data<-cbind("Category"=dat$Category,data0)
rm(imputed_dat)
sum(is.na(data))
```

```
## [1] 0
```

It can be observed that we no longer have missing values after the imputaion with the mice package

(d) Use model.matrix() to change the data matrix into numeric. Dummy variables will be automatically created for each categorical predictor.

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

```
##   (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
```

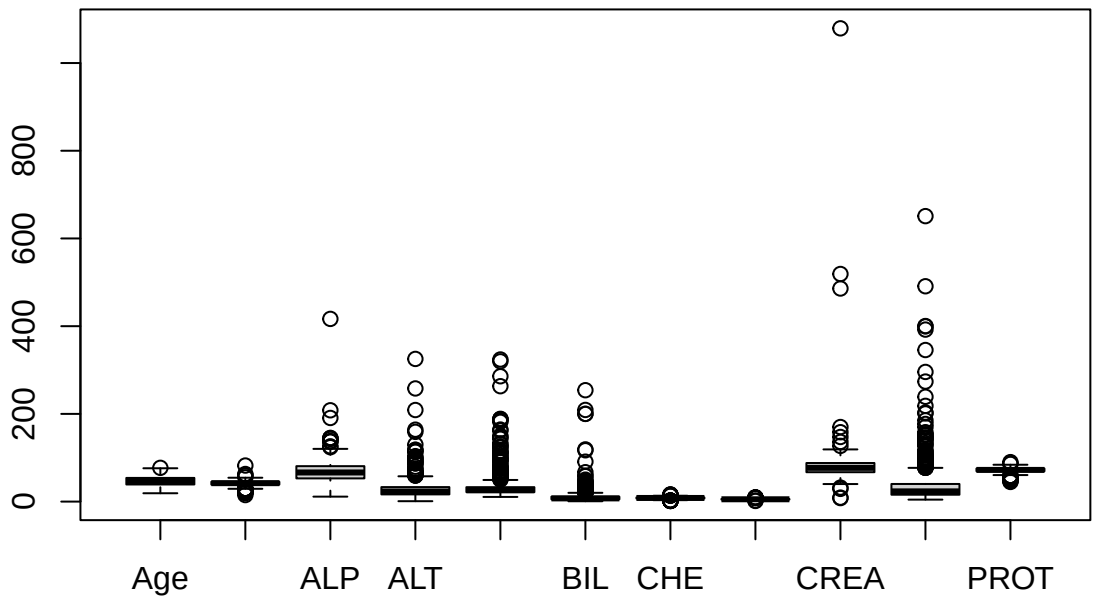
DATA EXPLORATION

(2a) View the range and variations of the predictors. Is there a need to normalize them?

```
str(data)
```

```
## 'data.frame':    615 obs. of  13 variables:
##  $ Category: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Age     : int  32 32 32 32 32 32 32 32 32 32 ...
##  $ Sex     : chr  "m" "m" "m" "m" ...
##  $ ALB     : num  38.5 38.5 46.9 43.2 39.2 41.6 46.3 42.2 50.9 42.4 ...
##  $ ALP     : num  52.5 70.3 74.7 52 74.1 43.3 41.3 41.9 65.5 86.3 ...
##  $ ALT     : num  7.7 18 36.2 30.6 32.6 18.5 17.5 35.8 23.2 20.3 ...
##  $ AST     : num  22.1 24.7 52.6 22.6 24.8 19.7 17.8 31.1 21.2 20 ...
##  $ BIL     : num  7.5 3.9 6.1 18.9 9.6 12.3 8.5 16.1 6.9 35.2 ...
##  $ CHE     : num  6.93 11.17 8.84 7.33 9.15 ...
##  $ CHOL    : num  3.23 4.8 5.2 4.74 4.32 6.05 4.79 4.6 4.1 4.45 ...
##  $ CREA    : num  106 74 86 80 76 111 70 109 83 81 ...
##  $ GGT     : num  12.1 15.6 33.2 33.8 29.9 91 16.9 21.5 13.7 15.9 ...
##  $ PROT    : num  69 76.5 79.3 75.7 68.7 74 74.5 67.1 71.3 69.9 ...
```

```
dat1<-data[-c(1,3)]
boxplot(dat1)
```



From the boxplot above, there seems to be an unequal variations between the predictor variables and unequal range that can be seen in "CHOL", "GGT" and "CREA" thus, scaling may be necessary

(2b) Use EDA techniques to explore the association between the target and predictors and identify those that are highly predictive of Hepatitis C incidence. Present at least THREE interesting findings and explain them with clear language.

```
prop.table(table(data$Category))*100
```

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

From the above output, it can be observed that about 87.8% of subjects are healthy donors and 12.2% of subjects are unhealthy or Hepatitis C from the Category

```
vars<-data[,c("Category","Age","ALB","ALP","ALT","AST","BIL",
                  "CHE","CHOL","CREA","GGT","PROT")]
cor_data<-cor(vars)

print("Correlation matrix")
```

```
## [1] "Correlation matrix"
```

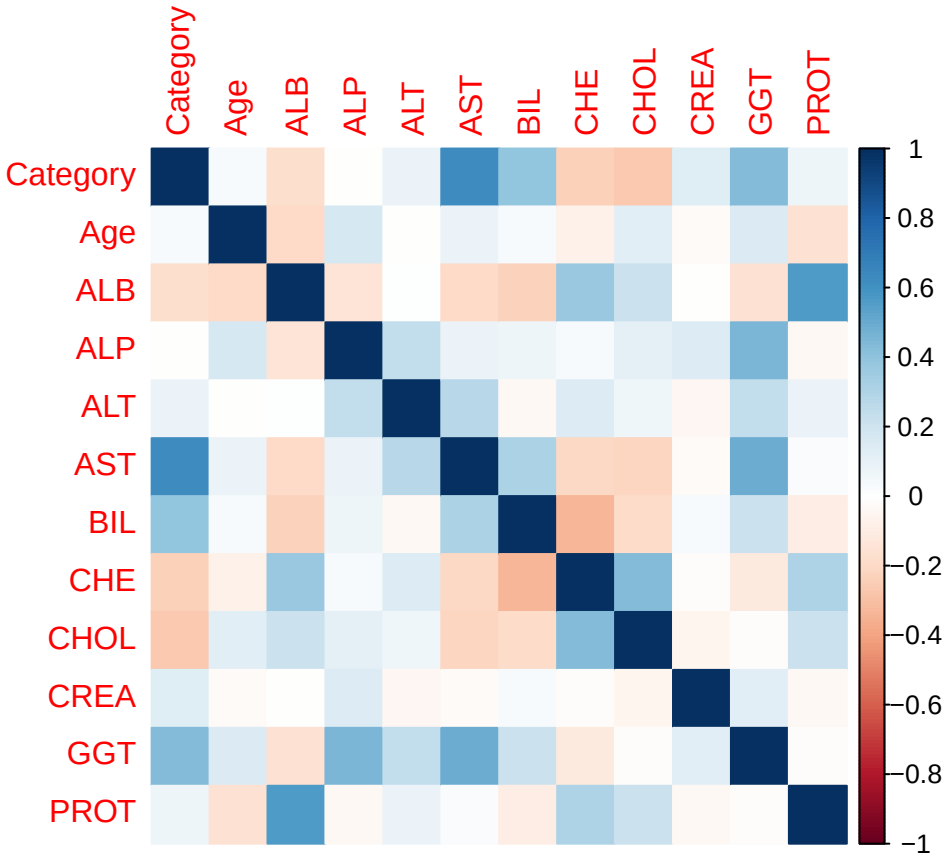
```
print(cor_data)
```

```
##              Category          Age          ALB          ALP          ALT
## Category  1.00000000  0.03778132 -0.177645726 -0.00568165  0.088768572
## Age       0.03778132  1.00000000 -0.195544534  0.17965858 -0.006031510
## ALB      -0.17764573 -0.19554453  1.000000000 -0.14247770  0.001983769
## ALP      -0.00568165  0.17965858 -0.142477705  1.00000000  0.243349317
## ALT       0.08876857 -0.00603151  0.001983769  0.24334932  1.000000000
## AST       0.62172398  0.08866590 -0.192886292  0.08672631  0.273324218
## BIL       0.39845142  0.03249182 -0.221538690  0.07155586 -0.038471360
## CHE      -0.23078510 -0.07509348  0.375515448  0.03975729  0.146962924
## CHOL     -0.26822485  0.12086810  0.218329036  0.11271417  0.061703794
## CREA      0.13677183 -0.02229637 -0.001757043  0.14790145 -0.043026034
## GGT       0.43768040  0.15308684 -0.154521663  0.45523402  0.248079511
## PROT      0.07142453 -0.15208907  0.564339663 -0.03367737  0.087857176
##                 AST          BIL          CHE         CHOL         CREA
## Category  0.62172398  0.39845142 -0.23078510 -0.26822485  0.136771831
## Age       0.08866590  0.03249182 -0.07509348  0.12086810 -0.022296365
## ALB      -0.19288629 -0.22153869  0.37551545  0.21832904 -0.001757043
## ALP       0.08672631  0.07155586  0.03975729  0.11271417  0.147901454
## ALT       0.27332422 -0.03847136  0.14696292  0.06170379 -0.043026034
## AST       1.00000000  0.31223141 -0.20853580 -0.21321670 -0.021387209
## BIL       0.31223141  1.00000000 -0.33317203 -0.18587215  0.031223528
## CHE      -0.20853580 -0.33317203  1.00000000  0.43102555 -0.011156955
## CHOL     -0.21321670 -0.18587215  0.43102555  1.00000000 -0.051100156
## CREA     -0.02138721  0.03122353 -0.01115696 -0.05110016  1.000000000
## GGT       0.49126255  0.21702381 -0.11034518 -0.01099913  0.121003326
## PROT      0.02905112 -0.09631060  0.30501916  0.21453424 -0.034443555
##                 GGT         PROT
## Category  0.43768040  0.07142453
## Age       0.15308684 -0.15208907
## ALB      -0.15452166  0.56433966
## ALP       0.45523402 -0.03367737
## ALT       0.24807951  0.08785718
## AST       0.49126255  0.02905112
## BIL       0.21702381 -0.09631060
## CHE      -0.11034518  0.30501916
## CHOL     -0.01099913  0.21453424
## CREA      0.12100333 -0.03444355
## GGT       1.00000000 -0.01973482
## PROT     -0.01973482  1.00000000
```

```
library("corrplot")
```

```
## corrplot 0.92 loaded
```

```
# as colour
corrplot(cor_data, method="color")
```



From the above correlation plot, We can observe that Category and AST are moderately positive correlated with a correlation coefficient of 0.62172, followed by GGT which has a correlation coefficient of 0.4377. Overall, apart from AST, the other variables have low correlation with the response variable, Category

BIVARIATE ASSOCIATION BETWEEN CATEGORY AND SEX

```
#Bivariate association of the category with the Sex predictor.
Chi_test <- chisq.test(table(data$Category, data$Sex))
Chi_test
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table(data$Category, data$Sex)
## X-squared = 2.7248, df = 1, p-value = 0.0988
```

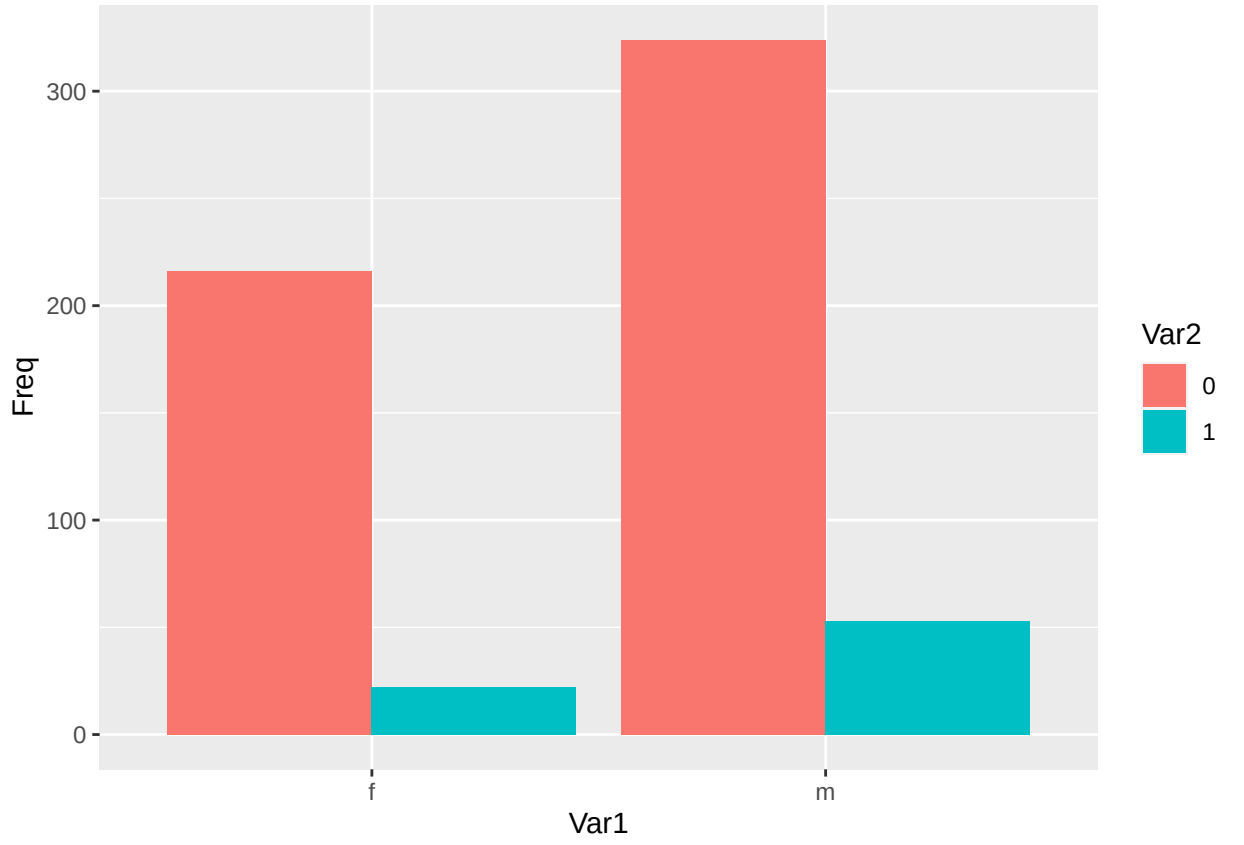
There is no association between Category and sex of the subject, since the p_value of the test is greater than 0.05. Thus, sex of the subject does not determine whether or not the individual is a healthy blood donor or Hepatitis C

SEX VERSES CATEGORY

```
vis_0<-table(data$Sex,data$Category)
d_vis_1<-as.data.frame(vis_0)
print(d_vis_1)
```

```
##   Var1 Var2 Freq
## 1    f    0  216
## 2    m    0  324
## 3    f    1   22
## 4    m    1   53
```

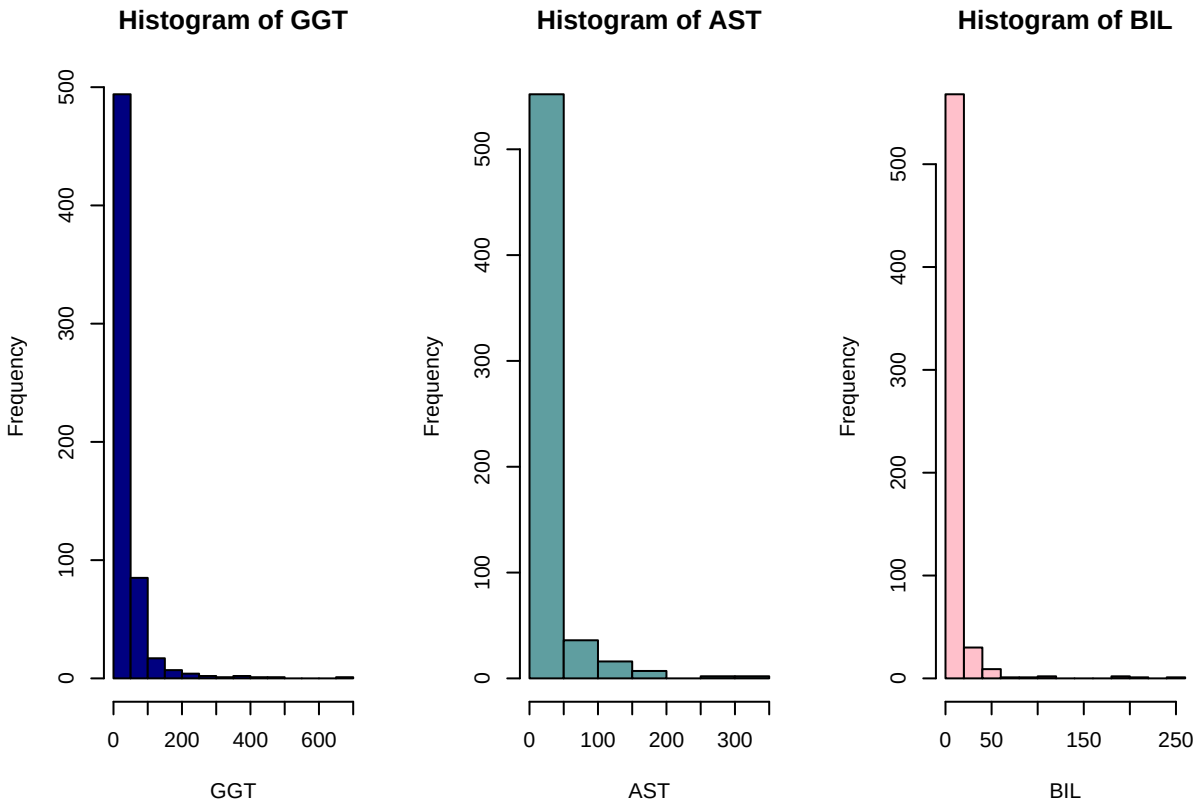
```
library(ggplot2)
a<-ggplot(d_vis_1, aes(x=Var1,y=Freq,fill=Var2)) +
geom_bar(position="dodge",stat="identity")
print(a)
```



From the plot above, it can be observed that category level that majority of male subjects falls in is either Blood Donor or Hepatitis C

DISTRIBUTION OF PREDICTORS THAT ARE POSITIVELY CORRELATED WITH THE RESPONSE

```
par(mfrow=c(1,3))
hist(data$GGT, col="navyblue", xlab = "GGT", main = "Histogram of GGT")
hist(data$AST, col="cadetblue", xlab = "AST", main = "Histogram of AST")
hist(data$BIL, col="pink", xlab = "BIL", main = "Histogram of BIL")
```



From the above plot, the distribution for the top three moderate positively correlated variables to the target variable, they were all found to be positively skewed. This means on the average, the three variables are likely to yield healthy blood donors compared to Hepatitis C.

OUTLIER DETECTION

(3) Choose one of the anomaly detection techniques that allows for prediction. First train the model with the data (excluding target variable) of healthy blood donors. Then apply it to predict the data of Hepatitis C patients and see if the method is able to successfully distinguish most of the Hepatitis C patients from healthy donors. In other words, can the method detect Hepatitis C patients as outliers based on what is learned from healthy blood donors?

```
D11<-data[data$Category==0,] #for healthy donors
D12<-data[data$Category==1,] #for Hepatitis C donors
```

```
library(isofor)
set.seed(123)
fit.isoforest <- iForest(D11[,-c(1,3)], nt=100, phi=256)
pred <- predict(fit.isoforest, newdata=D12[,-c(1,3)])
pred
```

```
##  [1] 0.3985627 0.5364788 0.6072965 0.5784038 0.4861767 0.4999842 0.4845095
##  [8] 0.5316853 0.5237067 0.4154769 0.4669449 0.4481212 0.4787512 0.6260988
## [15] 0.4287177 0.5012518 0.4555080 0.5171791 0.6765981 0.5990298 0.4329259
## [22] 0.4539581 0.4401225 0.6212884 0.5466856 0.5571624 0.5699262 0.5042411
```

```
## [29] 0.5574506 0.5645377 0.6686125 0.5275580 0.5839723 0.4924558 0.5077805
## [36] 0.4868639 0.6245868 0.5725779 0.4663631 0.4350244 0.4707522 0.4522716
## [43] 0.5572477 0.4515186 0.6512757 0.5868892 0.6103300 0.6109111 0.6374909
## [50] 0.6115919 0.6976680 0.5026322 0.6165161 0.6444548 0.6035325 0.6422369
## [57] 0.6701577 0.6612799 0.6949047 0.6226637 0.6174557 0.6328117 0.5784795
## [64] 0.4941291 0.6590888 0.6056954 0.5967213 0.5781564 0.5483015 0.6291435
## [71] 0.7009468 0.5736264 0.6611402 0.5251383 0.5281473
```

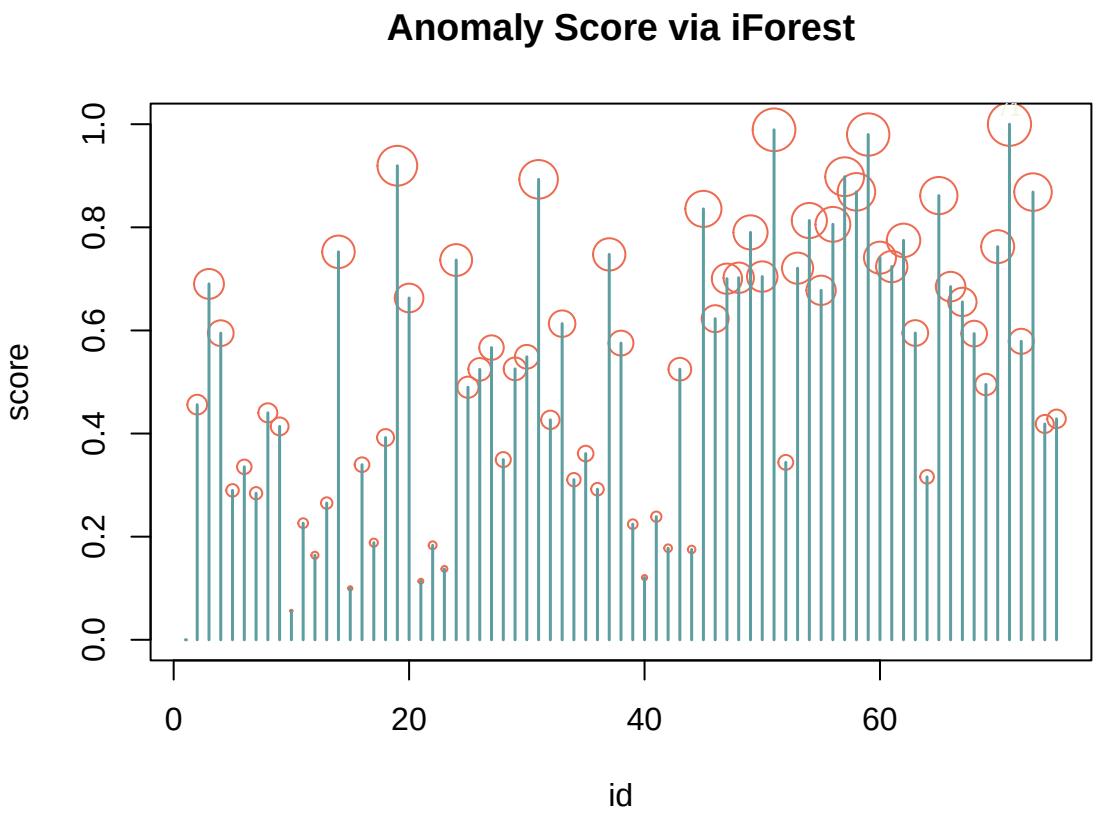
```
library(isofor)
score <- scale(pred, center = min(pred), scale = max(pred)-min(pred))

par(mfrow=c(1,1), mar=rep(4,4))
plot(x=1:length(score), score, type="p", pch=1,
main="Anomaly Score via iForest",
xlab="id", ylab="score", cex=score*3, col="coral2")

add.seg <- function(x) segments(x0=x[1], y0=0, x1=x[1], y1=x[2],lty=1, lwd=1.5, col="cadetblue")
apply(data.frame(id=1:length(score), score=score), 1, FUN=add.seg)
```

```
## NULL
```

```
eps <- 0.99
id.outliers <- which(score > quantile(score, eps))
text(id.outliers, score[id.outliers]+0.03, label=id.outliers,
col="beige", cex=0.7)
```



The above shows outlier detection within the data set.Interestingly, none of the potential outliers are visible.

DATA PARTITION

(4) We are going to try out a few machine learning tools to predict Hepatitis C incidence. Since we have a moderately-sized sample, we will use the following strategy of V -fold cross validation on the misclassfication errors, with V = 10. Randomly divide the data into V folds with stratification on the target variable Category. This warrants that similar proportions of 0s and 1s are preserved in each fold; see below for some exemplary R code. Use set.seed() to fix the random seed so that the results are easily reproducible.

```
set.seed(123)
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)
for (v in 1:V) {
train.v <- data[id.fold!=v, ]; test.v <- data[id.fold==v, ];
}
dim(train.v);dim(test.v)
```

```
## [1] 542  13
```

```
## [1] 73 13
```

From the V fold output, it can be observed that the training set has 542 observations and 13 variables. Also,the test set has 73 observations and 13 variables

PREDICTIVE MODELLING

LOGISTIC REGRESSION

(5a) Fit a regularized logistic regression model as one baseline classifier for comparison. You may use either LASSO or SCAD or any other penalty function of your choice. Explain how the optimal tuning parameter is determined. Present one final logistic model (when excluding one fold of data) during the process and interpret the results.

```
# Using LASSO
library(glmnet)
library(verification)



set.seed(125)
V <- 10
n <- NROW(data); n0 <- sum(data$Category==0); n1 <- n-n0;
missclass.rate = c()
error=c()
for (v in 1:V) {
error=c(error, v)
missclass.rate=c(missclass.rate, v)
}
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)
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$Category)
fit.lasso = glmnet(x=X, y=y, family="binomial", alpha=1,
lambda.min = 1e-4, nlambda = 100, standardize=T, thresh =
1e-07, maxit=1000)
CV = cv.glmnet(x=X, y=y, family="binomial", alpha = 1,
lambda.min = 1e-4, nlambda = 200, standardize = T,
thresh = 1e-07, maxit=1000)
#plot(CV)
# SELECTING THE BEST TUNING PARAMETER
best.lambda = CV$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)
yobs = test.v$Category
X.test = test.v[, -1]
pred.glm = predict(fit.final, newdata = X.test, type="response")
area = roc.area(yobs, pred.glm)$A
error[v] = area
print(paste("AUC for fold", v, ":", error[v]))
pred.rate = ifelse(pred.glm > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate
print(paste("Missclassification rate for fold", v,
":",missclass.rate[v]))
}
```

```
## [1] "AUC for fold 1 : 1"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] "AUC for fold 2 : 1"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [1] "AUC for fold 3 : 0.822463768115942"
## [1] "Missclassification rate for fold 3 : 0.0384615384615385"
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## [1] "AUC for fold 5 : 0.964622641509434"
## [1] "Missclassification rate for fold 5 : 0.0655737704918033"
## [1] "AUC for fold 6 : 0.994339622641509"
## [1] "Missclassification rate for fold 6 : 0.0476190476190476"
## [1] "AUC for fold 7 : 0.997916666666667"
## [1] "Missclassification rate for fold 7 : 0.0441176470588235"
## [1] "AUC for fold 8 : 0.955938697318008"
## [1] "Missclassification rate for fold 8 : 0.0447761194029851"
## [1] "AUC for fold 9 : 0.964912280701754"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] "AUC for fold 10 : 0.974842767295597"
```

```
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

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

```
## [1] "Average of AUC: 0.967503644424891"
```

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

```
## [1] "Average of Miss: 0.0450670732872178"
```

```
print(fit.best$beta)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  .
## Age          .
## Sexm         .
## ALB          .
## ALP         -0.008806889
## ALT          .
## AST          0.038990078
## BIL          0.020215125
## CHE          .
## CHOL        -0.336655483
## CREA         0.004692312
## GGT          0.006912545
## PROT         0.032345253
```

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

*The tuning parameter was selected by using the largest value of lambda such that error is within 1 standard error of the minimum. From the output above using the v-folds samples we observe that seven variables are selected with this choice of lamda selected via cross validation. **

```
set.seed(123)
fit.pen.lasso <- glm(factor(Category) ~ ALP + AST + BIL + CHOL +CREA + GGT + PROT, family = binomial, da
summary(fit.pen.lasso)
```

```
##
## Call:
## glm(formula = factor(Category) ~ ALP + AST + BIL + CHOL + CREA +
##     GGT + PROT, family = binomial, data = train.v)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0391  -0.2265  -0.1224  -0.0591   3.4011
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -12.672210   3.423586  -3.701 0.000214 ***
## ALP          -0.042174   0.009063  -4.653 3.27e-06 ***
## AST           0.067462   0.011611   5.810 6.25e-09 ***
## BIL           0.064657   0.021786   2.968 0.003000 **
## CHOL         -0.824572   0.224479  -3.673 0.000239 ***
## CREA          0.018765   0.004688   4.003 6.26e-05 ***
## GGT           0.021394   0.005137   4.165 3.11e-05 ***
## PROT          0.153358   0.045764   3.351 0.000805 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 404.38  on 552  degrees of freedom
## Residual deviance: 136.07  on 545  degrees of freedom
## AIC: 152.07
##
## Number of Fisher Scoring iterations: 7
```

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

```
##                    2.5 %      97.5 %
## (Intercept) -19.95666198 -6.54822374
## ALP          -0.06115543 -0.02512675
## AST           0.04573124  0.09178271
## BIL           0.02471414  0.10947605
## CHOL         -1.27711411 -0.38978675
## CREA          0.01011941  0.02914597
## GGT           0.01188190  0.03223723
## PROT          0.07045523  0.25001106
```

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

```
##  (Intercept)          ALP          AST          BIL         CHOL         CREA
## 3.137106e-06 9.587031e-01 1.069789e+00 1.066793e+00 4.384228e-01 1.018942e+00
##          GGT         PROT
## 1.021625e+00 1.165742e+00
```

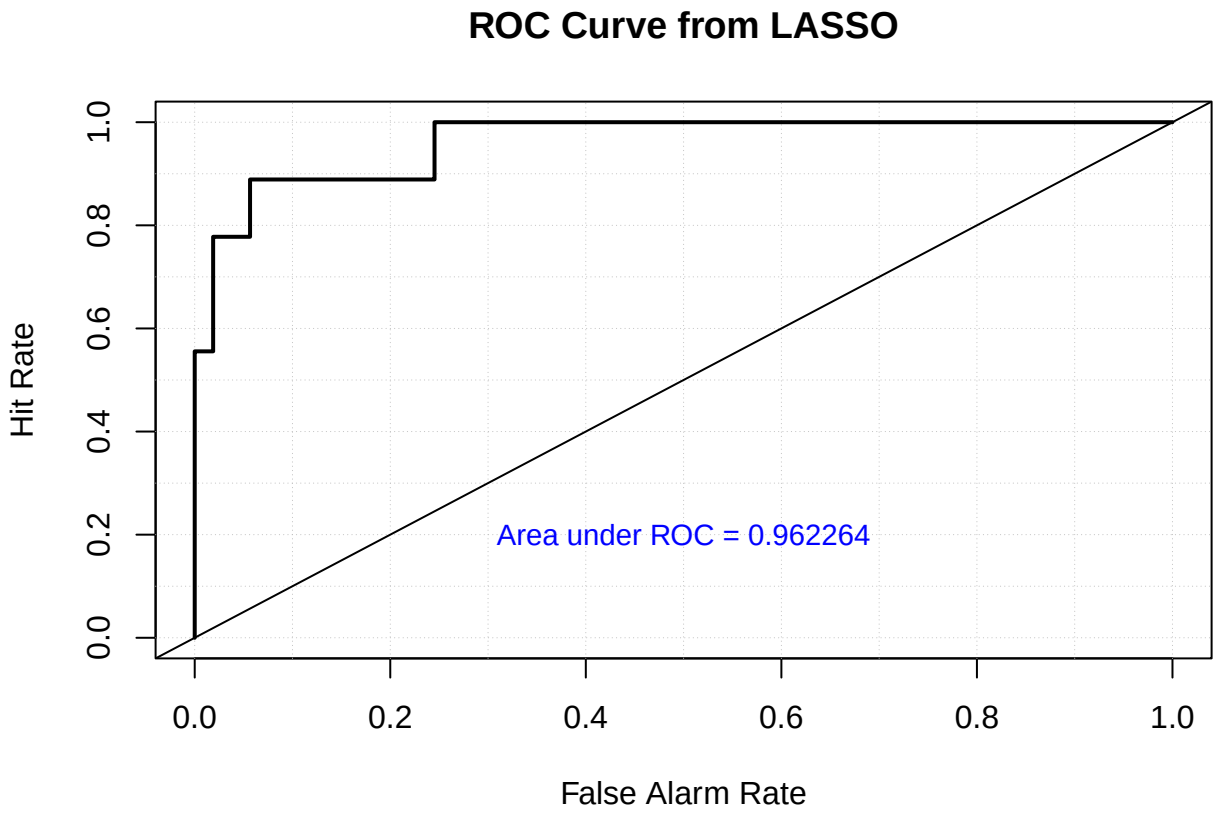
```
exp(confint(fit.pen.lasso, level=0.95))
```

```
##                    2.5 %      97.5 %
## (Intercept) 2.152444e-09 0.001432658
## ALP         9.406770e-01 0.975186297
## AST         1.046793e+00 1.096126617
## BIL         1.025022e+00 1.115693353
## CHOL        2.788408e-01 0.677201273
## CREA        1.010171e+00 1.029574873
## GGT         1.011953e+00 1.032762477
## PROT        1.072997e+00 1.284039612
```

```
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)
```



AUC provides an aggregate measure of performance across all possible classification thresholds. Hence at a threshold of 0.95, the probability that the model ranks a random blood donors more highly than a random Hepatitis C is 0.962264.

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

```
## [1] 0.08064516
```

The LASSO classifier has a miss-classification rate of 0.0806 with a threshold of 0.5.

RANDOM FOREST MODEL

(5b) Fit random forests as another baseline for comparison. Also, obtain the variable importance ranking from running RF (with one fold of data excluded)

```
library(randomForest)
set.seed(125)
V <- 10
n <- NROW(data); n0 <- sum(data$Category==0); n1 <- n-n0;
missclass.rate = c()
error=c()
for (v in 1:V) {
error=c(error, v)
missclass.rate=c(missclass.rate, v)
}
id.fold <- 1:n
id.fold[data$Category==0] <- sample(x=1:V, size=n0, replace=TRUE)
id.fold[data$Category==1] <- sample(x=1:V, size=n1, replace=TRUE)
for (v in 1:V) {
train.v <- data[id.fold!=v, ]; test.v <- data[id.fold==v, ];
mtry = tuneRF(train.v[ , -1], factor(train.v$Category), ntreeTry=200,
stepFactor=2, improve=0.05, trace=TRUE,
plot=FALSE, dobest=FALSE, printFlag = FALSE)
best.mtry = mtry[mtry[, 2] == min(mtry[, 2]), 1]
## Fitting model
fit.rf = randomForest(factor(Category) ~., mtry=best.mtry,
data=train.v, importance=TRUE, proximity=TRUE,
ntree=500)
yobs = test.v$Category
pred.rf = predict(fit.rf, newdata=test.v[, -c(1)], type="prob")[, 2]
area = roc.area(yobs, pred.rf)$A
error[v] = area
print(paste("AUC for fold", v, ":", error[v]))

pred.rate = ifelse(pred.rf > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate
print(paste("Missclassification rate for fold", v,
":",missclass.rate[v]))
}
```

```
## mtry = 3  OOB error = 3.24%
## Searching left ...
## mtry = 2     OOB error = 3.78%
## -0.1666667 0.05
## Searching right ...
## mtry = 6     OOB error = 3.42%
## -0.05555556 0.05
## [1] "AUC for fold 1 : 1"
## [1] "Missclassification rate for fold 1 : 0.0166666666666667"
## mtry = 3  OOB error = 3.62%
## Searching left ...
## mtry = 2     OOB error = 3.44%
```

```
## 0.05 0.05
## mtry = 1     OOB error = 4.34%
## -0.2631579 0.05
## Searching right ...
## mtry = 6     OOB error = 4.16%
## -0.2105263 0.05
## [1] "AUC for fold 2 : 1"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## mtry = 3  OOB error = 3.37%
## Searching left ...
## mtry = 2     OOB error = 2.49%
## 0.2631579 0.05
## mtry = 1     OOB error = 4.44%
## -0.7857143 0.05
## Searching right ...
## mtry = 6     OOB error = 3.55%
## -0.4285714 0.05
## [1] "AUC for fold 3 : 0.855072463768116"
## [1] "Missclassification rate for fold 3 : 0.0192307692307692"
## mtry = 3  OOB error = 3.2%
## Searching left ...
## mtry = 2     OOB error = 3.55%
## -0.1111111 0.05
## Searching right ...
## mtry = 6     OOB error = 3.37%
## -0.05555556 0.05
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## mtry = 3  OOB error = 3.61%
## Searching left ...
## mtry = 2     OOB error = 3.25%
## 0.1 0.05
## mtry = 1     OOB error = 3.79%
## -0.1666667 0.05
## Searching right ...
## mtry = 6     OOB error = 3.25%
## 0 0.05
## [1] "AUC for fold 5 : 1"
## [1] "Missclassification rate for fold 5 : 0.0491803278688525"
## mtry = 3  OOB error = 2.72%
## Searching left ...
## mtry = 2     OOB error = 3.08%
## -0.1333333 0.05
## Searching right ...
## mtry = 6     OOB error = 3.44%
## -0.2666667 0.05
## [1] "AUC for fold 6 : 0.979245283018868"
## [1] "Missclassification rate for fold 6 : 0.0634920634920635"
## mtry = 3  OOB error = 3.66%
## Searching left ...
## mtry = 2     OOB error = 3.66%
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 3.29%
```

```
## 0.1 0.05
## mtry = 12    OOB error = 3.29%
## 0 0.05
## [1] "AUC for fold 7 : 1"
## [1] "Missclassification rate for fold 7 : 0.0147058823529412"
## mtry = 3  OOB error = 2.55%
## Searching left ...
## mtry = 2     OOB error = 3.83%
## -0.5 0.05
## Searching right ...
## mtry = 6     OOB error = 3.47%
## -0.3571429 0.05
## [1] "AUC for fold 8 : 0.994252873563218"
## [1] "Missclassification rate for fold 8 : 0.0447761194029851"
## mtry = 3  OOB error = 3.47%
## Searching left ...
## mtry = 2     OOB error = 3.47%
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 3.29%
## 0.05263158 0.05
## mtry = 12    OOB error = 4.02%
## -0.2222222 0.05
## [1] "AUC for fold 9 : 0.995215311004785"
## [1] "Missclassification rate for fold 9 : 0.0588235294117647"
## mtry = 3  OOB error = 2.53%
## Searching left ...
## mtry = 2     OOB error = 2.53%
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 2.89%
## -0.1428571 0.05
## [1] "AUC for fold 10 : 0.967505241090147"
## [1] "Missclassification rate for fold 10 : 0.0645161290322581"
```

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

```
## [1] "Average of AUC: 0.979129117244513"
```

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

```
## [1] "Average of Miss: 0.0366751288947135"
```

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

```
## [1] 0.9791291
```

FITTING THE RF MODEL

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

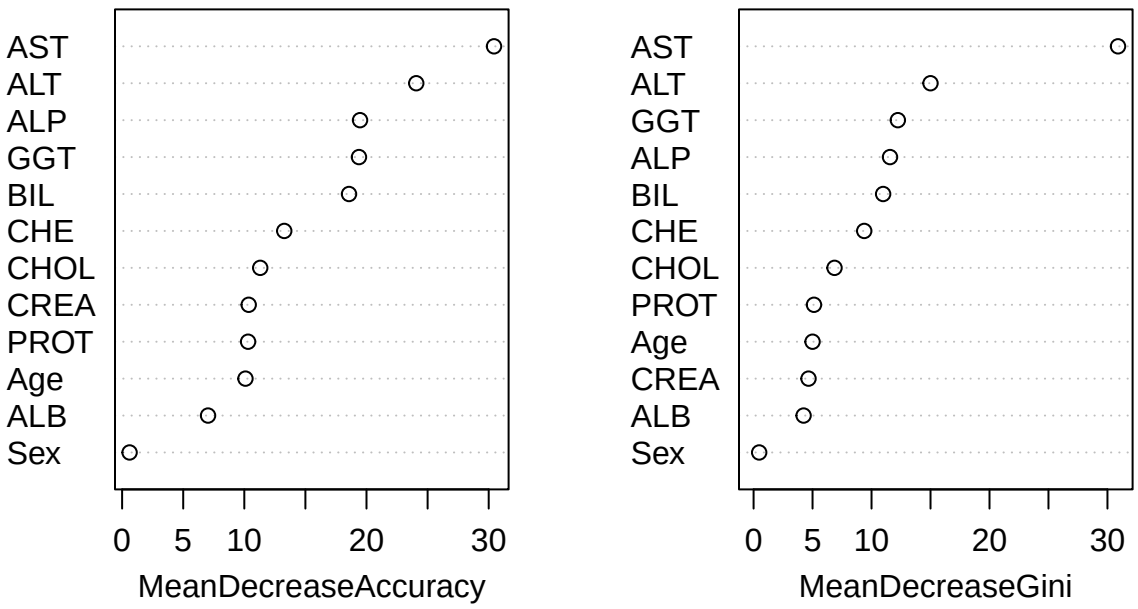
```
##
## Call:
##  randomForest(formula = factor(Category) ~ ., data = train.v,      mtry = best.mtry, importance = TRU
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.35%
## Confusion matrix:
##     0  1 class.error
## 0 485  2 0.004106776
## 1  11 55 0.166666667
```

```r
pred.rf = predict(fit.rf, newdata=test.v[, -1], type="prob")[, 2]
```

```r
# VARIABLE IMPORTANCE RANKING
round(importance(fit.rf), 2)
```

```
##          0     1 MeanDecreaseAccuracy MeanDecreaseGini
## Age   9.45  5.27                10.09             5.00
## Sex   1.86 -1.53                 0.61             0.48
## ALB   5.99  3.72                 7.03             4.24
## ALP  15.77 16.89                19.47            11.57
## ALT  22.37 15.91                24.07            15.00
## AST  23.12 30.32                30.44            30.90
## BIL   9.00 18.15                18.57            10.98
## CHE  11.37 10.07                13.27             9.38
## CHOL  7.89  8.04                11.30             6.86
## CREA  8.89  6.67                10.36             4.66
## GGT  12.94 16.43                19.38            12.23
## PROT 10.56  2.92                10.31             5.12
```

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



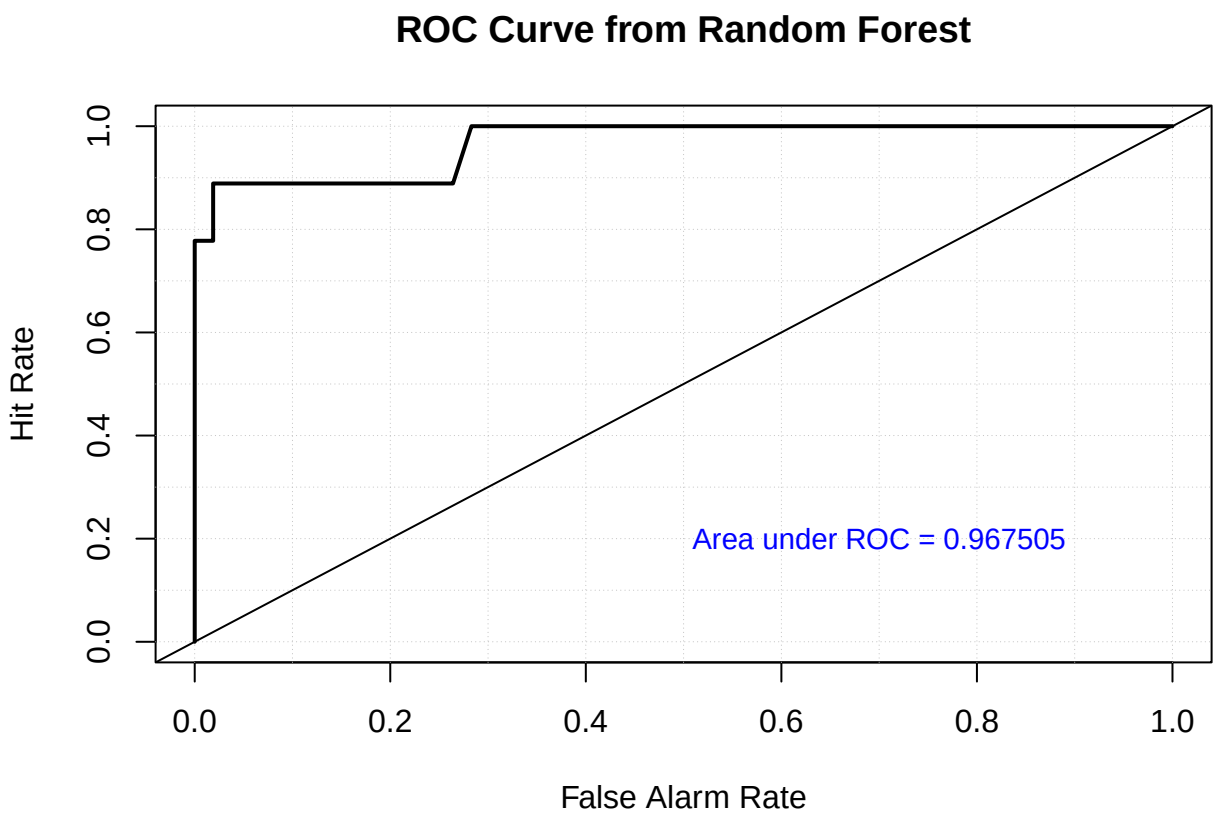
From the above plot, we can observe that AST ALT and ALP are the top three important variables in the model according to the random forest model

AREA UNDER THE CURVE OF RANDOM FOREST MODEL

```
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 area under the curve of the random Forest model is about 96.75%.Thus,the probability that the model ranks a random blood donors more highly than a random Hepatitis C is 96.75% according to the AUC of the random forest classifier above

MULTIVARIATE ADAPTIVE REGRESSION SPLINES THROUGH V-FOLD

```
set.seed(125)
library("earth")
library(ggplot2) # plotting
library(caret) # automating the tuning process


library(vip) # variable importance
library(pdp) # variable relationships
set.seed(125)
V <- 10
n <- NROW(data); n0 <- sum(data$Category==0); n1 <- n-n0;
missclass.rate = c()
error=c()
for (v in 1:V) {
error=c(error, v)
missclass.rate=c(missclass.rate, v)
}
id.fold <- 1:n
id.fold[data$Category==0] <- sample(x=1:V, size=n0, replace=TRUE)
id.fold[data$Category==1] <- sample(x=1:V, size=n1, replace=TRUE)
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$Category
yhat.mars <- predict(fit.mars, newdata=test.v[, -1], type="response")
area = roc.area(yobs, yhat.mars)$A
error[v] = area
print(paste("AUC for fold", v, ":", error[v]))
pred.rate = ifelse(pred.rf > 0.5, 1, 0)
miss.rate <- mean(yobs != pred.rate)
missclass.rate[v] = miss.rate
print(paste("Missclassification rate for fold", v,
":",missclass.rate[v]))
}
```

```
## [1] "AUC for fold 1 : 0.866071428571429"
## [1] "Missclassification rate for fold 1 : 0.0806451612903226"
## [1] "AUC for fold 2 : 0.982456140350877"
## [1] "Missclassification rate for fold 2 : 0.0967741935483871"
## [1] "AUC for fold 3 : 0.956521739130435"
## [1] "Missclassification rate for fold 3 : 0.17741935483871"
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.161290322580645"
## [1] "AUC for fold 5 : 0.860849056603774"
## [1] "Missclassification rate for fold 5 : 0.0483870967741935"
## [1] "AUC for fold 6 : 0.831132075471698"
## [1] "Missclassification rate for fold 6 : 0.0793650793650794"
## [1] "AUC for fold 7 : 0.866666666666667"
## [1] "Missclassification rate for fold 7 : 0.191176470588235"
## [1] "AUC for fold 8 : 0.984674329501916"
## [1] "Missclassification rate for fold 8 : 0.17910447761194"
## [1] "AUC for fold 9 : 0.813397129186603"
## [1] "Missclassification rate for fold 9 : 0.176470588235294"
## [1] "AUC for fold 10 : 0.979035639412998"
## [1] "Missclassification rate for fold 10 : 0.0645161290322581"
```

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

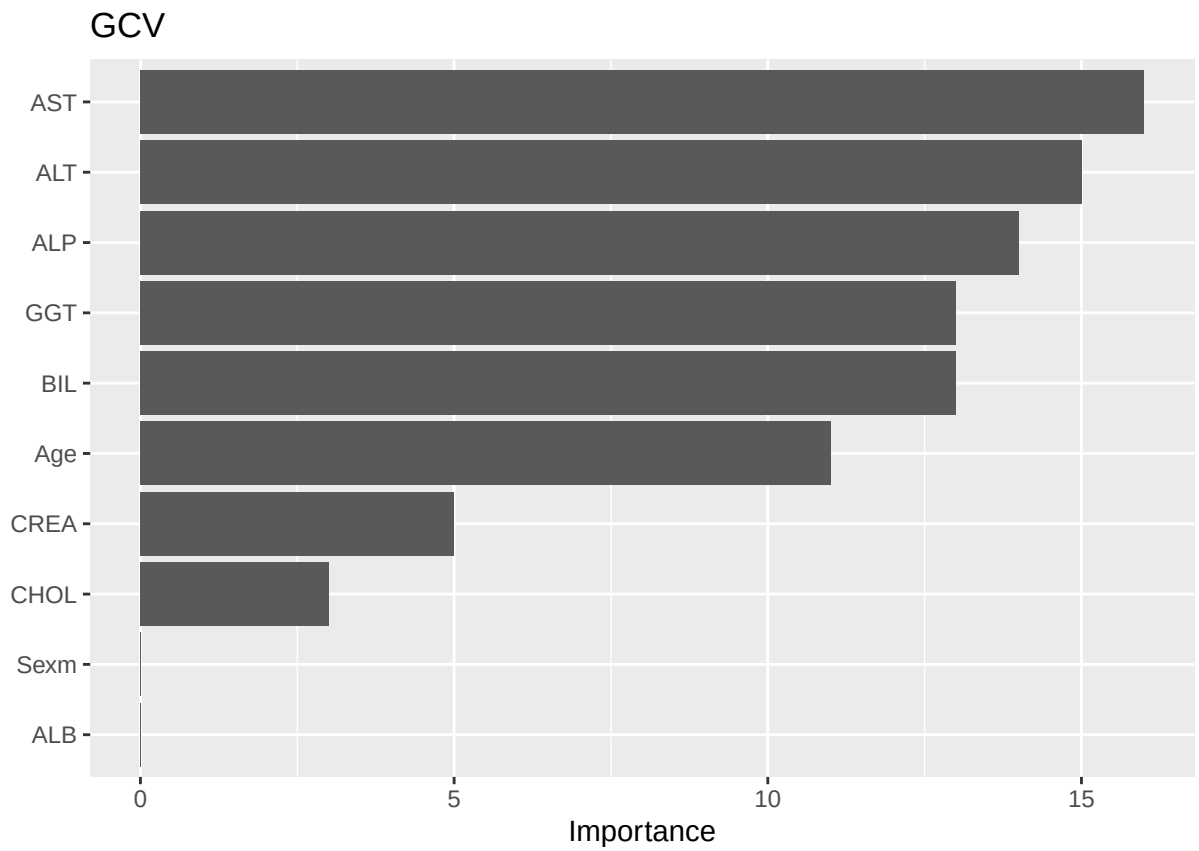
```
## [1] "Average of AUC: 0.91408042048964"
```

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

```
## [1] "Average of Miss: 0.125514887386507"
```

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

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



From the above variable importance plot, the top 3 important variables are AST, ALT and ALP

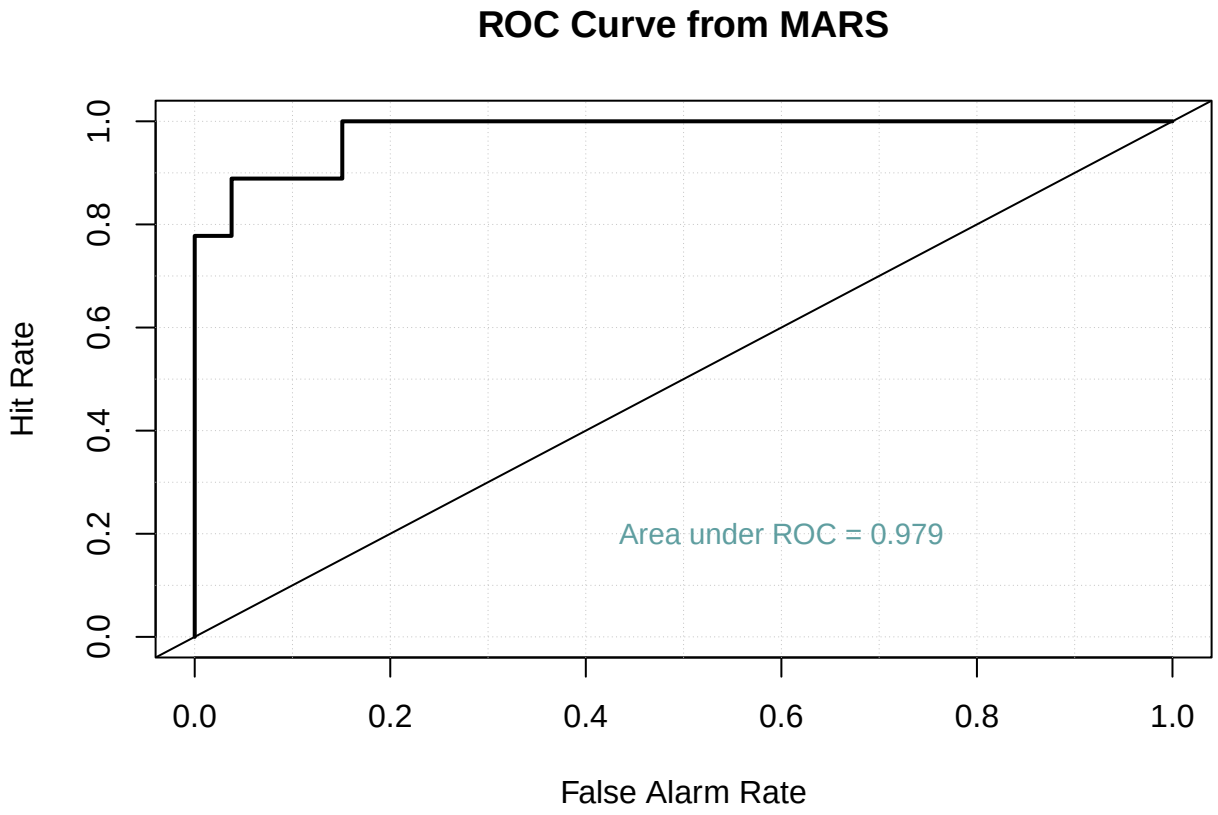
```
# PREDICTION
AUC.MARS <- ci.cvAUC(predictions=yhat.mars, labels=yobs,
folds=1:length(yhat.mars), confidence=0.95); AUC.MARS
```

```
## $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.
```

```
## 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 area under the curve of the MARS model is about 97.9%

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

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

From the above, the Multivariate Adaptive Regression Splines classifier has a missclassification rate of 0.0645 with a threshold of 0.5

ARTIFICIAL NEURAL NETWORKS

(5d) Fit at least two different artificial neural network (ANN) models, e.g., with different numbers of layers and different number of units, or using different ANN model types.

Data Preparing- Dealing with categorical variable and scaling

```
# 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))
```

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

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

missclass.rate = c()
error=c()

for (v in 1:V) {
  error=c(error, v)
  missclass.rate=c(missclass.rate, v)
}

for (v in 1:V){
    print(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
#plot(net1, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
#     col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)

    pred.11 = as.vector(neuralnet::compute(net1,
                                    covariate=valid_d.v[,-1])$net.result)
    area = roc.area(yobs, pred.11)$A
  error[v] = area
  print(paste("AUC for fold", v, ":", error[v]))

  pred.rate = ifelse(pred.11 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v,
           ":",missclass.rate[v]))

}
```

```
## [1] 1
## [1] "AUC for fold 1 : 1"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] 2
## [1] "AUC for fold 2 : 0.992982456140351"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [1] 3
## [1] "AUC for fold 3 : 0.855072463768116"
## [1] "Missclassification rate for fold 3 : 0.0576923076923077"
## [1] 4
## [1] "AUC for fold 4 : 0.991489361702128"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## [1] 5
## [1] "AUC for fold 5 : 0.974056603773585"
## [1] "Missclassification rate for fold 5 : 0.0819672131147541"
## [1] 6
## [1] "AUC for fold 6 : 0.975471698113208"
## [1] "Missclassification rate for fold 6 : 0.0476190476190476"
## [1] 7
## [1] "AUC for fold 7 : 0.99375"
## [1] "Missclassification rate for fold 7 : 0.0294117647058824"
## [1] 8
## [1] "AUC for fold 8 : 0.934865900383142"
## [1] "Missclassification rate for fold 8 : 0.0895522388059701"
## [1] 9
## [1] "AUC for fold 9 : 0.980861244019139"
## [1] "Missclassification rate for fold 9 : 0.0294117647058824"
## [1] 10
## [1] "AUC for fold 10 : 0.907756813417191"
## [1] "Missclassification rate for fold 10 : 0.0645161290322581"
```

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

```
## [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(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is about 96.06% based on the resulting average AUC

Model 1: 1 hidden layer, 3 units

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

V <- 10
```

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

missclass.rate = c()
error=c()

for (v in 1:V) {
  error=c(error, v)
  missclass.rate=c(missclass.rate, v)
}

for (v in 1:V){
    print(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
#plot(net1, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
#     col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)

    pred.11 = as.vector(neuralnet::compute(net1,
                                  covariate=valid_d.v[,-1])$net.result)
    area = roc.area(yobs, pred.11)$A
  error[v] = area
  print(paste("AUC for fold", v, ":", error[v]))

  pred.rate = ifelse(pred.11 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v,
           ":",missclass.rate[v]))

}
```

```
## [1] 1
## [1] "AUC for fold 1 : 1"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] 2
```

```
## [1] "AUC for fold 2 : 0.992982456140351"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [1] 3
## [1] "AUC for fold 3 : 0.855072463768116"
## [1] "Missclassification rate for fold 3 : 0.0576923076923077"
## [1] 4
## [1] "AUC for fold 4 : 0.991489361702128"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## [1] 5
## [1] "AUC for fold 5 : 0.974056603773585"
## [1] "Missclassification rate for fold 5 : 0.0819672131147541"
## [1] 6
## [1] "AUC for fold 6 : 0.975471698113208"
## [1] "Missclassification rate for fold 6 : 0.0476190476190476"
## [1] 7
## [1] "AUC for fold 7 : 0.99375"
## [1] "Missclassification rate for fold 7 : 0.0294117647058824"
## [1] 8
## [1] "AUC for fold 8 : 0.934865900383142"
## [1] "Missclassification rate for fold 8 : 0.0895522388059701"
## [1] 9
## [1] "AUC for fold 9 : 0.980861244019139"
## [1] "Missclassification rate for fold 9 : 0.0294117647058824"
## [1] 10
## [1] "AUC for fold 10 : 0.907756813417191"
## [1] "Missclassification rate for fold 10 : 0.0645161290322581"
```

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

```
## [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(error)
```

With 1 hidden layer and 3 unite, the average of the AUC is about 96.06% and average of missiclassification rate is about 0.048

Model 2: 2 hidden layer, 3 units

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

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

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

```r
for (v in 1:V) {
  error=c(error, v)
  missclass.rate=c(missclass.rate, v)
}

for (v in 1:V){
    print(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  2 hidden layer, 3 units
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)


# PLOT THE MODEL
#plot(net2, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
 #    col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)

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

  pred.rate = ifelse(pred.11 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v,
           ":",missclass.rate[v]))

}
```

```
## [1] 1
## [1] "AUC for fold 1 : 1"
## [1] "Missclassification rate for fold 1 : 0.0333333333333333"
## [1] 2
## [1] "AUC for fold 2 : 0.989473684210526"
## [1] "Missclassification rate for fold 2 : 0.0483870967741935"
## [1] 3
## [1] "AUC for fold 3 : 0.884057971014493"
```

```
## [1] "Missclassification rate for fold 3 : 0.0576923076923077"
## [1] 4
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0"
## [1] 5
## [1] "AUC for fold 5 : 0.780660377358491"
## [1] "Missclassification rate for fold 5 : 0.0983606557377049"
## [1] 6
## [1] "AUC for fold 6 : 0.911320754716981"
## [1] "Missclassification rate for fold 6 : 0.0952380952380952"
## [1] 7
## [1] "AUC for fold 7 : 1"
## [1] "Missclassification rate for fold 7 : 0.0147058823529412"
## [1] 8
## [1] "AUC for fold 8 : 0.92911877394636"
## [1] "Missclassification rate for fold 8 : 0.0746268656716418"
## [1] 9
## [1] "AUC for fold 9 : 0.925039872408293"
## [1] "Missclassification rate for fold 9 : 0.0735294117647059"
## [1] 10
## [1] "AUC for fold 10 : 0.884696016771488"
## [1] "Missclassification rate for fold 10 : 0.0967741935483871"
```

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

```
## [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(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is about 93.04% on the average based on the resulting AUC of the 2 hidden layer, 3 units of Artificial Neural Network with a average missclassification probability of 0.0592.

Model 3: 5 hidden layer, 4 units

```
# Using  5 hidden layer, 4 units
library(neuralnet);
set.seed(125)

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

missclass.rate = c()
error=c()

for (v in 1:V) {
  error=c(error, v)
```

```
  missclass.rate=c(missclass.rate, v)
}

for (v in 1:V){
    print(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  2 hidden layer, 3 units
options(digits=3)
net3 = neuralnet(Category~., data=train_scaled.v, hidden=c(5,4), rep = 1,#5 hidden layer, 4 units
        threshold = 0.05, stepmax = 1e+05, algorithm = "rprop+",
        err.fct = "ce", act.fct = "logistic",
        linear.output=FALSE, likelihood=TRUE)


# PLOT THE MODEL
#plot(net3, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
 #    col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)

    pred.11 = as.vector(neuralnet::compute(net3,
                                  covariate=valid_d.v[,-1])$net.result)
    area = roc.area(yobs, pred.11)$A
  error[v] = area
  print(paste("AUC for fold", v, ":", error[v]))

  pred.rate = ifelse(pred.11 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v,
           ":",missclass.rate[v]))

}
```

```
## [1] 1
## [1] "AUC for fold 1 : 0.928571428571429"
## [1] "Missclassification rate for fold 1 : 0.116666666666667"
## [1] 2
## [1] "AUC for fold 2 : 0.985964912280702"
## [1] "Missclassification rate for fold 2 : 0.0483870967741935"
## [1] 3
## [1] "AUC for fold 3 : 0.873188405797101"
## [1] "Missclassification rate for fold 3 : 0.0192307692307692"
## [1] 4
```

```
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.0384615384615385"
## [1] 5
## [1] "AUC for fold 5 : 0.806603773584906"
## [1] "Missclassification rate for fold 5 : 0.0819672131147541"
## [1] 6
## [1] "AUC for fold 6 : 0.967924528301887"
## [1] "Missclassification rate for fold 6 : 0.0793650793650794"
## [1] 7
## [1] "AUC for fold 7 : 0.966666666666667"
## [1] "Missclassification rate for fold 7 : 0.0588235294117647"
## [1] 8
## [1] "AUC for fold 8 : 0.796934865900383"
## [1] "Missclassification rate for fold 8 : 0.0746268656716418"
## [1] 9
## [1] "AUC for fold 9 : 0.96969696969697"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] 10
## [1] "AUC for fold 10 : 0.90356394129979"
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

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

```
## [1] "Average of AUC: 0.919911549209983"
```

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

```
## [1] "Average of Miss: 0.0642291567045554"
```

```
ANN3.miss<-mean(missclass.rate)
ANN3.AUC<-mean(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is about 91.99% on the average based on the resulting AUC of the 5 hidden layers, 4 units of Artificial Neural Network with a average missclassification probability of 0.064.

SUPPORT VECTOR MACHINE (SVM)

(5e) Fit at least two different SVM models, e.g., by varying the choice of kernel function and/or the parameters involved.

```
# Using Linear Kernel
set.seed(125)
library(caret)

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

set.seed(125)
V <- 10
index.cv <- sample(1:V, size=NROW(newdata), replace=TRUE)
```

```
missclass.rate = c()
error=c()
for (v in 1:V) {error=c(error, v)
missclass.rate=c(missclass.rate, v)
}


for (v in 1:V){
    print(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_Linear <- train(factor(Category) ~.,
                    data =train_scaled.v , method = "svmLinear",trControl=trctrl,
                  tuneLength = 10)

test_pred.svm1 <- predict(svm_Linear, newdata = valid_d.v[,-1])
test_pred.svm1 <- as.numeric(as.character(test_pred.svm1))

area = roc.area(yobs, test_pred.svm1)$A
error[v] = area
    print(paste("AUC for fold", v, ":", error[v]))

    pred.rate = ifelse(test_pred.svm1 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v, ":",
              missclass.rate[v]))
}
```

```
## [1] 1
## [1] "AUC for fold 1 : 0.991071428571429"
## [1] "Missclassification rate for fold 1 : 0.0166666666666667"
## [1] 2
## [1] "AUC for fold 2 : 0.882456140350877"
## [1] "Missclassification rate for fold 2 : 0.0483870967741935"
## [1] 3
## [1] "AUC for fold 3 : 0.822463768115942"
## [1] "Missclassification rate for fold 3 : 0.0576923076923077"
## [1] 4
## [1] "AUC for fold 4 : 0.88936170212766"
## [1] "Missclassification rate for fold 4 : 0.0384615384615385"
## [1] 5
## [1] "AUC for fold 5 : 0.803066037735849"
## [1] "Missclassification rate for fold 5 : 0.0655737704918033"
```

```
## [1] 6
## [1] "AUC for fold 6 : 0.9"
## [1] "Missclassification rate for fold 6 : 0.0317460317460317"
## [1] 7
## [1] "AUC for fold 7 : 0.8125"
## [1] "Missclassification rate for fold 7 : 0.0441176470588235"
## [1] 8
## [1] "AUC for fold 8 : 0.769157088122605"
## [1] "Missclassification rate for fold 8 : 0.0746268656716418"
## [1] 9
## [1] "AUC for fold 9 : 0.863636363636364"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] 10
## [1] "AUC for fold 10 : 0.722222222222222"
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

```
averageAUC = print(c("Average AUC:", mean(error)))
```

```
## [1] "Average AUC:"      "0.845593475088295"
```

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

```
## [1] "Average Misclassification rate:" "0.0502034732912153"
```

```
SVM1.miss<-mean(missclass.rate)
SVM1.AUC<-mean(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is about 84.55% on the average based on the resulting AUC of the Linear SVM with an average missclassification probability of 0.0502.

‘

C VALUE IN LINEAR CLASSIFIER SVM

```
# 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)

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

missclass.rate = c()
error=c()
for (v in 1:V) {error=c(error, v)
missclass.rate=c(missclass.rate, v)
}
```

```r
for (v in 1:V){
    print(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))

area = roc.area(yobs, test_pred.svm2)$A
error[v] = area
    print(paste("AUC for fold", v, ":", error[v]))

    pred.rate = ifelse(test_pred.svm2 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v, ":",
              missclass.rate[v]))
}
```

```
## [1] 1
## [1] "AUC for fold 1 : 0.982142857142857"
## [1] "Missclassification rate for fold 1 : 0.0333333333333333"
## [1] 2
## [1] "AUC for fold 2 : 0.891228070175439"
## [1] "Missclassification rate for fold 2 : 0.032258064516129"
## [1] 3
## [1] "AUC for fold 3 : 0.822463768115942"
## [1] "Missclassification rate for fold 3 : 0.0576923076923077"
## [1] 4
## [1] "AUC for fold 4 : 0.98936170212766"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## [1] 5
## [1] "AUC for fold 5 : 0.803066037735849"
## [1] "Missclassification rate for fold 5 : 0.0655737704918033"
## [1] 6
## [1] "AUC for fold 6 : 0.9"
## [1] "Missclassification rate for fold 6 : 0.0317460317460317"
## [1] 7
## [1] "AUC for fold 7 : 0.8125"
```

```
## [1] "Missclassification rate for fold 7 : 0.0441176470588235"
## [1] 8
## [1] "AUC for fold 8 : 0.769157088122605"
## [1] "Missclassification rate for fold 8 : 0.0746268656716418"
## [1] 9
## [1] "AUC for fold 9 : 0.863636363636364"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] 10
## [1] "AUC for fold 10 : 0.722222222222222"
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

```
averageAUC = print(c("Average AUC:", mean(error)))
```

```
## [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(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is about 85.56% on the average based on the resulting AUC of the C value in Linear Classifier SVM with a average missclassification probability of 0.0483.

Non-Linear Kernel 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)

missclass.rate = c()
error=c()
for (v in 1:V) {error=c(error, v)
missclass.rate=c(missclass.rate, v)
}


for (v in 1:V){
    print(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))


area = roc.area(yobs, test_pred.svm3)$A
error[v] = area
    print(paste("AUC for fold", v, ":", error[v]))

    pred.rate = ifelse(test_pred.svm3 > 0.5, 1, 0)
  miss.rate <- mean(yobs != pred.rate)
  missclass.rate[v] = miss.rate
  print(paste("Missclassification rate for fold", v, ":",
              missclass.rate[v]))
}
```

```
## [1] 1
## [1] "AUC for fold 1 : 0.982142857142857"
## [1] "Missclassification rate for fold 1 : 0.0333333333333333"
## [1] 2
## [1] "AUC for fold 2 : 0.891228070175439"
## [1] "Missclassification rate for fold 2 : 0.032258064516129"
## [1] 3
## [1] "AUC for fold 3 : 0.916666666666667"
## [1] "Missclassification rate for fold 3 : 0.0192307692307692"
## [1] 4
## [1] "AUC for fold 4 : 0.88936170212766"
## [1] "Missclassification rate for fold 4 : 0.0384615384615385"
## [1] 5
## [1] "AUC for fold 5 : 0.721698113207547"
## [1] "Missclassification rate for fold 5 : 0.114754098360656"
## [1] 6
## [1] "AUC for fold 6 : 0.840566037735849"
## [1] "Missclassification rate for fold 6 : 0.0634920634920635"
## [1] 7
## [1] "AUC for fold 7 : 1"
## [1] "Missclassification rate for fold 7 : 0"
## [1] 8
## [1] "AUC for fold 8 : 0.824712643678161"
## [1] "Missclassification rate for fold 8 : 0.0597014925373134"
## [1] 9
```

```
## [1] "AUC for fold 9 : 0.863636363636364"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] 10
## [1] "AUC for fold 10 : 0.722222222222222"
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

```
averageAUC = print(c("Average AUC:", mean(error)))
```

```
## [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(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is about 86.522 based on the resulting AUC of the Non-Linear Kernel SVM with a average missclassification probability of 0.04859

MODEL COMPARISM

```
Missclassification_Rate <-c(lasso.miss,rf.miss,MARS.miss,
                            SVM1.miss,SVM2.miss,SVM3.miss,ANN1.miss,
                            ANN2.miss,ANN3.miss)
AUC <- c(lasso.AUC,rf.AUC,MARS.AUC,SVM1.AUC,SVM2.AUC,
         SVM3.AUC,ANN1.AUC,ANN2.AUC,ANN3.AUC)
Measures <- data.frame("Classifier Method"= c("LASSO Regularized Regression",
                                    "RF","MARS","SVM- Linear","SVM- C value Linear","SVM- Radial","ANN-c
                       "MissClassification Rate"= Missclassification_Rate, "AUC"=AUC);
knitr::kable(Measures, align = "lcc")
```

Classifier.Method	MissClassification.Rate	AUC
LASSO Regularized Regression	0.045	0.968
RF	0.037	0.979
MARS	0.126	0.914
SVM- Linear	0.050	0.846
SVM- C value Linear	0.048	0.856
SVM- Radial	0.049	0.865
ANN-c(1,3)	0.049	0.961
ANN-c(2,3)	0.059	0.930
ANN-c(5,4)	0.064	0.920

Among the classifier techniques,The Random Forest which is a tree-based models turned out to be the best and stable classifiers as it properly create split directions, thus keeping only the efficient information. Given its highest C statistics Index and minimum miss classification rate.