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

Quaye E. George gequaye@miners.utep.edu

Due: 12/07/2020

Contents

1 Data Preparation

Bring in the data D and name it as, say, hr. Change the categorical variable salary in the data set to ordinal:

```
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
```

The data set has 615 observations with 13 variables. Missing Values was found to be present in the data set.

1.1 Modification of the target variable

a) 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 Dor
```

The target variable "Category" has been categorize into 0 and 1 indicating healthy donors and unhealthy donors.

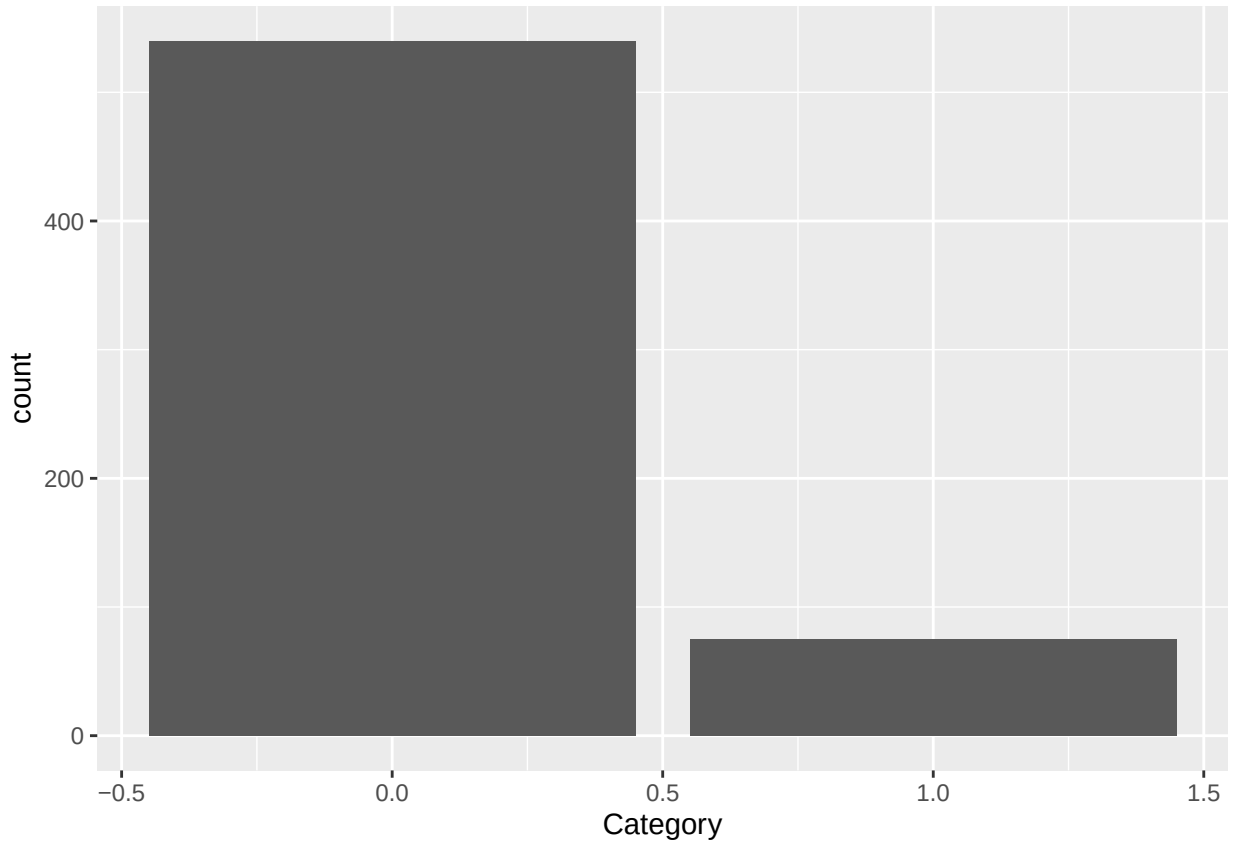
1.2 Distribution of the target variable

b) 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.

```
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
```

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




```
unique(dat$Category) ### Checking for missing values
```

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

A bar plot is drawn to check the distribution of the target variable 'Category' and to ascertain whether or not there exist a balance or unbalanced classification, given the output above there appears to be an unbalanced classification with "0" having a by far higher percentage

then 1 shown by the bars from the plot. Also the table indicates that 87.8% of the total observations falls under the healthy donors whiles 12.2% are unhealthy donors.

Note: This unbalanced classifiers can be balanced. The main objective of balancing classes is to either increasing the frequency of the minority class or decreasing the frequency of the majority class. This is done in order to obtain approximately the same number of instances for both the classes. Noticeable methods are Random Under-Sampling, Random Over-Sampling, Cluster-Based Over Sampling, Bagging Based techniques for imbalanced data etc.

```
#Mising values in Category
unique(is.na(dat$Category))
```

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

Checking for missing values in "Category", the output FALSE above indicates that there is no missing value in the target variable(category).

1.3 Inpect and impute missing values in the predictors

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
```

Given the table displayed above there appears some missing values with their corresponding missing percentages, noticeably is "ALP" with 18 missing observations making 2.927%.

```
#Missing  value imputation
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)
```

Values for all missing values is imputed using the package MICE. It is observed that out of the 13 variables , none of them have no missing values after the imputation.

1.4 Changing data matrix into numeric

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)
tail(Model)
```

```
##     (Intercept) Age Sexm ALB   ALP   ALT   AST BIL  CHE CHOL  CREA   GGT PROT
## 610           1  59    0  39  51.3  19.6 285.8  40 5.77 4.51 136.1 101.1 70.5
## 611           1  62    0  32 416.6   5.9 110.3  50 5.57 6.30  55.7 650.9 68.5
## 612           1  64    0  24 102.8   2.9  44.4  20 1.54 3.02  63.0  35.9 71.3
## 613           1  64    0  29  87.3   3.5  99.0  48 1.66 3.63  66.7  64.2 82.0
## 614           1  46    0  33  50.6  39.0  62.0  20 3.56 4.20  52.0  50.0 71.0
## 615           1  59    0  36  72.8 100.0  80.0  12 9.07 5.30  67.0  34.0 68.0
```

The data Matrix called Model is created by using the model.matrix.

2 Exploratory Data Analysis

2.1 Preliminary Statistical Analysis

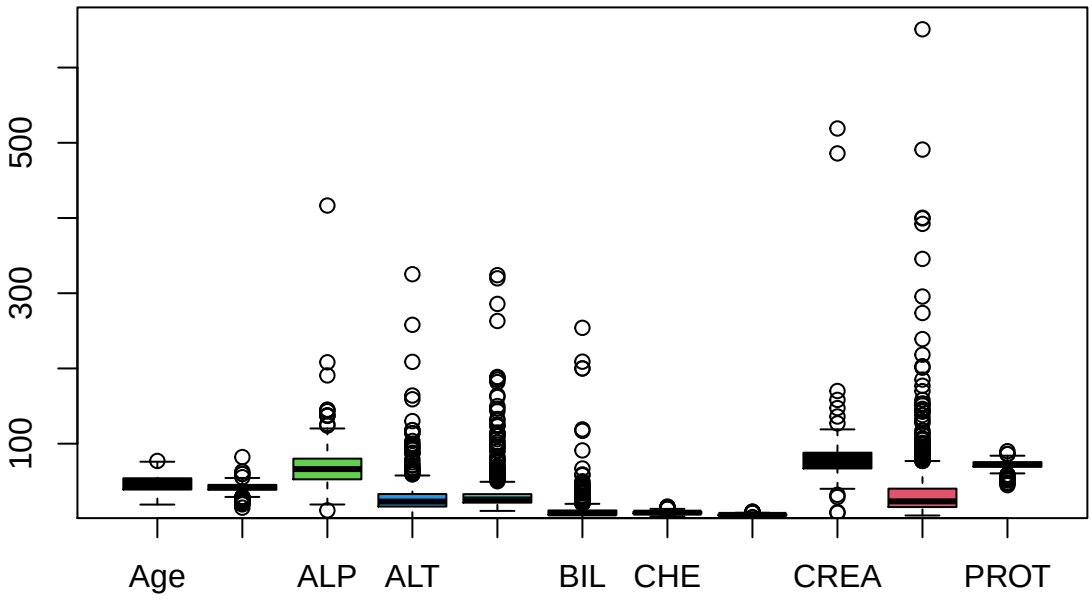
a) View the range and variations of the predictors. Is there a need to normalize them?

```
# Check the type of our features.
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 ...
```

The data has 11 numeric variables, one interger variable and one categorical variable (sex).

```
boxplot(data[, -c(1,3)], col = c(1:11), horizontal = F, ylim = c(1, 680), yaxs = "i")
```



Given the boxplot there appears to be unequal variations between the predictors variable and unequal range noticeably is the "CHOL", "GGT" and "CREA" hence scaling or normalization is necessary for some particular modelings approach.

```
#Percentage of Blood Donors and those of Hepatitis C
prop.table(table(data$Category))*100
```

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

Given the above output, it is observed that about 87.8049% of subjects are healthy donors and 12.1951% of subjects are unhealthy or Hepatitis C from the Category.

2.2 Association between the target and predictors

b) 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.

2.3 Correlation Matrix and Heat Map

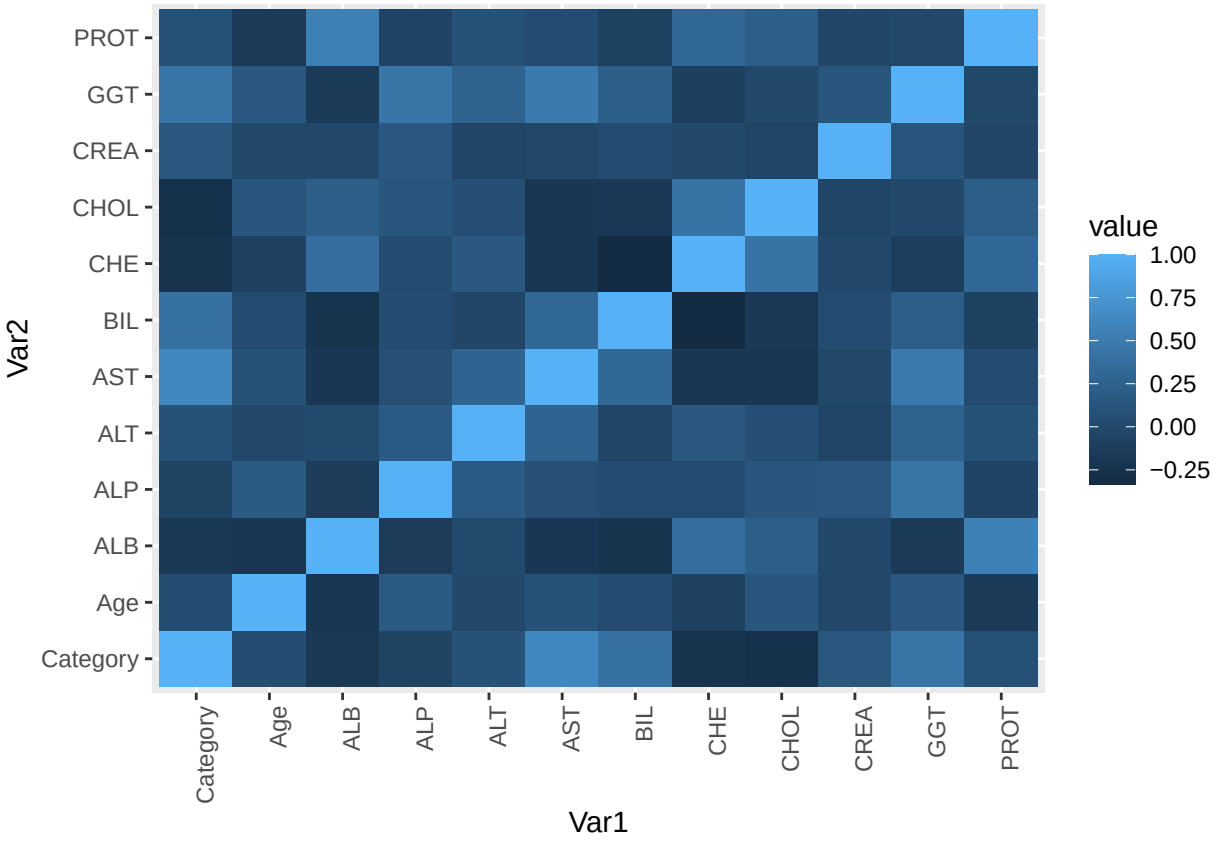
```
#Correlation Matrix
library(reshape2)
library(ggplot2)
cor_vars<-data[,c("Category","Age","ALB","ALP","ALT","AST","BIL",
                  "CHE","CHOL","CREA","GGT","PROT")]
cor(cor_vars)
```

```
##              Category          Age          ALB          ALP          ALT
## Category  1.00000000  0.037781320 -0.181441764 -0.05889966  0.088562579
## Age       0.03778132  1.000000000 -0.198028998  0.17746244 -0.005959648
## ALB      -0.18144176 -0.198028998  1.000000000 -0.13973796  0.001299239
## ALP      -0.05889966  0.177462438 -0.139737956  1.00000000  0.182887158
## ALT       0.08856258 -0.005959648  0.001299239  0.18288716  1.000000000
## AST       0.62172398  0.088665897 -0.193722675  0.06628314  0.273327843
## BIL       0.39845142  0.032491817 -0.221674074  0.03974631 -0.038454156
## CHE      -0.23078510 -0.075093477  0.375995441  0.02935035  0.146914603
## CHOL     -0.26496012  0.129185039  0.210104997  0.13076696  0.061884936
## CREA      0.13677183 -0.022296365 -0.001437393  0.14541462 -0.043020760
## GGT       0.43768040  0.153086840 -0.156275944  0.44874254  0.248108237
## PROT      0.07442785 -0.152624962  0.560080365 -0.05212340  0.089596462
```

```
##                  AST         BIL         CHE         CHOL         CREA
## Category  0.62172398  0.39845142 -0.23078510 -0.26496012  0.136771831
## Age       0.08866590  0.03249182 -0.07509348  0.12918504 -0.022296365
## ALB      -0.19372268 -0.22167407  0.37599544  0.21010500 -0.001437393
## ALP       0.06628314  0.03974631  0.02935035  0.13076696  0.145414622
## ALT       0.27332784 -0.03845416  0.14691460  0.06188494 -0.043020760
## AST       1.00000000  0.31223141 -0.20853580 -0.21126580 -0.021387209
## BIL       0.31223141  1.00000000 -0.33317203 -0.18223338  0.031223528
## CHE      -0.20853580 -0.33317203  1.00000000  0.42111809 -0.011156955
## CHOL     -0.21126580 -0.18223338  0.42111809  1.00000000 -0.045700340
## CREA     -0.02138721  0.03122353 -0.01115696 -0.04570034  1.000000000
## GGT       0.49126255  0.21702381 -0.11034518 -0.01053187  0.121003326
## PROT      0.03161215 -0.08358218  0.30259451  0.21452342 -0.033839832
##                  GGT        PROT
## Category  0.43768040  0.07442785
## Age       0.15308684 -0.15262496
## ALB      -0.15627594  0.56008036
## ALP       0.44874254 -0.05212340
## ALT       0.24810824  0.08959646
## AST       0.49126255  0.03161215
## BIL       0.21702381 -0.08358218
## CHE      -0.11034518  0.30259451
## CHOL     -0.01053187  0.21452342
## CREA      0.12100333 -0.03383983
## GGT       1.00000000 -0.01788994
## PROT     -0.01788994  1.00000000
```

```
trans<-cor(cor_vars)
melted_cormat <- melt(trans)

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Given the correlation matrix output table, it is observed that;

i. Category and AST are moderately positive correlated variables (0.62172398).

ii. AST and GGT are moderately positive correlated variables (0.4913).

iii. Category and GGT are moderately positive correlated variables (0.43768040).

iv. Category and BIL are moderately positive correlated variables (0.3985).

Also by the heat map, there is a positive correlation between Category, AST, and GGT.

2.4 Bivariate association of the category with the Sex predictor

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

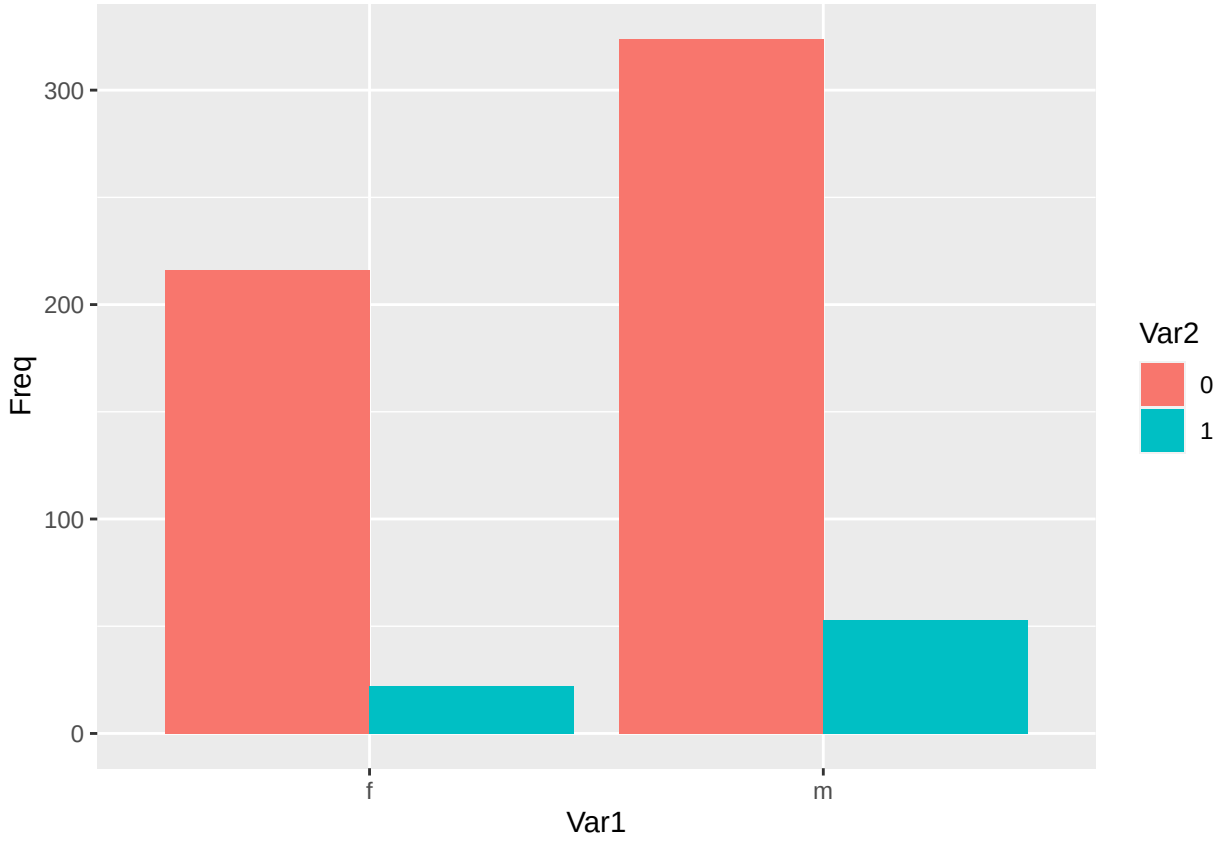
There is no association between Category and sex of the subject, since the p_value of the test is greater than 0.05. Hence the sex of the subject does not determine whether or not the individual is a healthy blood donor or Hepatitis C.

2.5 Sex V.S. Category

```
vis_1<-table(data$Sex,data$Category)
d_vis_1<-as.data.frame(vis_1)
print(d_vis_1)
```

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

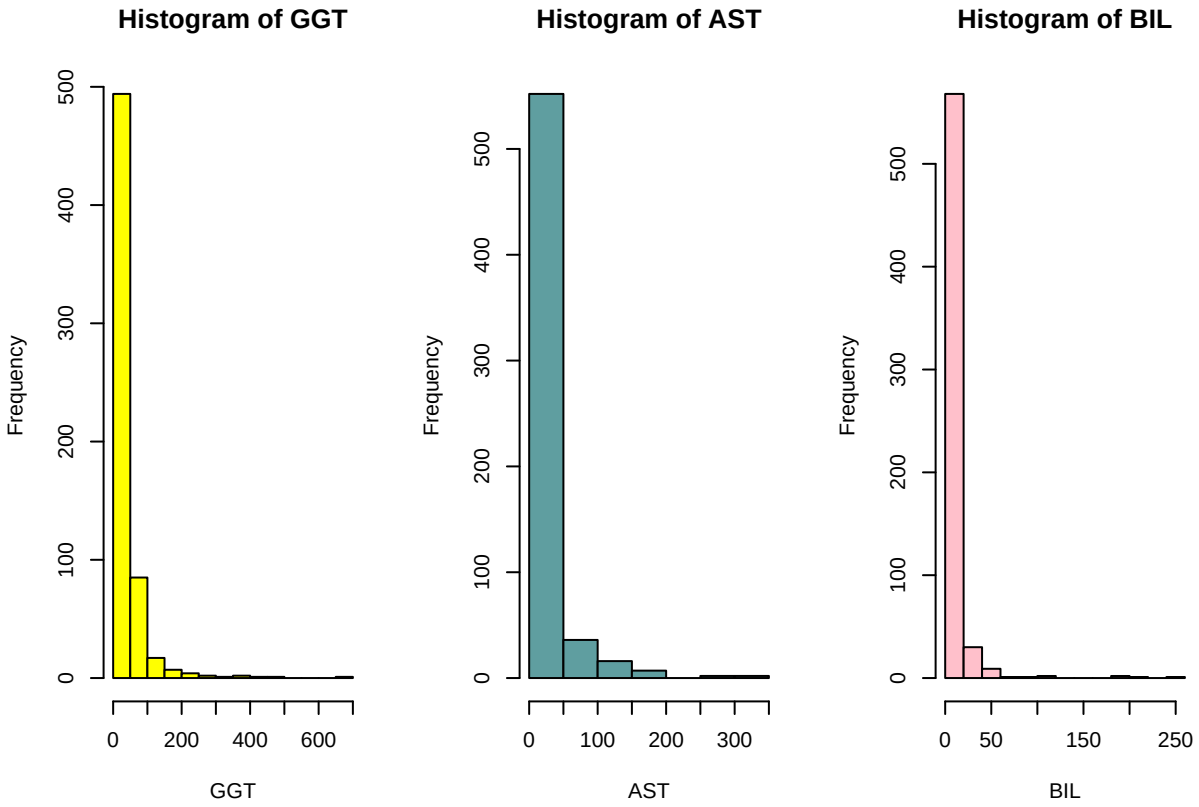
```
library(ggplot2)
p<-ggplot(d_vis_1, aes(x=Var1,y=Freq,fill=Var2)) +
 geom_bar(position="dodge",stat='identity')
print(p)
```



Given the plot above, it is observed for category level that majority of male subjects falls in either Blood Donor or Hepatitis C.

2.6 Distribution of predictor variables that positively correlate to the response

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



Given the plot of distribution for the 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.

```
R1<-data[data$Category==0,] #for healthy donors
R2<-data[data$Category==1,] #for Hepatitis C donors
```

3 Outlier Detection

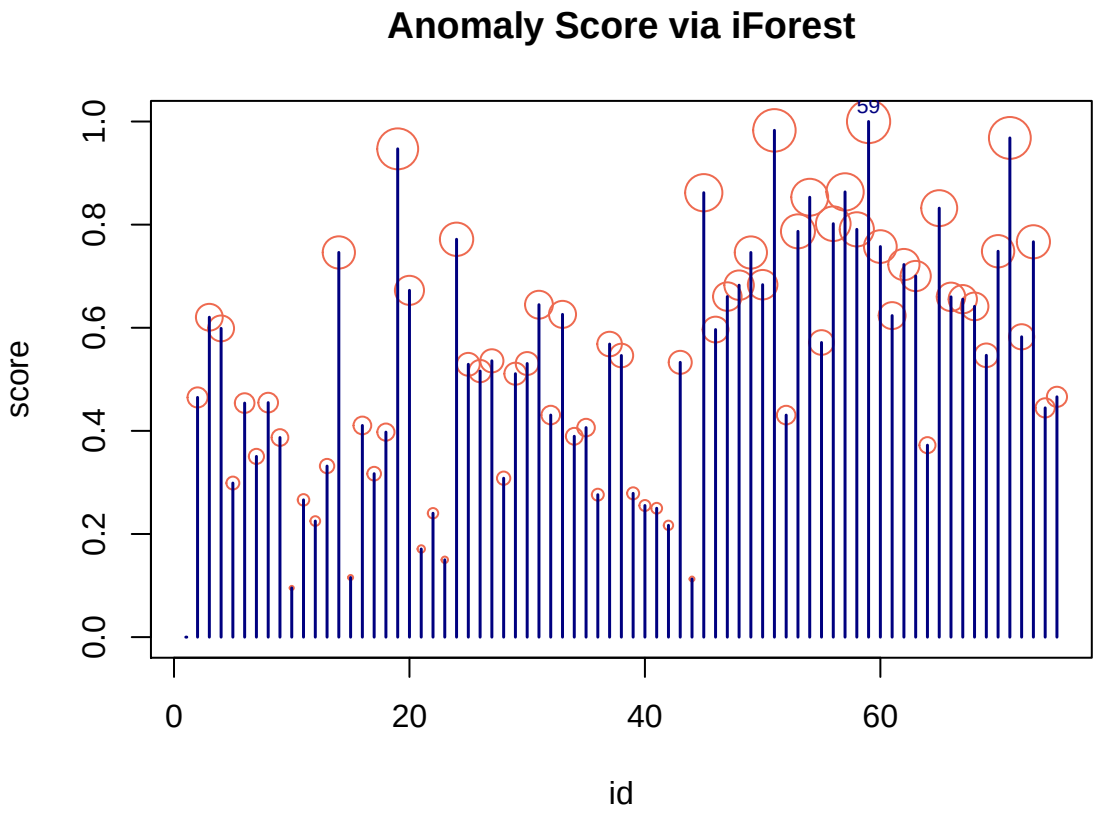
```
library(isofor)
set.seed(125)
fit.isoforest <- iForest(R1[,-c(1,3)], nt=100, phi=256)
pred <- predict(fit.isoforest, newdata=R2[,-c(1,3)])
```

```
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="navyblue")
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="navyblue", cex=0.7)
```



With probability of 0.99, it is observe from the plot that the defective part 59 is included in the top list of potential or expected outlier. Hence the method can detect Hepatitis C patients as outlier based on what is learned from healthy blood donors.

4 Data Partitioning

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

```
## [1] 62 13
```

```
dim(train.v)
```

```
## [1] 553  13
```

The test data has 62 observations and the train data has 553 observations all with 13 variables from the V-folds.

5 Predictive Modeling

5.1 Logistic Regression (LASSO) Via V-folds

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

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

```
## Loaded glmnet 4.0-2
```

```
library(verification)
```

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

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

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

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

```
## Spam version 2.5-1 (2019-12-12) 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

## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code

##
## Attaching package: 'fields'

## The following object is masked from 'package:questionr':
##
##     describe

## 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 v1.21-3. 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()
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]))

}
print(paste("Average of AUC:", mean(error)))
print(paste("Average of Miss:", mean(missclass.rate)))
print(fit.best$beta)
lasso.miss<-mean(missclass.rate)
lasso.AUC<-mean(error)
```

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is 0.96918 based on the resulting AUC of the LASSO with a missclassification probability of 0.04097.

Fitting the final best Logistic regression model with significant predictors from the best model of the V-fold

```
set.seed(125)
fit.pen.lasso <- glm(factor(Category) ~ ALP + AST + BIL + CHOL +CREA + GGT + PROT, famil
```

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 7 variables are selected with this choice of λ (selected via cross validation).

Output fitting results:

```
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
## -3.9469  -0.2085  -0.1088  -0.0497   3.4302
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.588310   3.528734  -3.284 0.001024 **
## ALP          -0.053451   0.010361  -5.159 2.48e-07 ***
## AST           0.065139   0.012134   5.368 7.94e-08 ***
## BIL           0.069309   0.022988   3.015 0.002570 **
## CHOL         -0.835095   0.238671  -3.499 0.000467 ***
## CREA          0.020324   0.005112   3.976 7.01e-05 ***
## GGT           0.026939   0.005864   4.594 4.35e-06 ***
## PROT          0.142764   0.047483   3.007 0.002642 **
## ---
## 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: 126.26  on 545  degrees of freedom
## AIC: 142.26
##
## Number of Fisher Scoring iterations: 8
```

From the output, we observe that ALP,AST,BIL,CHOL,CREA,GGT and PROT are all significant. This implies that all these features affect the simulation Categories.

The 95% confidence intervals for coeffcients β_j 's:

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

```
##                    2.5 %      97.5 %
## (Intercept) -19.16904876 -5.28407208
## ALP          -0.07553458 -0.03446937
```

```
## AST           0.04233024  0.09029228
## BIL           0.02676022  0.11644690
## CHOL         -1.31671073 -0.37733925
## CREA          0.01104208  0.03125848
## GGT           0.01623777  0.03937480
## PROT          0.05685014  0.24366951
```

The associated odds ratio:

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

```
##  (Intercept)          ALP          AST          BIL         CHOL         CREA
## 9.273869e-06 9.479524e-01 1.067308e+00 1.071767e+00 4.338335e-01 1.020532e+00
##          GGT         PROT
## 1.027305e+00 1.153457e+00
```

The 95% confidence intervals for the odds ratio:

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

```
##                    2.5 %      97.5 %
## (Intercept) 4.731381e-09 0.005071736
## ALP         9.272477e-01 0.966117927
## AST         1.043239e+00 1.094494131
## BIL         1.027121e+00 1.123497852
## CHOL        2.680154e-01 0.685683416
## CREA        1.011103e+00 1.031752160
## GGT         1.016370e+00 1.040160264
## PROT        1.058497e+00 1.275922585
```

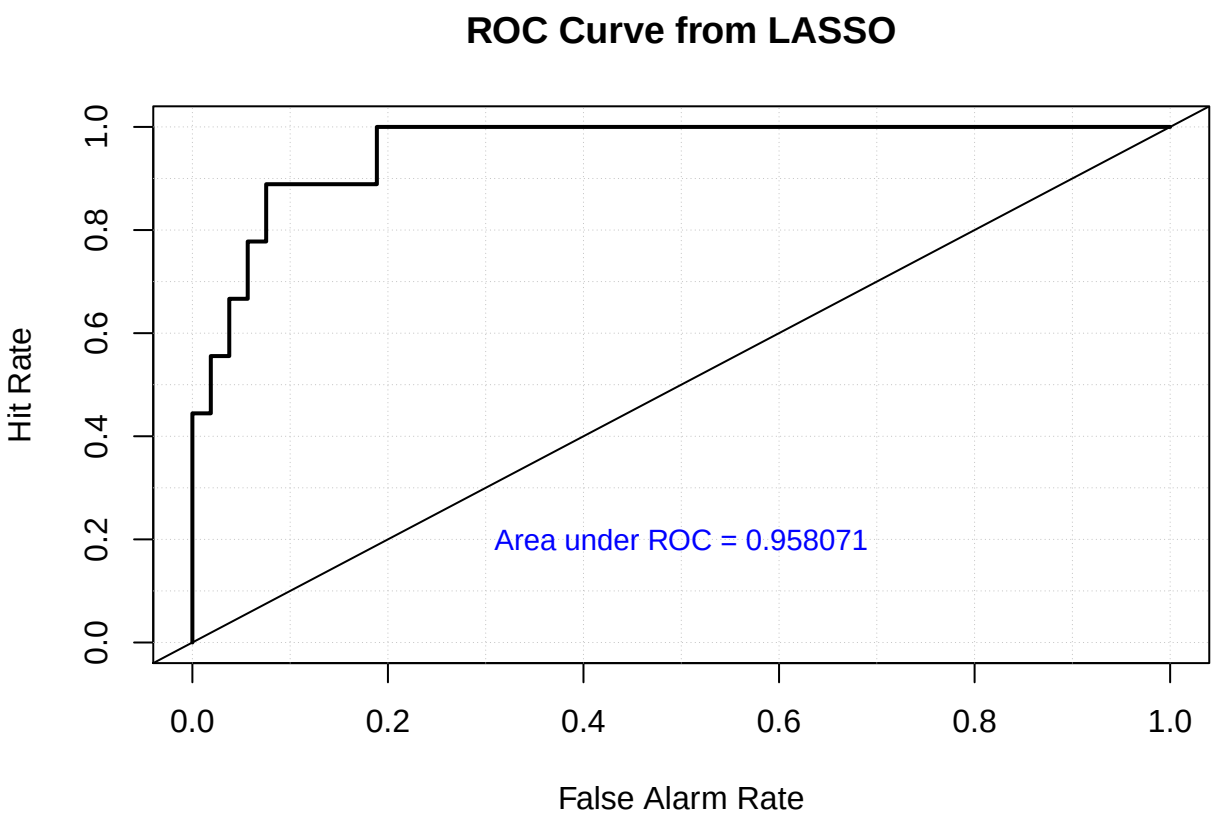
From the above, we only make inferences on the variables with CI ,excludes 1. All the variables which excludes 1 in the CI are significant.

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)
```



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 0.958071.

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

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

LASSO classifier has a missclassification rate of 0.080645 with a threshold of 0.5.

Advantages of LASSO classifier:
1. Dependent variable "Category" does not need to be normally distributed.
2. No homogeneity of variance assumption required.
3. Effective interpretation of results with the aid of the graph and output.

Disadvantages of LASSO classifier:
1. Requires more data to achieve stability.
2. Effective mostly on linearly separable.

5.2 Random Forest on V-Folds

```
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 = 2.88%
## Searching left ...
## mtry = 2     OOB error = 3.24%
## -0.125 0.05
## Searching right ...
## mtry = 6     OOB error = 3.42%
## -0.1875 0.05
## [1] "AUC for fold 1 : 1"
## [1] "Missclassification rate for fold 1 : 0.0166666666666667"
## mtry = 3  OOB error = 3.25%
## Searching left ...
## mtry = 2     OOB error = 3.25%
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 3.07%
## 0.05555556 0.05
## mtry = 12    OOB error = 4.16%
## -0.3529412 0.05
## [1] "AUC for fold 2 : 1"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## mtry = 3  OOB error = 2.84%
## Searching left ...
## mtry = 2     OOB error = 3.37%
## -0.1875 0.05
## Searching right ...
## mtry = 6     OOB error = 3.91%
## -0.375 0.05
## [1] "AUC for fold 3 : 0.858695652173913"
## [1] "Missclassification rate for fold 3 : 0.0192307692307692"
## mtry = 3  OOB error = 3.02%
## Searching left ...
## mtry = 2     OOB error = 3.73%
## -0.2352941 0.05
## Searching right ...
## mtry = 6     OOB error = 3.02%
## 0 0.05
```

```
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## mtry = 3  OOB error = 3.07%
## Searching left ...
## mtry = 2     OOB error = 2.89%
## 0.05882353 0.05
## mtry = 1     OOB error = 4.51%
## -0.5625 0.05
## Searching right ...
## mtry = 6     OOB error = 3.61%
## -0.25 0.05
## [1] "AUC for fold 5 : 1"
## [1] "Missclassification rate for fold 5 : 0.0491803278688525"
## mtry = 3  OOB error = 3.08%
## Searching left ...
## mtry = 2     OOB error = 3.08%
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 2.9%
## 0.05882353 0.05
## mtry = 12    OOB error = 3.08%
## -0.0625 0.05
## [1] "AUC for fold 6 : 0.975471698113208"
## [1] "Missclassification rate for fold 6 : 0.0634920634920635"
## mtry = 3  OOB error = 3.29%
## Searching left ...
## mtry = 2     OOB error = 3.29%
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 3.47%
## -0.05555556 0.05
## [1] "AUC for fold 7 : 1"
## [1] "Missclassification rate for fold 7 : 0"
## mtry = 3  OOB error = 3.1%
## Searching left ...
## mtry = 2     OOB error = 3.47%
## -0.1176471 0.05
## Searching right ...
## mtry = 6     OOB error = 3.47%
## -0.1176471 0.05
## [1] "AUC for fold 8 : 0.998084291187739"
## [1] "Missclassification rate for fold 8 : 0.0298507462686567"
## mtry = 3  OOB error = 3.11%
## Searching left ...
## mtry = 2     OOB error = 3.11%
```

```
## 0 0.05
## Searching right ...
## mtry = 6     OOB error = 3.29%
## -0.05882353 0.05
## [1] "AUC for fold 9 : 0.996810207336523"
## [1] "Missclassification rate for fold 9 : 0.0588235294117647"
## mtry = 3  OOB error = 2.17%
## Searching left ...
## mtry = 2     OOB error = 2.53%
## -0.1666667 0.05
## Searching right ...
## mtry = 6     OOB error = 3.44%
## -0.5833333 0.05
## [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: 0.980809748822438"
```

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

```
## [1] "Average of Miss: 0.0337120033459865"
```

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

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is 0.9808 based on the resulting AUC of the Random Forest with an average missclassification probability of 0.03371.

fitting one random forest model with training data through V-folds.

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

```
##
## Call:
##  randomForest(formula = factor(Category) ~ ., data = train.v,      mtry = best.mtry,
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
```

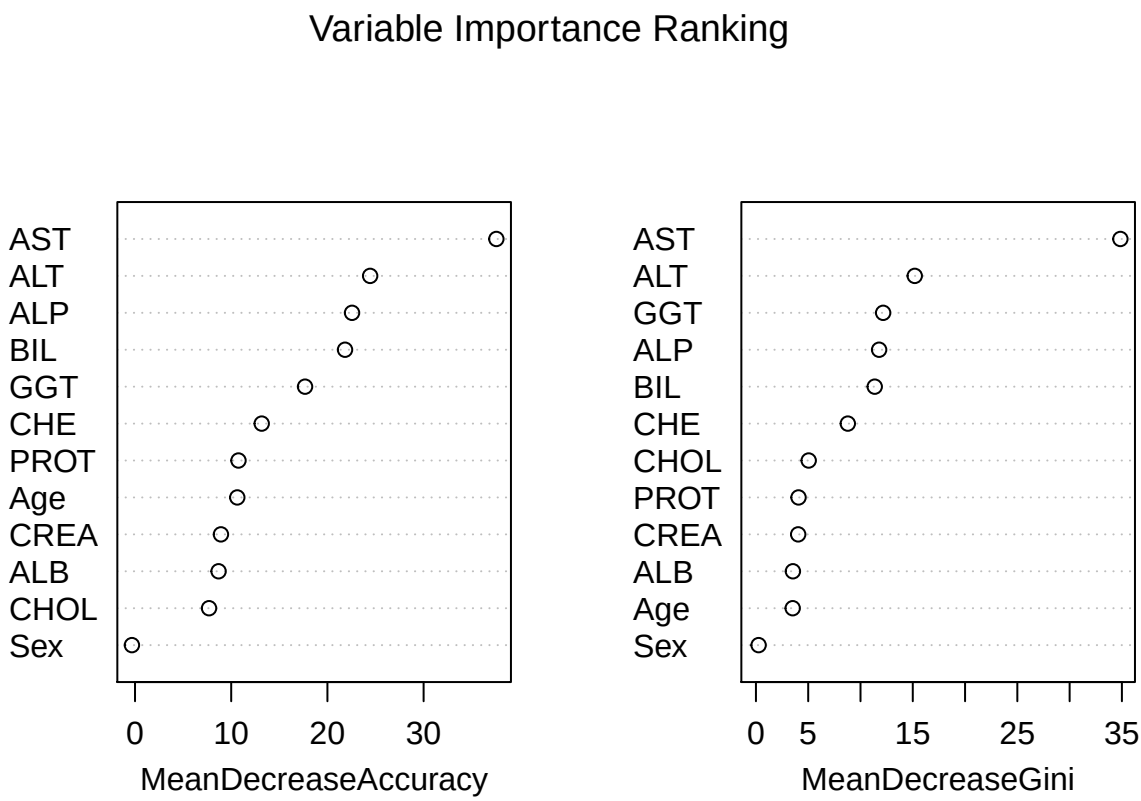
```
##         OOB estimate of  error rate: 2.17%
## Confusion matrix:
##     0  1 class.error
## 0 483  4 0.008213552
## 1   8 58 0.121212121
```

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

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

```
##          0     1 MeanDecreaseAccuracy MeanDecreaseGini
## Age   9.85  6.47                10.63             3.52
## Sex   0.23 -0.70                -0.32             0.26
## ALB   7.63  4.71                 8.69             3.54
## ALP  18.30 20.05                22.56            11.78
## ALT  22.48 17.87                24.44            15.19
## AST  27.82 36.85                37.56            34.86
## BIL   6.60 23.69                21.83            11.36
## CHE  11.44  9.15                13.17             8.79
## CHOL  6.27  4.98                 7.69             5.05
## CREA  7.28  6.15                 8.93             4.04
## GGT  10.82 16.17                17.68            12.17
## PROT 10.81  1.45                10.76             4.07
```

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



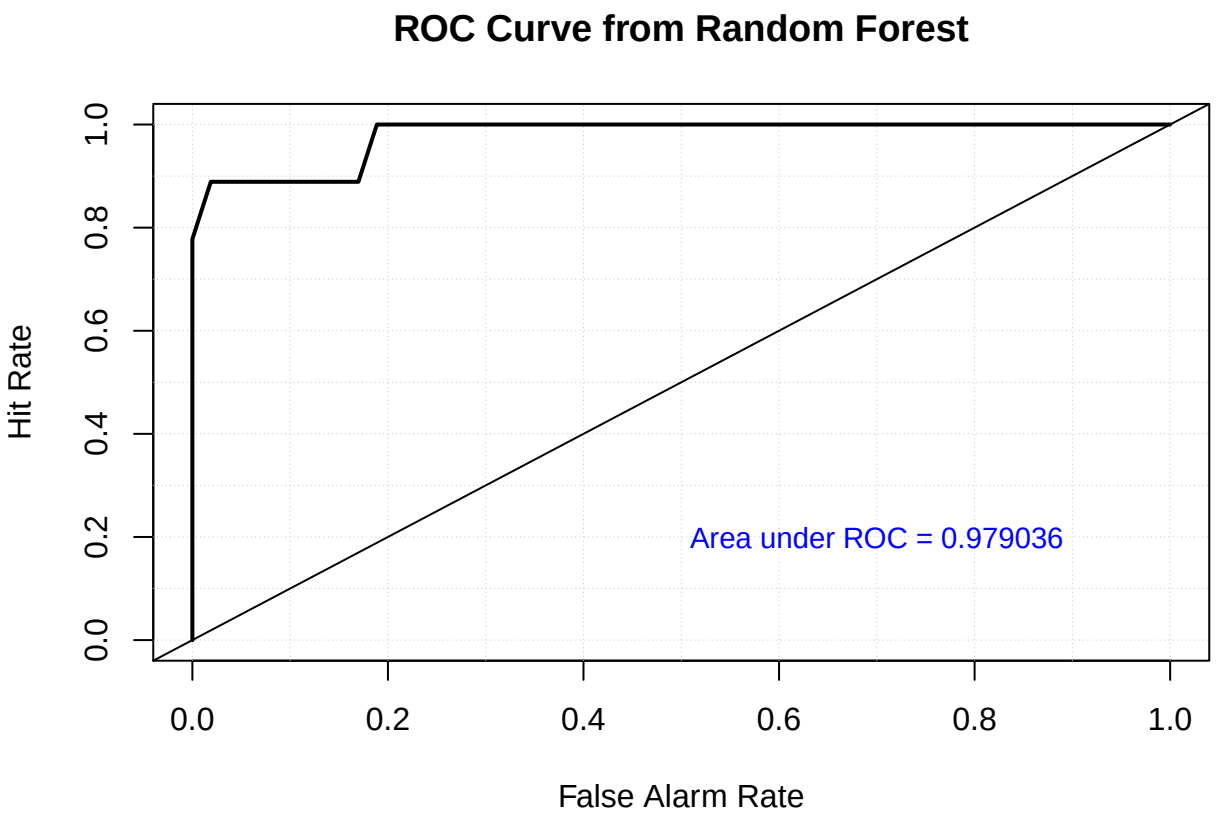
Based on the MeanDecreaseAccuracy, the top three variables according to the variable importance ranking for random forest are AST, ALT and ALP. The least significant variable is sex as indicated earlier on from the LASSO best variables.

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 probability that the model ranks a random blood donors more highly than a random Hepatitis C is 0.979036 according to the AUC of the random forest classifier above.

Advantages of Random Forest classifier:
1. High performance and accurate with reagrds to the modelling
2. Provides feature importance estimate
3. Can automatically handle missing values
4. No feature scaling is required

Disadvantages of Random Forest classifier:
1. Less interpret-ability, black box approach
2. Can over fit the data.
3. Requires more computational resources
4. Prediction time is high

5.3 Multivariate Adaptive Regression Splines through V-folds

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

```r
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 : 0.991071428571429"
## [1] "Missclassification rate for fold 1 : 0.0806451612903226"
## [1] "AUC for fold 2 : 0.8"
## [1] "Missclassification rate for fold 2 : 0.0967741935483871"
## [1] "AUC for fold 3 : 0.934782608695652"
```

```
## [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 : 0.879716981132076"
## [1] "Missclassification rate for fold 5 : 0.0483870967741935"
## [1] "AUC for fold 6 : 0.979245283018868"
## [1] "Missclassification rate for fold 6 : 0.0793650793650794"
## [1] "AUC for fold 7 : 0.86875"
## [1] "Missclassification rate for fold 7 : 0.191176470588235"
## [1] "AUC for fold 8 : 0.982758620689655"
## [1] "Missclassification rate for fold 8 : 0.17910447761194"
## [1] "AUC for fold 9 : 0.736044657097289"
## [1] "Missclassification rate for fold 9 : 0.176470588235294"
## [1] "AUC for fold 10 : 0.968553459119497"
## [1] "Missclassification rate for fold 10 : 0.0645161290322581"
```

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

```
## [1] "Average of AUC: 0.914092303832446"
```

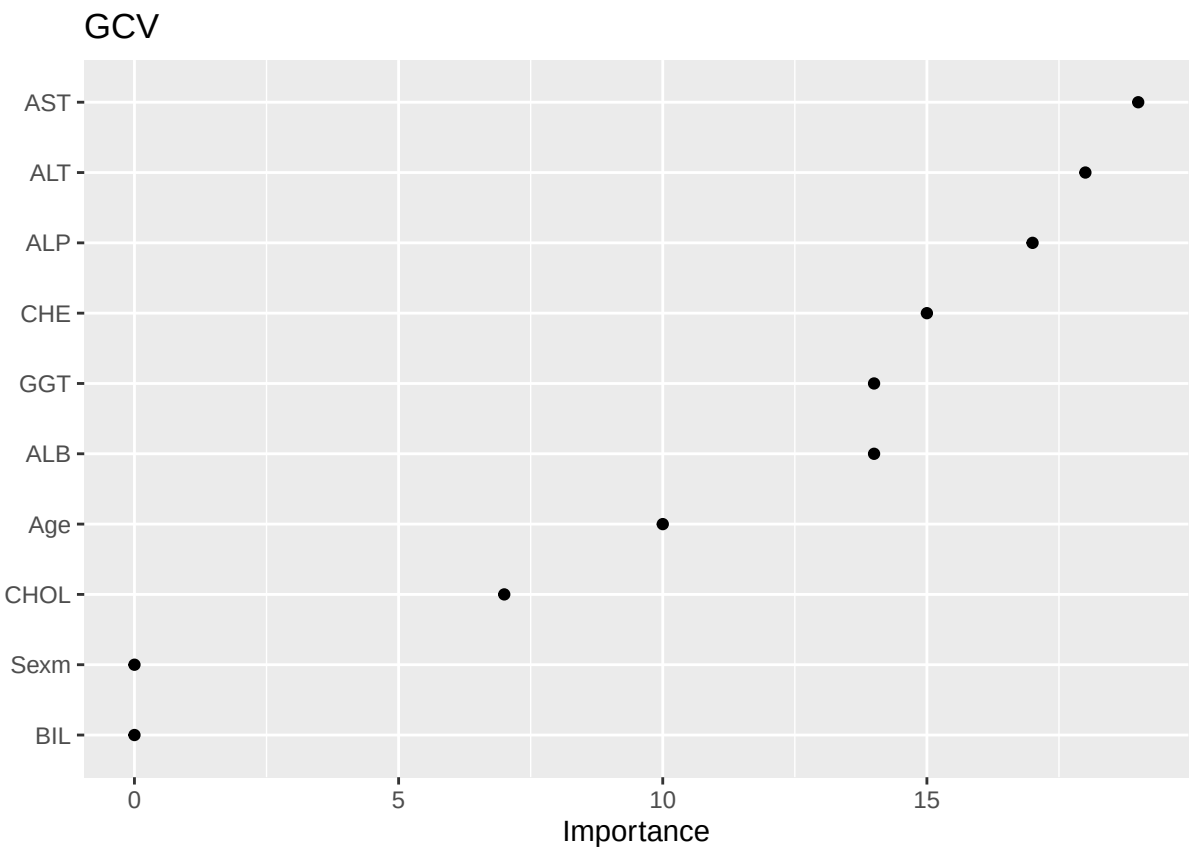
```
print(paste("Average of Miss:", mean(missclass.rate)))
```

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

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

The average probability that the model ranks a random blood donors more highly than a random Hepatitis C is 0.91409 based on the resulting AUC of the Multivariate Adaptive Regression Splines with a average missclassification probability of 0.1255.

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



Given the graph, the three top important variables is AST, ALT and ALP. This implies AST, ALT and ALP are the three top variables that predict a category of which a subject falls into.

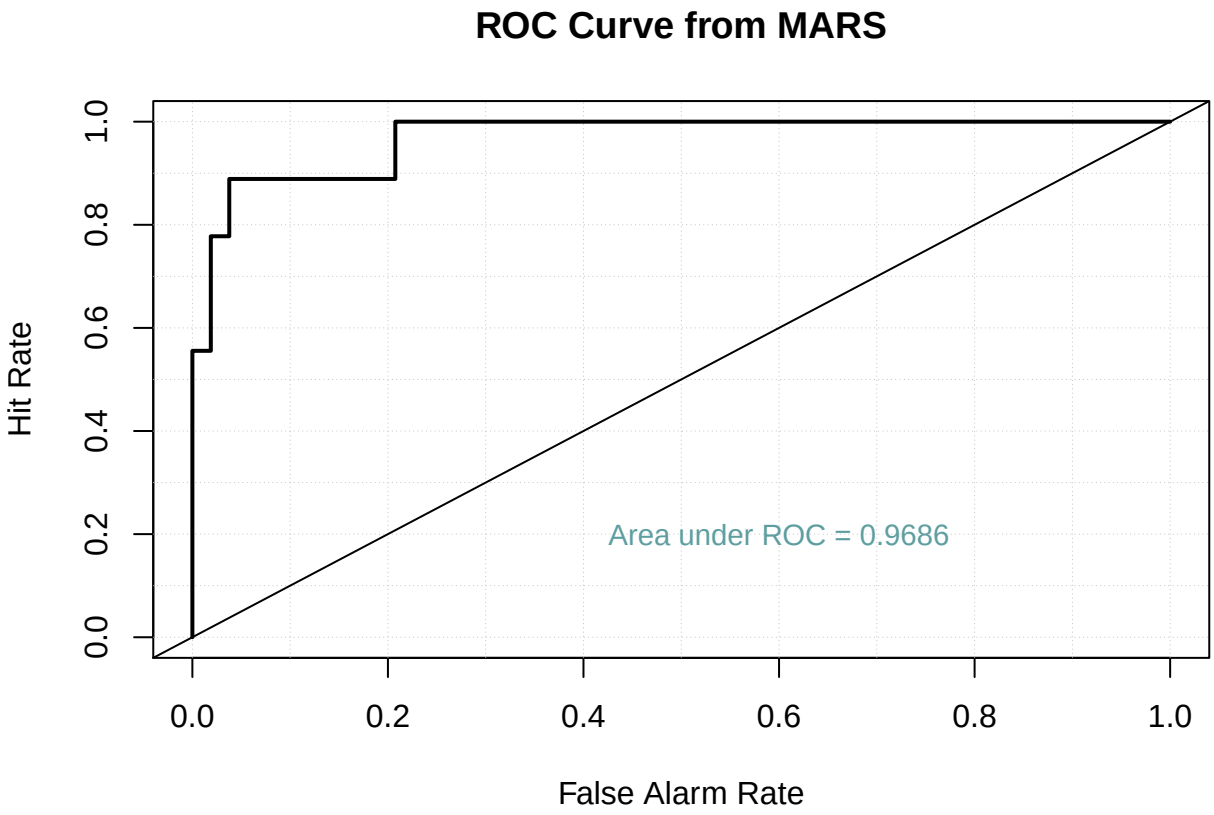
```
# PREDICTION
AUC.MARS <- ci.cvAUC(predictions=yhat.mars, labels=yobs,
                     folds=1:length(yhat.mars), confidence=0.95); AUC.MARS
```

```
## $cvAUC
## [1] 0.9685535
##
## $se
## [1] 0.02360281
##
## $ci
## [1] 0.9222928 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 probability that the model ranks a random blood donors more highly than a random Hepatitis C is 0.9686 based on the resulting AUC of the Multivariate Adaptive Regression Splines (MARS).

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

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

Multivariate Adaptive Regression Splines classifier has a missclassification rate of 0.0645 with a threshold of 0.5.

Advantages of Multivariate Adaptive Regression Splines classifier:
1. Robust to outliers
2. Works well with a large number of predictor variables
3. Automatically detects interactions between variables
4. It is an efficient and fast algorithm, despite its complexity

Disadvantages of Multivariate Adaptive Regression Splines classifier:
1. The resulting fitted function is not smooth(not differentiable along hinges)
2. Susceptible to overfitting
3. More difficult to understand and interpret than other methods
4. Not good with missing data

5.4 Support Vector Machines (SVM)

5.4.1 Linear SVM

```
# 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 : 0.857142857142857"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] 2
## [1] "AUC for fold 2 : 0.9"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [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 : 0.75"
## [1] "Missclassification rate for fold 6 : 0.0793650793650794"
## [1] 7
```

```
## [1] "AUC for fold 7 : 0.9375"
## [1] "Missclassification rate for fold 7 : 0.0147058823529412"
## [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:"      "0.837010559465905"
```

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

```
## [1] "Average Misclassification rate:" "0.0506391911508194"
```

```
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 0.8370 based on the resulting AUC of the Linear SVM with an average missclassification probability of 0.0506.

5.4.2 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)
}


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 : 0.857142857142857"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] 2
## [1] "AUC for fold 2 : 0.991228070175439"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [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 : 0.75"
## [1] "Missclassification rate for fold 6 : 0.0793650793650794"
## [1] 7
## [1] "AUC for fold 7 : 0.9375"
## [1] "Missclassification rate for fold 7 : 0.0147058823529412"
## [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:"      "0.840577810927894"
```

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

```
## [1] "Average Misclassification rate:" "0.0521317284642522"
```

```
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 0.84058 based on the resulting AUC of the C value in Linear Classifier SVM with a average missclassification probability of 0.05213.

5.4.3 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 : 0.991228070175439"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [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 : 0.784198113207547"
## [1] "Missclassification rate for fold 5 : 0.0983606557377049"
## [1] 6
## [1] "AUC for fold 6 : 0.881132075471698"
## [1] "Missclassification rate for fold 6 : 0.0634920634920635"
## [1] 7
## [1] "AUC for fold 7 : 0.991666666666667"
## [1] "Missclassification rate for fold 7 : 0.0147058823529412"
## [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 : 0.777777777777778"
## [1] "Missclassification rate for fold 10 : 0.0645161290322581"
```

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

```
## [1] "Average AUC:"      "0.890252293655084"
```

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

```
## [1] "Average Misclassification rate:" "0.045204854349481"
```

```
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 0.89025 based on the resulting AUC of the Non-Linear Kernel SVM with a average missclassification probability of 0.04520.

Advantages Support Vector Machines (SVM) classifier:
1. SVM works relatively well when there is a clear margin of separation between classes.
2. SVM is more effective in high dimensional spaces.
3. SVM is effective in cases where the number of dimensions is greater than the number of samples.
4. SVM is relatively memory efficient

Disadvantages of Support Vector Machines (SVM) classifiers
1. SVM algorithm is not suitable for large data sets.
2. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
3. In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
4. As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

5.5 Artificial Neural Networks (ANN)

5.5.1 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))
```

5.5.2 Model 1: 1 hidden layer, 3 units

```
#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 : 0.991071428571429"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] 2
## [1] "AUC for fold 2 : 0.992982456140351"
## [1] "Missclassification rate for fold 2 : 0.0483870967741935"
## [1] 3
## [1] "AUC for fold 3 : 0.847826086956522"
## [1] "Missclassification rate for fold 3 : 0.0384615384615385"
## [1] 4
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.0384615384615385"
## [1] 5
## [1] "AUC for fold 5 : 0.983490566037736"
## [1] "Missclassification rate for fold 5 : 0.0655737704918033"
## [1] 6
## [1] "AUC for fold 6 : 0.971698113207547"
## [1] "Missclassification rate for fold 6 : 0.0952380952380952"
## [1] 7
## [1] "AUC for fold 7 : 0.985416666666667"
## [1] "Missclassification rate for fold 7 : 0.0294117647058824"
## [1] 8
## [1] "AUC for fold 8 : 0.96360153256705"
## [1] "Missclassification rate for fold 8 : 0.0597014925373134"
## [1] 9
## [1] "AUC for fold 9 : 0.982456140350877"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] 10
## [1] "AUC for fold 10 : 0.943396226415094"
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

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

```
## [1] "Average of AUC: 0.966193921691327"
```

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

```
## [1] "Average of Miss: 0.0549998105019511"
```

```
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 0.9662 based on the resulting AUC of the 1 hidden layer, 3 units of Artificial Neural Network with a average missclassification probability of 0.0550.

5.5.3 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()

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 : 0.986607142857143"
## [1] "Missclassification rate for fold 1 : 0.05"
## [1] 2
## [1] "AUC for fold 2 : 0.982456140350877"
## [1] "Missclassification rate for fold 2 : 0.0161290322580645"
## [1] 3
## [1] "AUC for fold 3 : 0.811594202898551"
## [1] "Missclassification rate for fold 3 : 0.0769230769230769"
## [1] 4
## [1] "AUC for fold 4 : 1"
## [1] "Missclassification rate for fold 4 : 0.0192307692307692"
## [1] 5
## [1] "AUC for fold 5 : 0.839622641509434"
## [1] "Missclassification rate for fold 5 : 0.0819672131147541"
## [1] 6
## [1] "AUC for fold 6 : 0.983018867924528"
## [1] "Missclassification rate for fold 6 : 0.0317460317460317"
## [1] 7
```

```
## [1] "AUC for fold 7 : 0.985416666666667"
## [1] "Missclassification rate for fold 7 : 0.0588235294117647"
## [1] 8
## [1] "AUC for fold 8 : 0.957854406130268"
## [1] "Missclassification rate for fold 8 : 0.0746268656716418"
## [1] 9
## [1] "AUC for fold 9 : 0.925039872408293"
## [1] "Missclassification rate for fold 9 : 0.0441176470588235"
## [1] 10
## [1] "AUC for fold 10 : 0.744234800838574"
## [1] "Missclassification rate for fold 10 : 0.0806451612903226"
```

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

```
## [1] "Average of AUC: 0.921584474158434"
```

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

```
## [1] "Average of Miss: 0.0534209326705249"
```

```
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 0.9216 based on the resulting AUC of the 2 hidden layer, 3 units of Artificial Neural Network with a average missclassification probability of 0.0534.

5.5.4 Model 2: 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 laye
        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 : 0.995535714285714"
## [1] "Missclassification rate for fold 1 : 0.0166666666666667"
## [1] 2
## [1] "AUC for fold 2 : 1"
## [1] "Missclassification rate for fold 2 : 0"
## [1] 3
## [1] "AUC for fold 3 : 0.894927536231884"
## [1] "Missclassification rate for fold 3 : 0.0192307692307692"
## [1] 4
## [1] "AUC for fold 4 : 0.961702127659575"
## [1] "Missclassification rate for fold 4 : 0.0576923076923077"
## [1] 5
## [1] "AUC for fold 5 : 0.827830188679245"
## [1] "Missclassification rate for fold 5 : 0.0655737704918033"
## [1] 6
## [1] "AUC for fold 6 : 0.969811320754717"
## [1] "Missclassification rate for fold 6 : 0.0634920634920635"
## [1] 7
## [1] "AUC for fold 7 : 0.975"
## [1] "Missclassification rate for fold 7 : 0.0294117647058824"
## [1] 8
## [1] "AUC for fold 8 : 0.952107279693487"
## [1] "Missclassification rate for fold 8 : 0.0746268656716418"
## [1] 9
## [1] "AUC for fold 9 : 0.968102073365231"
## [1] "Missclassification rate for fold 9 : 0.0588235294117647"
## [1] 10
## [1] "AUC for fold 10 : 0.989517819706499"
## [1] "Missclassification rate for fold 10 : 0.032258064516129"
```

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

```
## [1] "Average of AUC: 0.953453406037635"
```

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

```
## [1] "Average of Miss: 0.0417775801879028"
```

```
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 0.95345 based on the resulting AUC of the 5 hidden layer, 4 units of Artificial Neural Network with a average missclassification probability of 0.04178.

Advantages of Artificial Neural Networks (ANN) classifier
1. ANN learning methods are quite robust to noise in the training data.
2. It is used where the fast evaluation of the learned target function required.
3. ANNs have the ability to learn and model non-linear and complex relationships.
4. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables.

Disadvantages of Artificial Neural Networks (ANN) classifier
1. Unexplained functioning of the network that is, when ANN gives a probing solution, it does not give a clue as to why and how.
2. Assurance of proper network structure: 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.

5.6 Model Comparison

```
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-
                       "MissClassification Rate"= Missclassification_Rate, "AUC"=AUC);
knitr::kable(Measures, align = "lcc")
```

Classifier.Method	MissClassification.Rate	AUC
LASSO Regularized Regression	0.041	0.969
RF	0.034	0.981
MARS	0.126	0.914
SVM- Linear	0.051	0.837
SVM- C value Linear	0.052	0.841
SVM- Radial	0.045	0.890
ANN-c(1,3)	0.055	0.966
ANN-c(2,3)	0.053	0.922
ANN-c(5,4)	0.042	0.953

Among all these 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 from the table.

6 Reference

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

2. Friedman JH (1991) Multivariate adaptive regression splines. The annals of statistics: 1-67.

3. Leathwick JR, Elith J, Hastie T (2006) Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions. Ecological modelling, 199(2): 188-196.

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