Project VII

Hortencia J. Hernandez* University of Texas at El Paso (UTEP)

9 December, 2022

Contents

*hjhernandez4@miners.utep.edu

1 Data Preparation

```
## [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
```

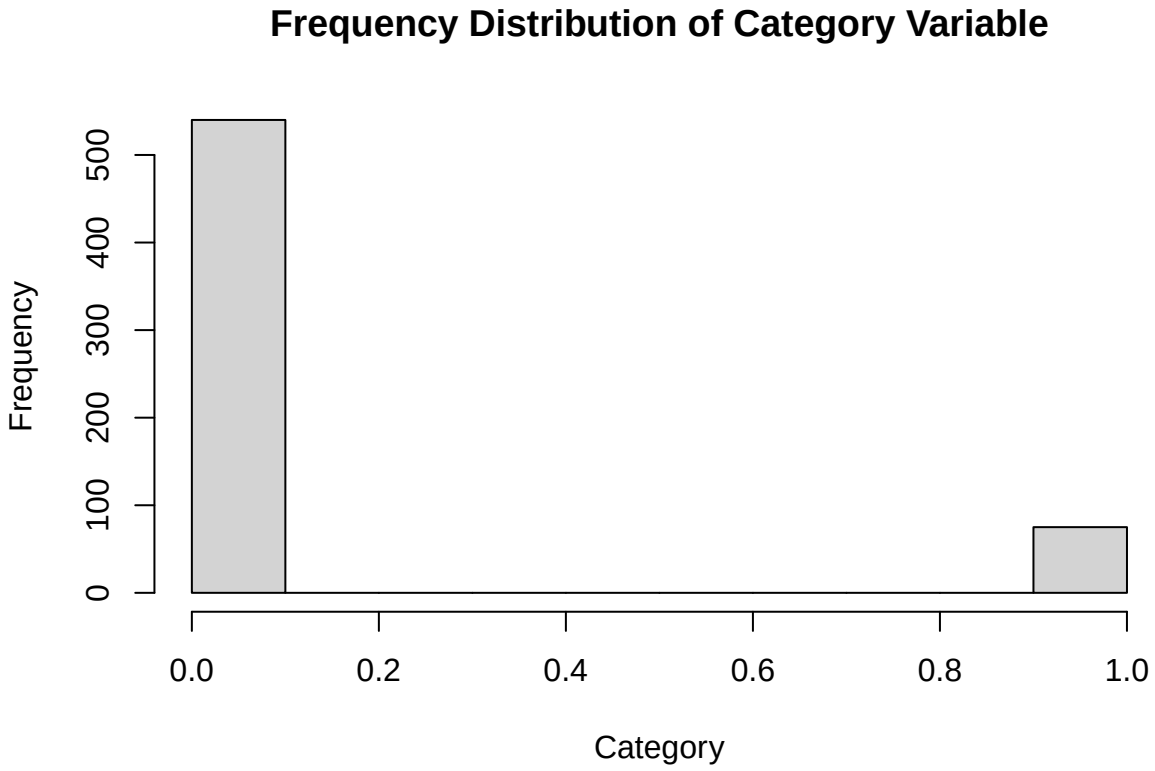
```
## [1] TRUE
```

1.1 (a) Modify target variable: Category

```
##
##          0=Blood Donor 0s=suspect Blood Donor            1=Hepatitis
##                    533                       7                      24
##             2=Fibrosis           3=Cirrhosis
##                     21                      30
```

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

1.2 (b) Frequency Distribution of Category

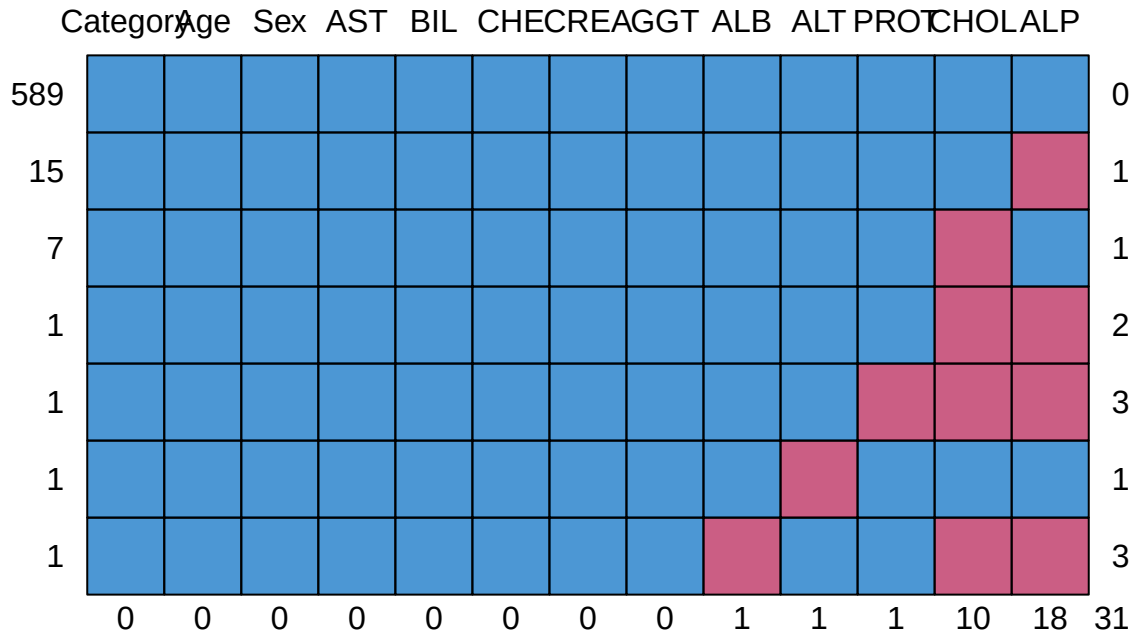


From (a) we can see the distribution that it is imbalanced classification problem as 87.8% is classified as a Blood Donor (0), where only 12.2% is not a Blood Donor. From the graph we can also see how unbalanced the Category variable is. There are 0 NAs thus we do not have to remove those variables.

1.3 (c) Missing Values

```
##    Category         Age         Sex         ALB         ALP         ALT
## 0.000000000 0.000000000 0.000000000 0.001626016 0.029268293 0.001626016
##         AST         BIL         CHE        CHOL        CREA         GGT
## 0.000000000 0.000000000 0.000000000 0.016260163 0.000000000 0.000000000
##        PROT
## 0.001626016
```

From the above table we can see that there are several predictor variables that have NAs: ALB, ALP, ALT, CHOL, and PROT. Thus we need to impute the missing values for those predictors




```
##     Category Age Sex AST BIL CHE CREA GGT ALB ALT PROT CHOL ALP
## 589        1   1   1   1   1   1    1   1   1   1    1    1   0
## 15         1   1   1   1   1   1    1   1   1   1    1    0   1
## 7          1   1   1   1   1   1    1   1   1   1    0    1   1
## 1          1   1   1   1   1   1    1   1   1   1    0    0   2
## 1          1   1   1   1   1   1    1   1   1   0    0    0   3
## 1          1   1   1   1   1   1    1   1   1   0    1    1   1
## 1          1   1   1   1   1   1    1   1   0   1    1    0   3
##            0   0   0   0   0   0    0   0   1   1    1   10  18 31
```

```
##
##  iter imp variable
##   1   1  ALB  ALP  ALT  CHOL  PROT
##   2   1  ALB  ALP  ALT  CHOL  PROT
##   3   1  ALB  ALP  ALT  CHOL  PROT
##   4   1  ALB  ALP  ALT  CHOL  PROT
##   5   1  ALB  ALP  ALT  CHOL  PROT
##   6   1  ALB  ALP  ALT  CHOL  PROT
##   7   1  ALB  ALP  ALT  CHOL  PROT
##   8   1  ALB  ALP  ALT  CHOL  PROT
##   9   1  ALB  ALP  ALT  CHOL  PROT
##   10   1  ALB  ALP  ALT  CHOL  PROT
```

```
##   11   1  ALB  ALP  ALT  CHOL  PROT
##   12   1  ALB  ALP  ALT  CHOL  PROT
##   13   1  ALB  ALP  ALT  CHOL  PROT
##   14   1  ALB  ALP  ALT  CHOL  PROT
##   15   1  ALB  ALP  ALT  CHOL  PROT
##   16   1  ALB  ALP  ALT  CHOL  PROT
##   17   1  ALB  ALP  ALT  CHOL  PROT
##   18   1  ALB  ALP  ALT  CHOL  PROT
##   19   1  ALB  ALP  ALT  CHOL  PROT
##   20   1  ALB  ALP  ALT  CHOL  PROT
##   21   1  ALB  ALP  ALT  CHOL  PROT
##   22   1  ALB  ALP  ALT  CHOL  PROT
##   23   1  ALB  ALP  ALT  CHOL  PROT
##   24   1  ALB  ALP  ALT  CHOL  PROT
##   25   1  ALB  ALP  ALT  CHOL  PROT
##   26   1  ALB  ALP  ALT  CHOL  PROT
##   27   1  ALB  ALP  ALT  CHOL  PROT
##   28   1  ALB  ALP  ALT  CHOL  PROT
##   29   1  ALB  ALP  ALT  CHOL  PROT
##   30   1  ALB  ALP  ALT  CHOL  PROT
##   31   1  ALB  ALP  ALT  CHOL  PROT
##   32   1  ALB  ALP  ALT  CHOL  PROT
##   33   1  ALB  ALP  ALT  CHOL  PROT
##   34   1  ALB  ALP  ALT  CHOL  PROT
##   35   1  ALB  ALP  ALT  CHOL  PROT
##   36   1  ALB  ALP  ALT  CHOL  PROT
##   37   1  ALB  ALP  ALT  CHOL  PROT
##   38   1  ALB  ALP  ALT  CHOL  PROT
##   39   1  ALB  ALP  ALT  CHOL  PROT
##   40   1  ALB  ALP  ALT  CHOL  PROT
##   41   1  ALB  ALP  ALT  CHOL  PROT
##   42   1  ALB  ALP  ALT  CHOL  PROT
##   43   1  ALB  ALP  ALT  CHOL  PROT
##   44   1  ALB  ALP  ALT  CHOL  PROT
##   45   1  ALB  ALP  ALT  CHOL  PROT
##   46   1  ALB  ALP  ALT  CHOL  PROT
##   47   1  ALB  ALP  ALT  CHOL  PROT
##   48   1  ALB  ALP  ALT  CHOL  PROT
##   49   1  ALB  ALP  ALT  CHOL  PROT
##   50   1  ALB  ALP  ALT  CHOL  PROT

## Warning: Number of logged events: 1

## Class: mids
## Number of multiple imputations:  1
## Imputation methods:
## Category      Age      Sex      ALB      ALP      ALT      AST      BIL
```

```
##       ""       ""       ""    "pmm"    "pmm"    "pmm"       ""       ""
##      CHE     CHOL     CREA      GGT     PROT
##       ""    "pmm"       ""       ""    "pmm"
## PredictorMatrix:
##          Category Age Sex ALB ALP ALT AST BIL CHE CHOL CREA GGT PROT
## Category        0   1   0   1   1   1   1   1   1    1    1   1    1
## Age             1   0   0   1   1   1   1   1   1    1    1   1    1
## Sex             1   1   0   1   1   1   1   1   1    1    1   1    1
## ALB             1   1   0   0   1   1   1   1   1    1    1   1    1
## ALP             1   1   0   1   0   1   1   1   1    1    1   1    1
## ALT             1   1   0   1   1   0   1   1   1    1    1   1    1
## Number of logged events:  1
##   it im dep     meth out
## 1  0  0     constant Sex

## [1] 615  13
```

After imputing with the package 'mice' we can see that there are no NAs anymore

```
## Category      Age      Sex      ALB      ALP      ALT      AST      BIL
##        0        0        0        0        0        0        0        0
##      CHE     CHOL     CREA      GGT     PROT
##        0        0        0        0        0
```

1.4 (d) Change Data Matrix into Numeric

In this section, we changed the data matrix into numeric by using the model.matrix() function.

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

2 EDA

2.1 (a) Range and Variations

The following table outputs the range of each predictor

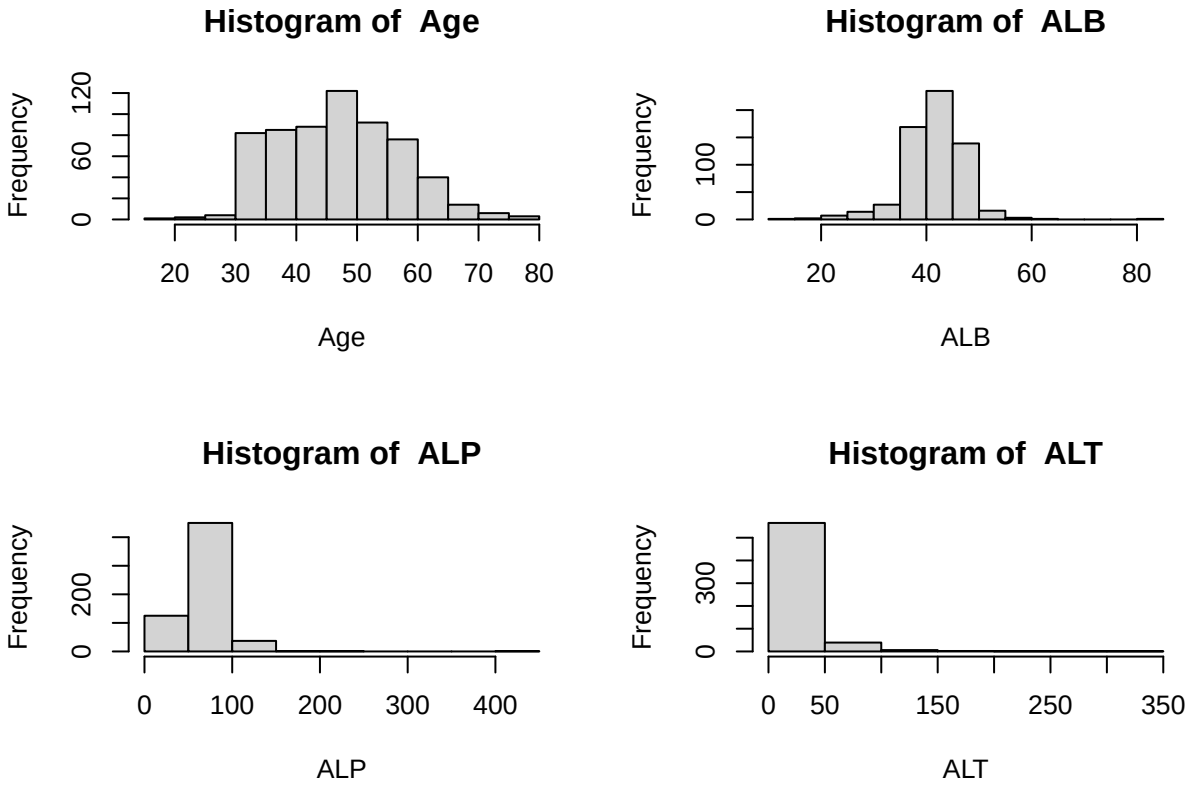
	Age	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
min	19	14.9	11.3	0.9	10.6	0.8	1.42	1.43	8	4.5	44.8
max	77	82.2	416.6	325.3	324	254	16.41	9.67	1079.1	650.9	90

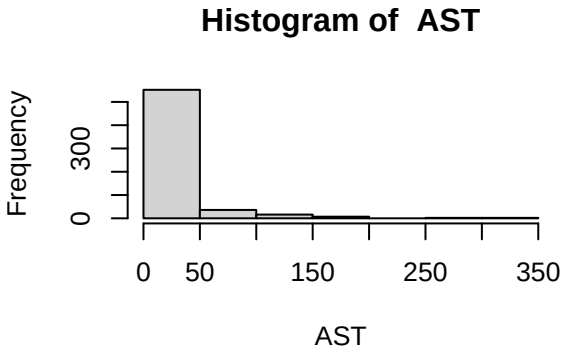
Table 1: Range of Predictors

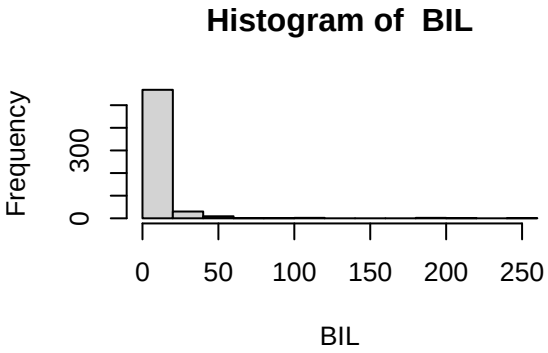
The following table outputs the Variance of each predictor

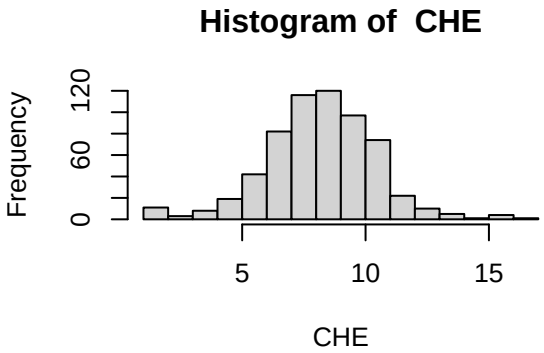
	Age	ALB	ALP	ALT	AST	BIL
Variance	101.1051455	33.3697046	673.0809658	647.7999697	1094.9937871	387.0328233
	CHE	CHOL	CREA	GGT	PROT	
Variance	4.864924	1.3143953	2475.6760562	2987.832709	30.3478338	

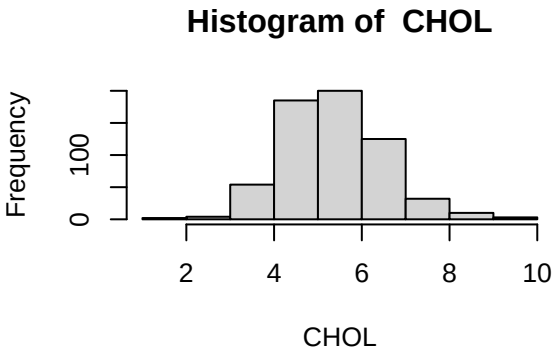
Table 2: Variance of Predictors

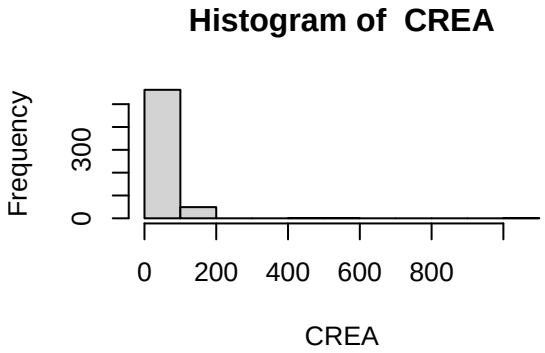


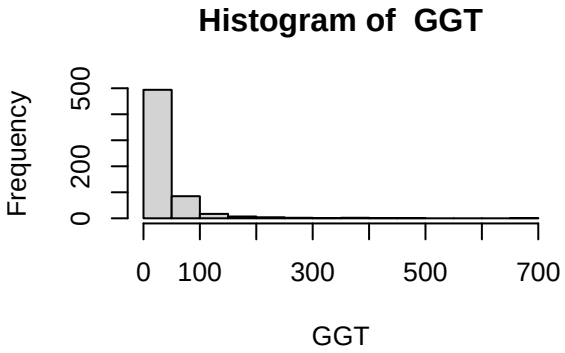

Histogram of AST
Frequency
0
300
0
50
150
250
350
AST

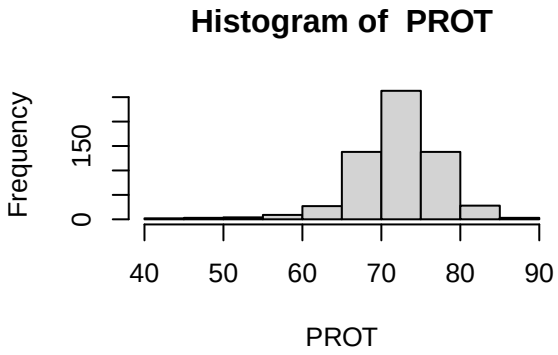

Histogram of BIL
Frequency
0
300
0
50
100
150
200
250
BIL


Histogram of CHE
Frequency
0
60
120
5
10
15
CHE


Histogram of CHOL
Frequency
0
100
2
4
6
8
10
CHOL



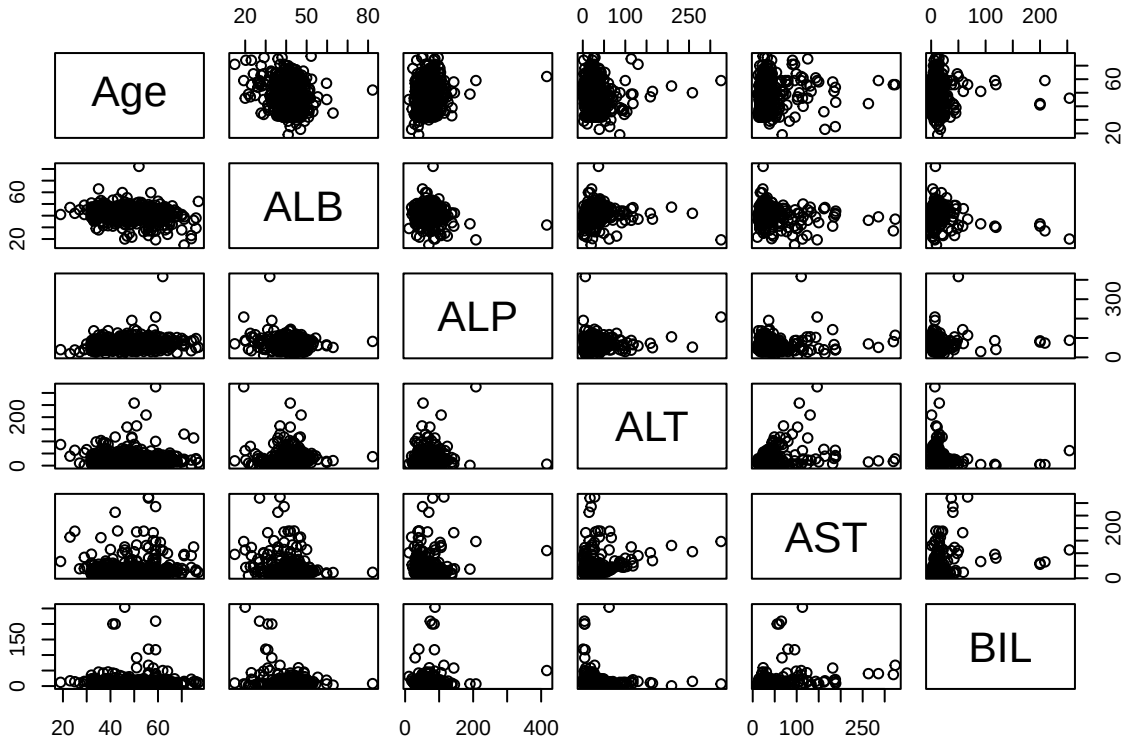


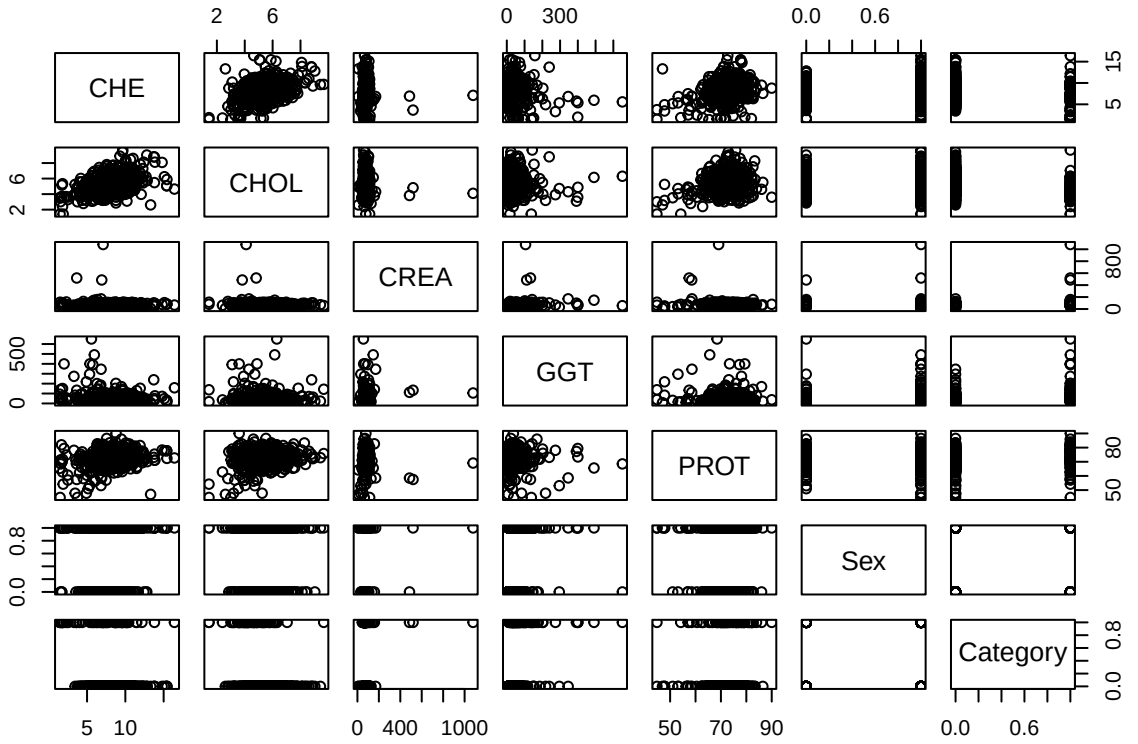


From the above plots we can see that the predictors: ALP, ALT, AST, BIL, CREA, and GGT are right skewed and PROT looks to be left skewed ish.

2.2 (b) Explore Association between Target and Predictors

```
##         Age         ALB         ALP         ALT         AST         BIL
##  0.03778132 -0.17811267 -0.10408920  0.08705973  0.62172398  0.39845142
##         CHE        CHOL        CREA         GGT        PROT         Sex
## -0.23078510 -0.27840208  0.13677183  0.43768040  0.06067504  0.07166335
##    Category
##  1.00000000
```


Age
ALB
ALP
ALT
AST
BIL
20 50 80
0 100 250
0 100 200
20 60
20 60
0 300
0 200
0 200
0 150
20 40 60
0 200 400
0 100 250



We can see that ALB, ALP, CHE, and CHOL have a negative correlation with the target variable (Category)

3 Outlier Detection

Goal: Can the method detect Hepatitis C patients as outliers based on what is learned from healthy blood donors?

The outlier detection I used was One-Class SVM for Novelty Detection.

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

```
## Number of Support Vectors:  73
##
##
##
##
## Number of Classes: 1

##
## pred    FALSE TRUE
##   FALSE    22   15
##   TRUE     53  525

## Loading required package: ggplot2

## Loading required package: lattice

## Confusion Matrix and Statistics
##
##
## pred    FALSE TRUE
##   FALSE    22   15
##   TRUE     53  525
##
##                Accuracy : 0.8894
##                  95% CI : (0.8619, 0.9131)
##     No Information Rate : 0.878
##     P-Value [Acc > NIR] : 0.2132
##
##                   Kappa : 0.3396
##
##  Mcnemar's Test P-Value : 7.226e-06
##
##             Sensitivity : 0.9722
##             Specificity : 0.2933
##          Pos Pred Value : 0.9083
##          Neg Pred Value : 0.5946
##              Prevalence : 0.8780
##          Detection Rate : 0.8537
##    Detection Prevalence : 0.9398
##       Balanced Accuracy : 0.6328
##
##        'Positive' Class : TRUE
##
```

4 Data Partitioning

For the predictive modeling we will use the following code for V-fold cross validation on misclassification errors, with V = 10, data partitioning for each.

```
#-------------------
# Data Partitioning
#-------------------

set.seed(3983)

V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;

id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)

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

}
```

5 Predictive Modeling

Here, extract the following:

- AUC under the ROC curve
- Predicted Dichotomous Outcomes

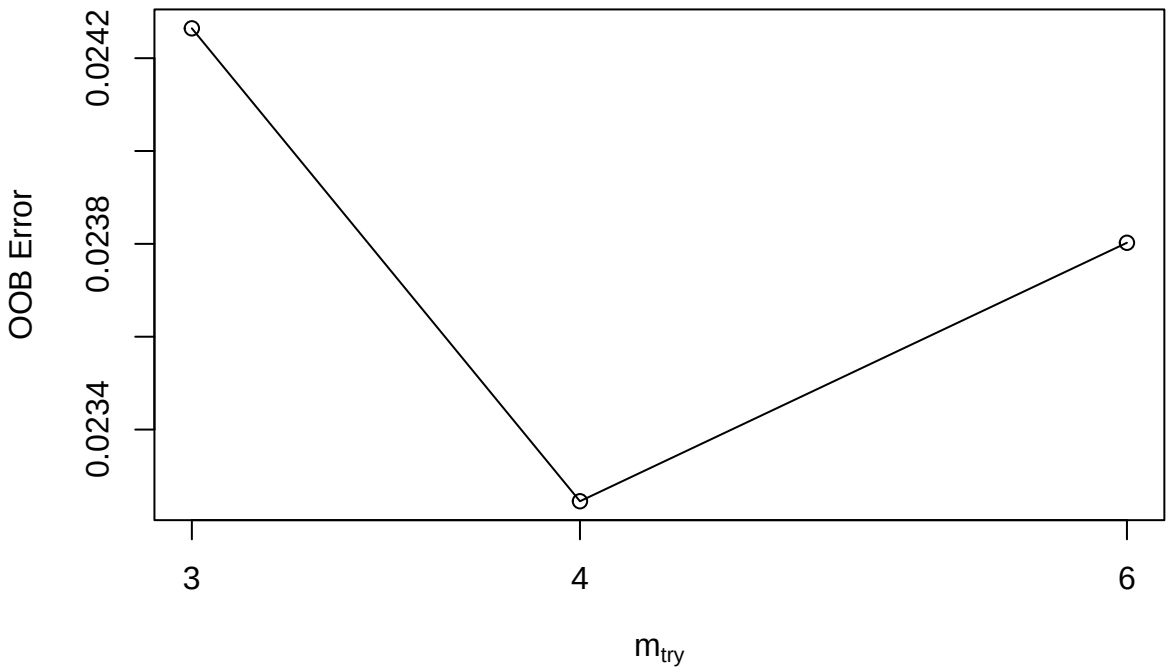
5.1 (a) Logistic Regression

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0004435 0.0125608 0.0732455 0.2474561 0.3918909 1.0000000
## [1] "AUC for fold 1 : 0.988095238095238"
## [1] "Misclassified rate for fold 1 : 0.0769230769230769"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000063 0.0079926 0.0343062 0.1724727 0.1084004 0.9999994
## [1] "AUC for fold 2 : 0.985119047619048"
## [1] "Misclassified rate for fold 2 : 0.0181818181818182"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

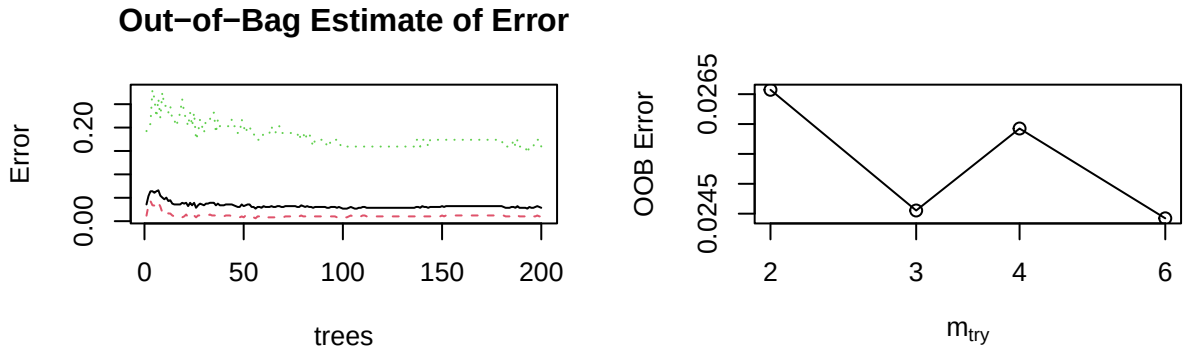
```
## 0.0003853 0.0054824 0.0252987 0.2134738 0.2730150 1.0000000
## [1] "AUC for fold 3 : 0.937406855439642"
## [1] "Misclassified rate for fold 3 : 0.0833333333333333"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000541 0.0087125 0.0264471 0.1408523 0.1212069 1.0000000
## [1] "AUC for fold 4 : 0.971938775510204"
## [1] "Misclassified rate for fold 4 : 0.0158730158730159"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0009182 0.0061923 0.0544311 0.2458130 0.3134922 1.0000000
## [1] "AUC for fold 5 : 1"
## [1] "Misclassified rate for fold 5 : 0.0212765957446809"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000024 0.0006774 0.0048341 0.1172399 0.0370916 1.0000000
## [1] "AUC for fold 6 : 0.968992248062015"
## [1] "Misclassified rate for fold 6 : 0.0434782608695652"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0003544 0.0123215 0.0351241 0.1300778 0.0962317 1.0000000
## [1] "AUC for fold 7 : 1"
## [1] "Misclassified rate for fold 7 : 0.0153846153846154"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000020 0.002283 0.013334 0.129726 0.099885 1.000000
## [1] "AUC for fold 8 : 0.895833333333333"
## [1] "Misclassified rate for fold 8 : 0.075"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0003316 0.0120928 0.0507401 0.1846934 0.1889478 0.9999956
## [1] "AUC for fold 9 : 1"
## [1] "Misclassified rate for fold 9 : 0.0508474576271186"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0007051 0.0075202 0.0281477 0.1425242 0.1055532 1.0000000
## [1] "AUC for fold 10 : 0.994897959183674"
## [1] "Misclassified rate for fold 10 : 0.0317460317460317"
```

5.2 (b) RF

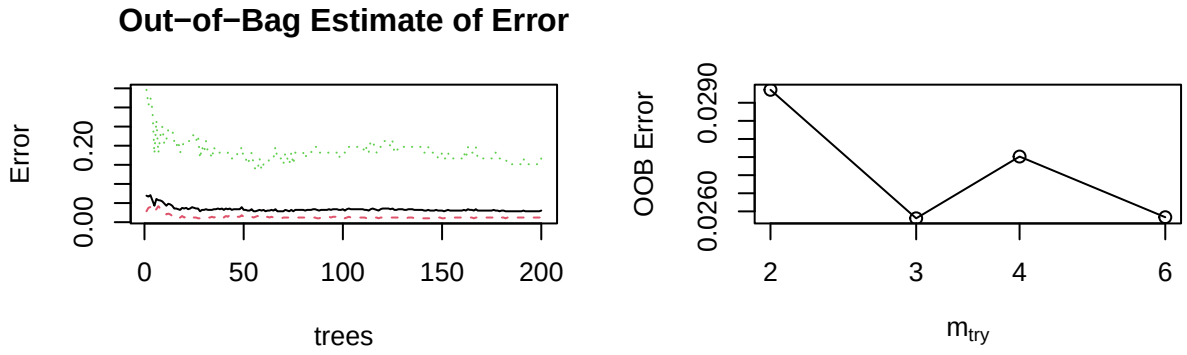
```
## mtry = 4  OOB error = 0.02324559
## Searching left ...
## mtry = 3     OOB error = 0.02426423
## -0.04382081 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02380234
## -0.02395055 0.01
```



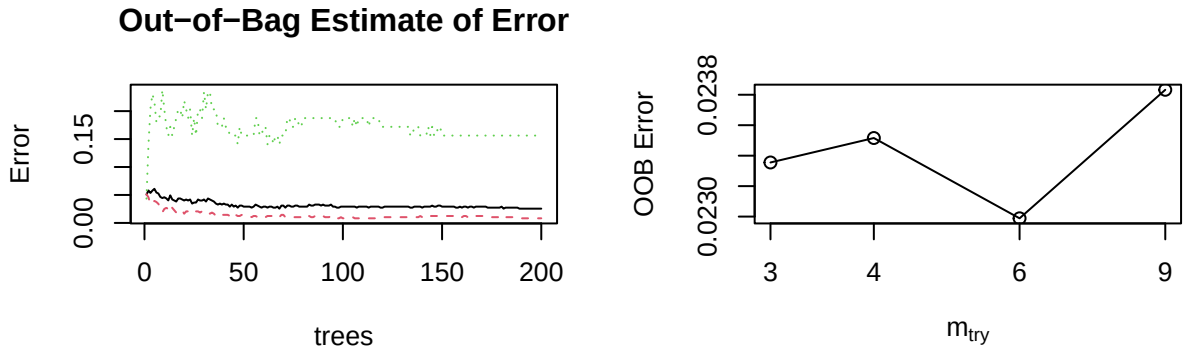

```
##  0  1
## 50  5
## [1] "AUC for fold 1 : 0.82312925170068"
## [1] "Misclassified rate for fold 1 : 0.206349206349206"
## mtry = 4  OOB error = 0.02592518
## Searching left ...
## mtry = 3     OOB error = 0.02455321
## 0.05292066 0.01
## mtry = 2     OOB error = 0.02657287
## -0.08225681 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02442267
## 0.0053166 0.01
```



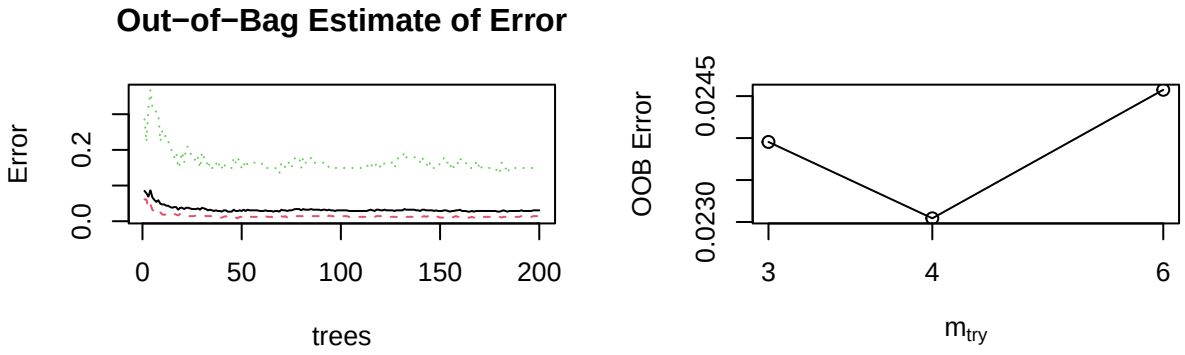

```
##  0  1
## 45  9
## [1] "AUC for fold 2 : 1"
## [1] "Misclassified rate for fold 2 : 0.253968253968254"
## mtry = 4  OOB error = 0.02751423
## Searching left ...
## mtry = 3     OOB error = 0.02580851
## 0.06199422 0.01
## mtry = 2     OOB error = 0.02935784
## -0.1375257 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02583826
## -0.001152541 0.01
```



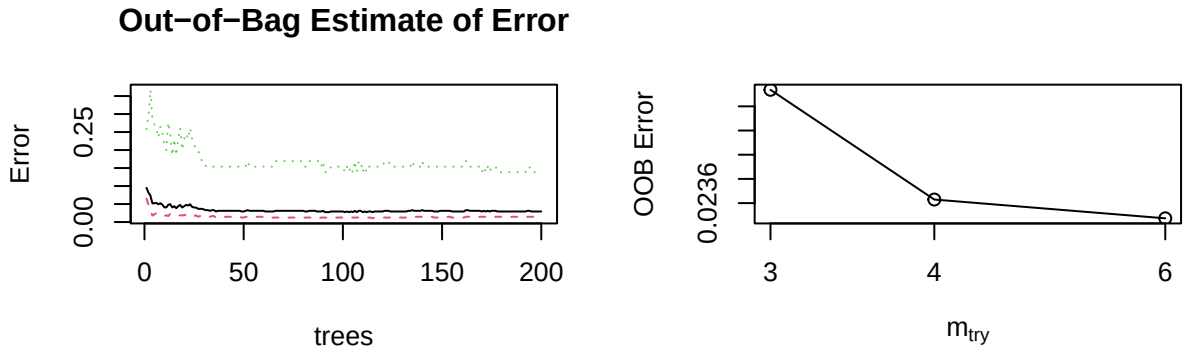

```
##  0  1
## 54 12
## [1] "AUC for fold 3 : 0.990909090909091"
## [1] "Misclassified rate for fold 3 : 0.0909090909090909"
## mtry = 4  OOB error = 0.02351579
## Searching left ...
## mtry = 3     OOB error = 0.02335528
## 0.006825513 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02298875
## 0.02241207 0.01
## mtry = 9     OOB error = 0.02383238
## -0.03669743 0.01
```



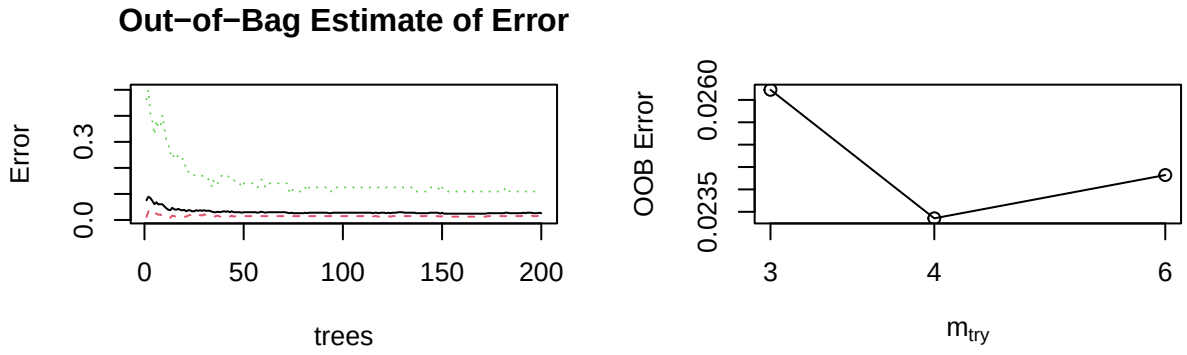

```
##  0  1
## 52  6
## [1] "AUC for fold 4 : 0.875"
## [1] "Misclassified rate for fold 4 : 0.206349206349206"
## mtry = 4  OOB error = 0.02304398
## Searching left ...
## mtry = 3     OOB error = 0.02395388
## -0.03948538 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02457508
## -0.06644272 0.01
```



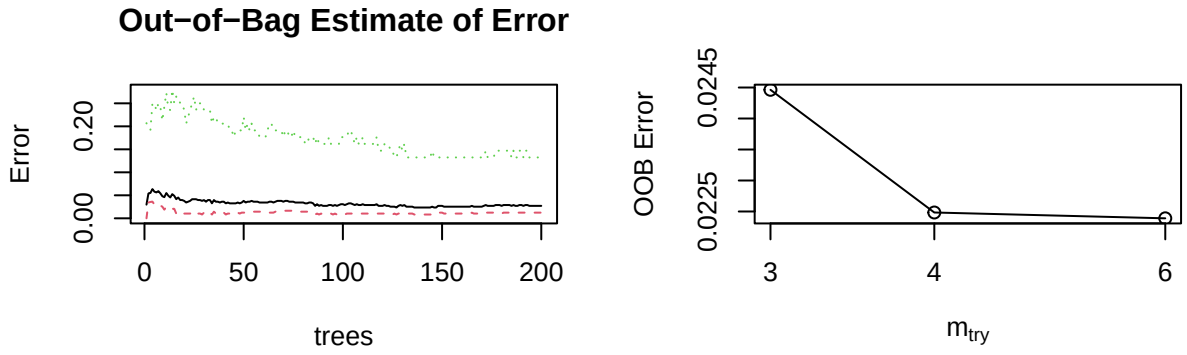

```
##  0  1
## 60 10
## [1] "AUC for fold 5 : 0.883333333333333"
## [1] "Misclassified rate for fold 5 : 0.157142857142857"
## mtry = 4  OOB error = 0.0236295
## Searching left ...
## mtry = 3     OOB error = 0.02453664
## -0.03838985 0.01
## Searching right ...
## mtry = 6     OOB error = 0.0234743
## 0.006568244 0.01
```



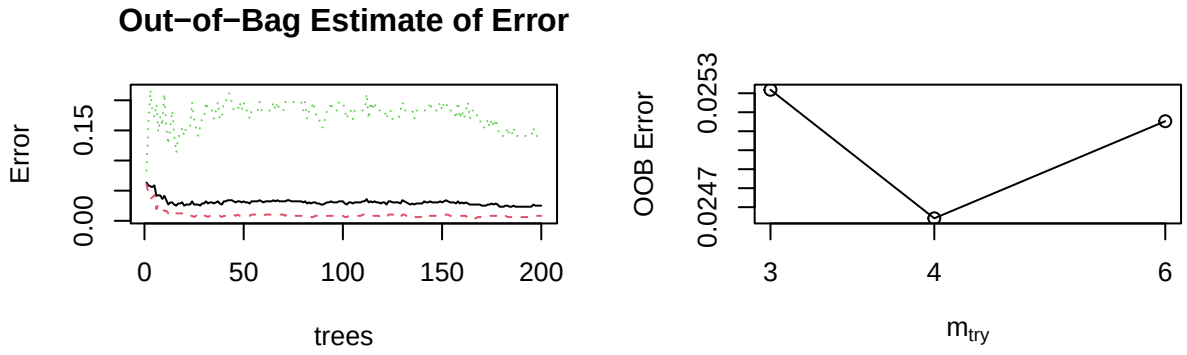

```
##  0  1
## 67 10
## [1] "AUC for fold 6 : 0.901515151515151"
## [1] "Misclassified rate for fold 6 : 0.233766233766234"
## mtry = 4  OOB error = 0.02335448
## Searching left ...
## mtry = 3     OOB error = 0.02622597
## -0.1229528 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02432111
## -0.04138949 0.01
```



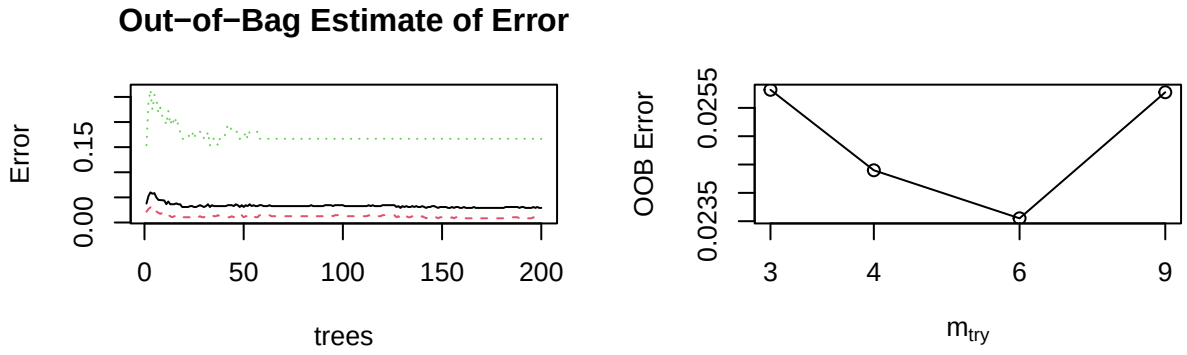

```
##  0  1
## 54  7
## [1] "AUC for fold 7 : 1"
## [1] "Misclassified rate for fold 7 : 0.0952380952380952"
## mtry = 4  OOB error = 0.02248361
## Searching left ...
## mtry = 3     OOB error = 0.024463
## -0.08803707 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02239011
## 0.004158576 0.01
```



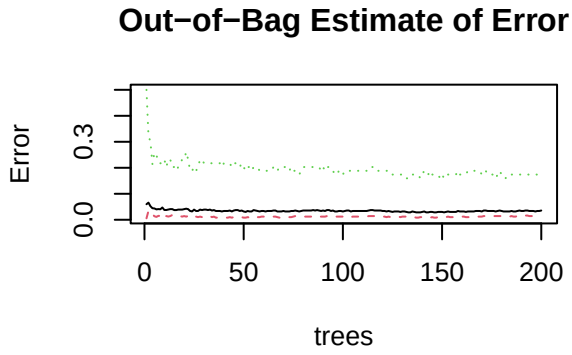

```
##  0  1
## 55  3
## [1] "AUC for fold 8 : 0.875"
## [1] "Misclassified rate for fold 8 : 0.142857142857143"
## mtry = 4  OOB error = 0.02464138
## Searching left ...
## mtry = 3     OOB error = 0.02531814
## -0.02746404 0.01
## Searching right ...
## mtry = 6     OOB error = 0.0251527
## -0.02075028 0.01
```




```
##  0  1
## 60  4
## [1] "AUC for fold 9 : 0.991803278688525"
## [1] "Misclassified rate for fold 9 : 0.09375"
## mtry = 4  OOB error = 0.02439857
## Searching left ...
## mtry = 3     OOB error = 0.02581889
## -0.05821344 0.01
## Searching right ...
## mtry = 6     OOB error = 0.02355099
## 0.03473888 0.01
## mtry = 9     OOB error = 0.02577565
## -0.09446153 0.01
```




```
##  0  1
## 47  5
## [1] "AUC for fold 10 : 0.916666666666667"
## [1] "Misclassified rate for fold 10 : 0.206349206349206"
```



5.3 (c) MARS

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi

##        1
##  Min.   :0.0000015
##  1st Qu.:0.0000467
##  Median :0.0003144
##  Mean   :0.1607031
##  3rd Qu.:0.0050631
##  Max.   :1.0000000
## [1] "AUC for fold 1 : 1"
## [1] "Misclassified rate for fold 1 : 0.158730158730159"
##        1
##  Min.   :0.0000215
##  1st Qu.:0.0004868
##  Median :0.0020120
```

```
##  Mean   :0.1438792
##  3rd Qu.:0.0139286
##  Max.   :0.9997056
## [1] "AUC for fold 2 : 0.990196078431373"
## [1] "Misclassified rate for fold 2 : 0.206349206349206"
##        1
##  Min.   :0.0005489
##  1st Qu.:0.0011335
##  Median :0.0031266
##  Mean   :0.1017605
##  3rd Qu.:0.0077857
##  Max.   :0.9999722
## [1] "AUC for fold 3 : 1"
## [1] "Misclassified rate for fold 3 : 0.111111111111111"
##        1
##  Min.   :0.0000159
##  1st Qu.:0.0001354
##  Median :0.0005318
##  Mean   :0.1205333
##  3rd Qu.:0.0060693
##  Max.   :0.9999994
## [1] "AUC for fold 4 : 1"
## [1] "Misclassified rate for fold 4 : 0.222222222222222"
##        1
##  Min.   :0.0005213
##  1st Qu.:0.0014940
##  Median :0.0040272
##  Mean   :0.1351632
##  3rd Qu.:0.0217791
##  Max.   :1.0000000
## [1] "AUC for fold 5 : 1"
## [1] "Misclassified rate for fold 5 : 0.03125"
##        1
##  Min.   :0.0000000
##  1st Qu.:0.0002343
##  Median :0.0020226
##  Mean   :0.1522152
##  3rd Qu.:0.0124528
##  Max.   :0.9999763
## [1] "AUC for fold 6 : 1"
## [1] "Misclassified rate for fold 6 : 0.0769230769230769"
##        1
##  Min.   :0.0000099
##  1st Qu.:0.0001057
##  Median :0.0002626
##  Mean   :0.0949866
##  3rd Qu.:0.0039489
##  Max.   :0.9999630
```

```
## [1] "AUC for fold 7 : 0.913746630727763"
## [1] "Misclassified rate for fold 7 : 0.126984126984127"
##        1
##  Min.   :0.0000122
##  1st Qu.:0.0001467
##  Median :0.0008077
##  Mean   :0.1333177
##  3rd Qu.:0.0270018
##  Max.   :0.9999663
## [1] "AUC for fold 8 : 0.995283018867924"
## [1] "Misclassified rate for fold 8 : 0.111111111111111"
##        1
##  Min.   :0.0000023
##  1st Qu.:0.0000888
##  Median :0.0003959
##  Mean   :0.0801083
##  3rd Qu.:0.0028294
##  Max.   :0.9999982
## [1] "AUC for fold 9 : 0.967272727272727"
## [1] "Misclassified rate for fold 9 : 0.0952380952380952"
##        1
##  Min.   :0.0000066
##  1st Qu.:0.0001790
##  Median :0.0008247
##  Mean   :0.1090305
##  3rd Qu.:0.0050073
##  Max.   :0.9999977
## [1] "AUC for fold 10 : 0.998046875"
## [1] "Misclassified rate for fold 10 : 0.208333333333333"
```

5.4 (d) ANN

```
##        V1
##  Min.   :0.01076
##  1st Qu.:0.01076
##  Median :0.01076
##  Mean   :0.11524
##  3rd Qu.:0.01076
##  Max.   :0.97804
##        V1
##  Min.   :0.07021
##  1st Qu.:0.07021
##  Median :0.07021
##  Mean   :0.10614
##  3rd Qu.:0.07021
##  Max.   :0.86692
## [1] "AUC(1) for fold 1 : 0.995833333333333"
```

```
## [1] "AUC(2) for fold 1 : 0.80625"
## [1] "Misclassified rate for fold 1 : 0.132352941176471"
##        V1
##  Min.   :-0.0006697
##  1st Qu.: 0.0023388
##  Median : 0.0023388
##  Mean   : 0.1255244
##  3rd Qu.: 0.0023488
##  Max.   : 1.0171927
##        V1
##  Min.   :-0.18211
##  1st Qu.: 0.01608
##  Median : 0.01608
##  Mean   : 0.11887
##  3rd Qu.: 0.01608
##  Max.   : 1.02882
## [1] "AUC(1) for fold 2 : 1"
## [1] "AUC(2) for fold 2 : 0.984375"
## [1] "Misclassified rate for fold 2 : 0.208333333333333"
##        V1
##  Min.   :0.02041
##  1st Qu.:0.02041
##  Median :0.02041
##  Mean   :0.06946
##  3rd Qu.:0.02041
##  Max.   :0.88699
##        V1
##  Min.   :-0.20906
##  1st Qu.:-0.04714
##  Median : 0.04400
##  Mean   : 0.04564
##  3rd Qu.: 0.04400
##  Max.   : 0.85135
## [1] "AUC(1) for fold 3 : 0.941176470588235"
## [1] "AUC(2) for fold 3 : 0.470588235294118"
## [1] "Misclassified rate for fold 3 : 0.142857142857143"
##        V1
##  Min.   :-0.004045
##  1st Qu.: 0.012915
##  Median : 0.012915
##  Mean   : 0.106746
##  3rd Qu.: 0.012915
##  Max.   : 1.000628
##        V1
##  Min.   :0.03265
##  1st Qu.:0.03265
##  Median :0.03265
##  Mean   :0.09894
```

```
##  3rd Qu.:0.03265
##  Max.   :0.96308
## [1] "AUC(1) for fold 4 : 0.973684210526316"
## [1] "AUC(2) for fold 4 : 0.864035087719298"
## [1] "Misclassified rate for fold 4 : 0.0476190476190476"
##        V1
##  Min.   :0.01072
##  1st Qu.:0.01072
##  Median :0.01072
##  Mean   :0.13675
##  3rd Qu.:0.01072
##  Max.   :1.00199
##        V1
##  Min.   :-0.15169
##  1st Qu.: 0.01503
##  Median : 0.01503
##  Mean   : 0.10218
##  3rd Qu.: 0.01503
##  Max.   : 0.92918
## [1] "AUC(1) for fold 5 : 0.911007025761124"
## [1] "AUC(2) for fold 5 : 0.845433255269321"
## [1] "Misclassified rate for fold 5 : 0.147058823529412"
##        V1
##  Min.   :0.008086
##  1st Qu.:0.008086
##  Median :0.008086
##  Mean   :0.129725
##  3rd Qu.:0.008086
##  Max.   :1.007397
##        V1
##  Min.   :0.1147
##  1st Qu.:0.1147
##  Median :0.1147
##  Mean   :0.1272
##  3rd Qu.:0.1147
##  Max.   :0.8138
## [1] "AUC(1) for fold 6 : 1"
## [1] "AUC(2) for fold 6 : 0.571428571428571"
## [1] "Misclassified rate for fold 6 : 0.222222222222222"
##        V1
##  Min.   :-0.03775
##  1st Qu.: 0.01200
##  Median : 0.01200
##  Mean   : 0.14658
##  3rd Qu.: 0.01200
##  Max.   : 1.02264
##        V1
##  Min.   :0.01329
```

```
##  1st Qu.:0.01329
##  Median :0.01329
##  Mean   :0.08523
##  3rd Qu.:0.01329
##  Max.   :0.99203
## [1] "AUC(1) for fold 7 : 1"
## [1] "AUC(2) for fold 7 : 0.757142857142857"
## [1] "Misclassified rate for fold 7 : 0.222222222222222"
##        V1
##  Min.   :0.02202
##  1st Qu.:0.02202
##  Median :0.02202
##  Mean   :0.08557
##  3rd Qu.:0.02202
##  Max.   :0.99245
##        V1
##  Min.   :0.02605
##  1st Qu.:0.02605
##  Median :0.02605
##  Mean   :0.09027
##  3rd Qu.:0.02605
##  Max.   :0.98786
## [1] "AUC(1) for fold 8 : 0.783018867924528"
## [1] "AUC(2) for fold 8 : 1"
## [1] "Misclassified rate for fold 8 : 0.142857142857143"
##        V1
##  Min.   :0.01602
##  1st Qu.:0.01602
##  Median :0.01602
##  Mean   :0.20769
##  3rd Qu.:0.01602
##  Max.   :0.94723
##        V1
##  Min.   :0.05485
##  1st Qu.:0.05485
##  Median :0.05485
##  Mean   :0.11678
##  3rd Qu.:0.05485
##  Max.   :0.74978
## [1] "AUC(1) for fold 9 : 0.987412587412587"
## [1] "AUC(2) for fold 9 : 0.713986013986014"
## [1] "Misclassified rate for fold 9 : 0.117647058823529"
##        V1
##  Min.   :-0.0003106
##  1st Qu.: 0.0084811
##  Median : 0.0084811
##  Mean   : 0.2000719
##  3rd Qu.: 0.0085214
```

```
##  Max.   : 1.1172314
##        V1
##  Min.   :0.008358
##  1st Qu.:0.008358
##  Median :0.008358
##  Mean   :0.145945
##  3rd Qu.:0.008358
##  Max.   :1.205628
## [1] "AUC(1) for fold 10 : 0.871794871794872"
## [1] "AUC(2) for fold 10 : 0.958119658119658"
## [1] "Misclassified rate for fold 10 : 0.285714285714286"
```

5.5 (e) SVM

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.02994 -0.02803  0.02126  0.08876  0.02986  1.02992
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.08065 0.00000 1.00000
## [1] "AUC(1) for fold 1 : 1"
## [1] "AUC(2) for fold 1 : 0.916666666666667"
## [1] "Misclassified rate for fold 1 : 0.0476190476190476"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.04103 -0.02995  0.01079  0.08426  0.03040  1.03040
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1139  0.0000  1.0000
## [1] "AUC(1) for fold 2 : 1"
## [1] "AUC(2) for fold 2 : 0.85387323943662"
## [1] "Misclassified rate for fold 2 : 0.189873417721519"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.03444 -0.03421  0.03414  0.12649  0.03437  1.03446
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1333  0.0000  1.0000
## [1] "AUC(1) for fold 3 : 1"
## [1] "AUC(2) for fold 3 : 0.927884615384615"
## [1] "Misclassified rate for fold 3 : 0.142857142857143"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.121288 -0.023270  0.007131  0.086860  0.030630  1.030558
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1633  0.0000  1.0000
## [1] "AUC(1) for fold 4 : 1"
## [1] "AUC(2) for fold 4 : 0.965909090909091"
## [1] "Misclassified rate for fold 4 : 0.190476190476190"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.04856 -0.02972  0.02053  0.13726  0.03599  1.03586
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1493  0.0000  1.0000
## [1] "AUC(1) for fold 5 : 1"
```

```
## [1] "AUC(2) for fold 5 : 0.941228070175439"
## [1] "Misclassified rate for fold 5 : 0.0746268656716418"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.036172 -0.009832  0.035182  0.129052  0.035683  1.035671
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1273  0.0000  1.0000
## [1] "AUC(1) for fold 6 : 1"
## [1] "AUC(2) for fold 6 : 0.86436170212766"
## [1] "Misclassified rate for fold 6 : 0.238095238095238"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.032115 -0.028990  0.008308  0.113516  0.032048  1.032119
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1132  0.0000  1.0000
## [1] "AUC(1) for fold 7 : 1"
## [1] "AUC(2) for fold 7 : 1"
## [1] "Misclassified rate for fold 7 : 0.206349206349206"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.06497 -0.01883  0.03142  0.17261  0.03926  1.03929
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0     0.0     0.2     0.0     1.0
## [1] "AUC(1) for fold 8 : 1"
## [1] "AUC(2) for fold 8 : 0.943994601889339"
## [1] "Misclassified rate for fold 8 : 0.114285714285714"
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.03186 -0.02171  0.03163  0.11337  0.03176  0.96844
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.09259 0.00000 1.00000
## [1] "AUC(1) for fold 9 : 1"
## [1] "AUC(2) for fold 9 : 0.916666666666667"
## [1] "Misclassified rate for fold 9 : 0.206349206349206"
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.026789 -0.026569  0.006657  0.073664  0.026656  0.973392
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.1061  0.0000  1.0000
## [1] "AUC(1) for fold 10 : 1"
## [1] "AUC(2) for fold 10 : 0.983606557377049"
## [1] "Misclassified rate for fold 10 : 0.121212121212121"
```

5.5.1 Advantages and Disadvantages

SVM finds the optimal separating hyperplane that maximizes the margin thus, by the following table, and by the misclassified rate (since it is below .3) we can say that it is the best prediction model to used. However, logistic regression has the some of the smallest misclassified rates.

From the table, we can see that the second way I did ANN has the lowest average AUC values, thus maybe SVM would be the best way to predict the model.

	Log Reg.	RF	MARS	ANN(1)	ANN(2)	SVM(1)	SVM(2)
Avg AUC	0.9742283	0.9257357	0.9864545	0.9463927	0.7971359	1	1

Table 3: Average AUC values

6 Appendix: All Code Used in This Report

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


table(dat$Category, useNA = "ifany") #See variables
for(i in 1:nrow(dat)){
  if(dat[i, 1] == "0=Blood Donor" || dat[i, 1] == "0s=suspect Blood Donor"){
   dat[i, 1]  = 0
  } else if(is.na(dat[i, 1])){
    print(i) #Prints the location where there is an NA/missing value
  }else
    dat[i, 1] = 1
}

#Check that it Modified
table(dat$Category, useNA = "ifany")
dat$Category <- as.integer(dat$Category)
#dat$Category <- as.numeric(dat$Category) #make sure it is numericf
hist(dat$Category, main = "Frequency Distribution of Category Variable", xlab = "Category")
#mean(is.na(dat$Category))
# -did not want it outputted it is within the text.

# Missing Data
colMeans(is.na(dat)) #Check to see if there are any NAs



suppressPackageStartupMessages(library(mice))
md.pattern(dat)
dat.mice <- mice(dat, m = 1, maxit = 50, method = 'pmm', seed = 500)
summary(dat.mice)

dat.imputed <- complete(dat.mice, 1)
dim(dat.imputed)


# Check for no more NAs
```

```r
colMeans(is.na(dat.imputed))


X <- as.data.frame(model.matrix(Category~. -Sex + factor(Sex), data = dat.imputed))
X$"(Intercept)" <- NULL
#names(X)


X.scale <- scale(X)
dat.1 <- data.frame(cbind(X, Category = dat$Category))
names(dat.1)[12] <- "Sex"
#names(dat.1)
head(dat.1)


#-------------------------
# Range of Each Predictor
#-------------------------
n = ncol(dat.1) -2 #since Sex and Category is only 0 or 1

for(i in 1:n){
  colnames(dat.1[i])
  range(dat.1[i])

}


#----------------------------
# Variance of Each Predictor
#----------------------------

#diag(cov(dat.1))[1] #The variance of each predictor


par(mfrow = c(2,2))
for(i in 1:n){
  hist(dat.1[,i], xlab = colnames(dat.1)[i], main = paste("Histogram of ", colnames(dat.1)[i]))
}

#----
# EDA
#-----


# Correlation

dat.Cor <- cor(dat.1)
```

```
dat.Cor[,13]

pairs(dat.1[,1:6])
pairs(dat.1[,7:13])

#par(mfrow=c(2,1))

#for(j in 1:n){
#  for(i in 1:n){
#   plot(dat.1[,j], dat.1[,i+1],
#     col = factor(dat.1$Category), xlab = colnames(dat.1)[j], ylab = colnames(dat.1)[i+1])
#  }
#}
#--------------------
# Outlier Detection
#--------------------

library(e1071)

# Train the Model w/ Data
d.train <- subset(dat.1, Category = 0)

x <- subset(d.train, select = -Category);  y <- d.train$Category
p <- NCOL(x)
fit.OneClassSVM <- svm(x, y=NULL, type="one-classification", nu=0.02, # nu - OC-SVM TUNING PAR
    kernel="radial", gamma=1/p)  # gamma - PARAMETER IN RBF KERNEL
summary(fit.OneClassSVM)

# Apply to Predict Data of Hep C

pred <- predict(fit.OneClassSVM, subset(dat.1, select=-Category));
tab <- table(pred, as.character(dat.1$Category)=='0')
tab
library(caret)
confusionMatrix(tab,positive='TRUE')

#--------------------
# Data Partitioning
#--------------------

set.seed(3983)

V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;
```

```
id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)

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

}


suppressWarnings({
#----------------------
# Logistic Regression
#----------------------

suppressPackageStartupMessages(library(glmnet))
suppressPackageStartupMessages(library(verification))

set.seed(39832)

V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;

id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)


miscla = c(1:V)
err.log = c(1:V)



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

  X = model.matrix(as.factor(Category)~., data = train.v)
  y = factor(train.v$Category)
  fit.log <- glmnet(x = X, y = y, family="binomial",
                    alpha=1, lambda.min = 1e-4, nlambda = 20, standardize=T,
                    thresh = 1e-07, maxit=1000)
```

```
  CV = cv.glmnet(x = X, y = y, family = "binomial", alpha = 1,
                 lambda.min = 1e-4, nlambda = 20, standardize = T,
                 thresh = 1e-07, maxit = 1000)



#--------------------------
# Optimal Tuning Parameter
#--------------------------

  lambda = CV$lambda.1se; #best lambda

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

  fit.final = glm(Category~., family = "binomial", data = train.v)
  yobs = test.v$Category
  X.test = dat.1[id.fold==v, ]-1
  pred.glm = predict(fit.final, newdata = X.test, type = "response")
  print(summary(pred.glm))
  # AUC/ROC
  mod = roc.area(yobs, pred.glm)$A
  err.log[v] = mod
  print(paste("AUC for fold", v, ":", err.log[v]))
  pred.rate = ifelse(pred.glm > 0.5, 1, 0); pred.rate # predicted 0/1's
  missed.rate <- mean(yobs != pred.rate)
  miscla[v] = missed.rate
  print(paste("Misclassified rate for fold", v,
  ":",miscla[v]))

}


})

suppressWarnings({
#-----
# RF
#-----


suppressPackageStartupMessages(library(randomForest))


set.seed(39833)
```

```
V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;

id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)

err.rf = c(1:V)

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

  b.mtry <- tuneRF(train.v[-13],train.v$Category, ntreeTry=100,
                   stepFactor=1.5,improve=0.01, trace=TRUE,
                   plot=TRUE, dobest=FALSE)

  fit.rf=randomForest(factor(train.v$Category) ~ . , data = train.v , mtry= 4,
                      importance=TRUE, proximity=TRUE, ntree=200,
                      keep.forest=TRUE, oob.prox=FALSE)
  #fit.rf

  # rf plot and variable importance and Partial Depedence Plot

  par(mfrow = c(2,2))
  plot(fit.rf, main="Out-of-Bag Estimate of Error")
  round(importance(fit.rf), 2)
  #par(mfrow=c(2, 2), mar=rep(4,4));
  #partialPlot(fit.rf, pred.data=train.v, x.var=ALB, rug=TRUE)
  #partialPlot(fit.rf, pred.data=train.v, x.var=ALP, rug=TRUE)
  #partialPlot(fit.rf, pred.data=train.v, x.var=CHE, rug=TRUE)
  #partialPlot(fit.rf, pred.data=train.v, x.var=CHOL, rug=TRUE)

  yhat.rf <- predict(fit.rf, newdata=test.v)
  print(summary(yhat.rf))
  AUC <- roc.area(obs=yobs, pred=as.numeric(yhat.rf))$A

  err.rf[v] = AUC
  print(paste("AUC for fold", v, ":", err.rf[v]))
  pred.rate = ifelse(pred.glm > 0.5, 1, 0); pred.rate # predicted 0/1's
  missed.rate <- mean(yobs != pred.rate)
  miscla[v] = missed.rate
  print(paste("Misclassified rate for fold", v,
  ":",miscla[v]))
```

```
}

})

suppressWarnings({
#------
# MARS
#------
suppressPackageStartupMessages(library(earth))
suppressPackageStartupMessages(require(dplyr))


set.seed(39834)

V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;

id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)

err.mars = c(1:V)

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

  fit.mars1 <- earth(factor(Category) ~ ., data = train.v, degree=1,
                     glm=list(family=binomial(link = "logit")),
                     pmethod="cv", nfold=10)

  #---------------------
  # Variable Importance
  #---------------------
  library(vip)
  library(ggplot2)
  vip(fit.mars1, num_features = 10) + ggtitle("GCV")


  #------------------------
  # ROC and AUC Prediction
  #------------------------
```

```
  pred.mars <- predict(fit.mars1, newdata= test.v, type="response")
  print(summary(pred.mars))
  AUC <- roc.area(obs=yobs, pred=as.numeric(pred.mars))$A

  err.mars[v] = AUC
  print(paste("AUC for fold", v, ":", err.mars[v]))
  pred.rate = ifelse(pred.glm > 0.5, 1, 0); pred.rate # predicted 0/1's
  missed.rate <- mean(yobs != pred.rate)
  miscla[v] = missed.rate
  print(paste("Misclassified rate for fold", v,
  ":",miscla[v]))
}
})

suppressWarnings({
#------
# ANN
#------
set.seed(39835)

suppressPackageStartupMessages(library(neuralnet))


V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;

id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)

err.ann1 = c(1:V)
err.ann2 = c(1:V)

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

  fit1 <- neuralnet(Category~., data=train.v, hidden=c(5,3), rep = 1,
        threshold = 0.05, stepmax = 1e+05,
        algorithm = "rprop-", err.fct = "sse", act.fct = "logistic",
        linear.output=TRUE)
    fit2 <- neuralnet(Category~., data=train.v, hidden=8, rep = 1,
        threshold = 0.05, stepmax = 1e+05,
            algorithm = "rprop+",  err.fct = "sse", act.fct = "tanh",
```

```
        linear.output=TRUE)
    yhat1 <- as.vector(compute(fit1, covariate=test.v[, -13])$net.result)
    yhat2 <- as.vector(compute(fit2, covariate=test.v[, -13])$net.result)



  # PARAMETERS IN TWO MODELS
  c(length(fit1$result.matrix), length(fit2$result.matrix))

  yhat <- compute(fit1, covariate=train.v)$net.result


  # PREDICTION

  pred.fit1 <- compute(fit1, covariate=test.v)$net.result
  pred.fit2 <- compute(fit2, covariate=test.v)$net.result

  print(summary(pred.fit1))
  print(summary(pred.fit2) )

    #-------------------------
  # ROC and AUC Prediction
  #-------------------------

  AUC.1 <- roc.area(obs=yobs, pred=as.numeric(pred.fit1))$A
  AUC.2 <- roc.area(obs=yobs, pred=as.numeric(pred.fit2))$A

  err.ann1[v] = AUC.1
  err.ann2[v] = AUC.2
  print(paste("AUC(1) for fold", v, ":", err.ann1[v]))
  print(paste("AUC(2) for fold", v, ":", err.ann2[v]))
  pred.rate = ifelse(pred.glm > 0.5, 1, 0); pred.rate # predicted 0/1's
  missed.rate <- mean(yobs != pred.rate)
  miscla[v] = missed.rate
  print(paste("Misclassified rate for fold", v,
  ":",miscla[v]))

}


})
suppressWarnings({
#------
# SVM
#------
set.seed(39836)
```

```
suppressPackageStartupMessages(library("e1071"))
suppressPackageStartupMessages(library(MASS))
suppressPackageStartupMessages(library("kernlab")
                               )
V <- 10
n <- NROW(dat.1);
n0 <- sum(dat.1$Category == 0);
n1 <- n - n0;

id.fold <- 1:n
id.fold[dat.1$Category==0] <- sample(x = 1:V, size = n0, replace = TRUE)
id.fold[dat.1$Category==1] <- sample(x = 1:V, size = n1, replace = TRUE)

err.svm1 = c(1:V)
err.svm2 = c(1:V)

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

  fit.0 <- svm(Category ~ ., data=train.v, method = "C-classification",
               kernel = "radial", cost = 10, gamma = 0.1, scale = TRUE)

  gammas <- 10^(-5:1)
  Cs <-  10^(-2:2)
  tobj <- tune.svm(Category ~ ., data = train.v,
                   gamma = gammas, cost =Cs,
                   nrepeat=2, scale = TRUE,
                   tunecontrol = tune.control(sampling = "cross",cross=10))
  bestGamma <- tobj$best.parameters[[1]]
  bestC <- tobj$best.parameters[[2]]
 fit.best <- svm(Category ~ ., data=test.v, method = "C-classification",
                kernel = "radial", cost = bestC, gamma = bestGamma,
                probability=TRUE, scale = TRUE)



 fit.naive <- ksvm(Category~., data=train.v ,type = "C-svc",
                   kernel = "rbfdot", kpar=list(sigma = 0.1), C = 1,
                   scaled=TRUE, prob.model = TRUE, cross=10)
 # SELECT OPTIMAL PARAMETER C
# -----------------------------
  Cs <- 1:200/10;
  K <- 10  # K-fold CV
  Err.cv <- rep(0, length(Cs))
  for (i in 1:length(Cs)) {
```

```
    c <- Cs[i]
    fit.i <- ksvm(Category ~ ., data = train.v,
                  type = "C-svc", kernel = "rbfdot", kpar=list(sigma = 0.1), C = c,
                  scaled=TRUE, prob.model = TRUE, cross=K)
    Err.cv[i]<- attributes(fit.i)$cross
    #print(cbind(i=i, C=c, Error=Err.cv[i]))
  }
  # BEST CHOICE OF C
  C.best <- Cs[which.min(Err.cv)]; C.best


  # BEST SVM MODEL
  fit.b <- ksvm(Category ~ ., data = train.v,
                  type = "C-svc", kernel = "rbfdot",
                  kpar=list(sigma = 0.1), C = C.best,
                  scaled=TRUE, prob.model = TRUE, cross=10)
  #fit.b

  pred.fit1 <- predict(fit.best, test.v, type = "response")
  pred.fit2 <- predict(fit.b, test.v, type = "response")

  print(summary(pred.fit1))
  print(summary(pred.fit2))
    #------------------------
  # ROC and AUC Prediction
  #------------------------

  AUC.1 <- roc.area(obs=yobs, pred=as.numeric(pred.fit1))$A
  AUC.2 <- roc.area(obs=yobs, pred=as.numeric(pred.fit2))$A

  err.svm1[v] = AUC.1
  err.svm2[v] = AUC.2
  print(paste("AUC(1) for fold", v, ":", err.svm1[v]))
  print(paste("AUC(2) for fold", v, ":", err.svm2[v]))
  pred.rate = ifelse(pred.glm > 0.5, 1, 0); pred.rate # predicted 0/1's
  missed.rate <- mean(yobs != pred.rate)
  miscla[v] = missed.rate
  print(paste("Misclassified rate for fold", v,
  ":",miscla[v]))

}
})
```