ML_Model_Performance_Analysis_On_Spam_Information

```
info <- read_csv("spam-info.txt")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 121 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): 1. Title:  SPAM E-mail Database
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
test <- read.csv("spam-test.txt", header = FALSE)
train <- read.csv("spam-train.txt", header = FALSE)
names <- read_csv("spam-names.txt")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 86 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): | SPAM E-MAIL DATABASE ATTRIBUTES (in .names format)
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Question 1

```
#preprocessing
train1 <- as.data.frame(scale(train[1:57]))
test1 <- as.data.frame(scale(test[1:57]))
train_final <- cbind(train1,train[58])
test_final <- cbind(test1,test[58])
```

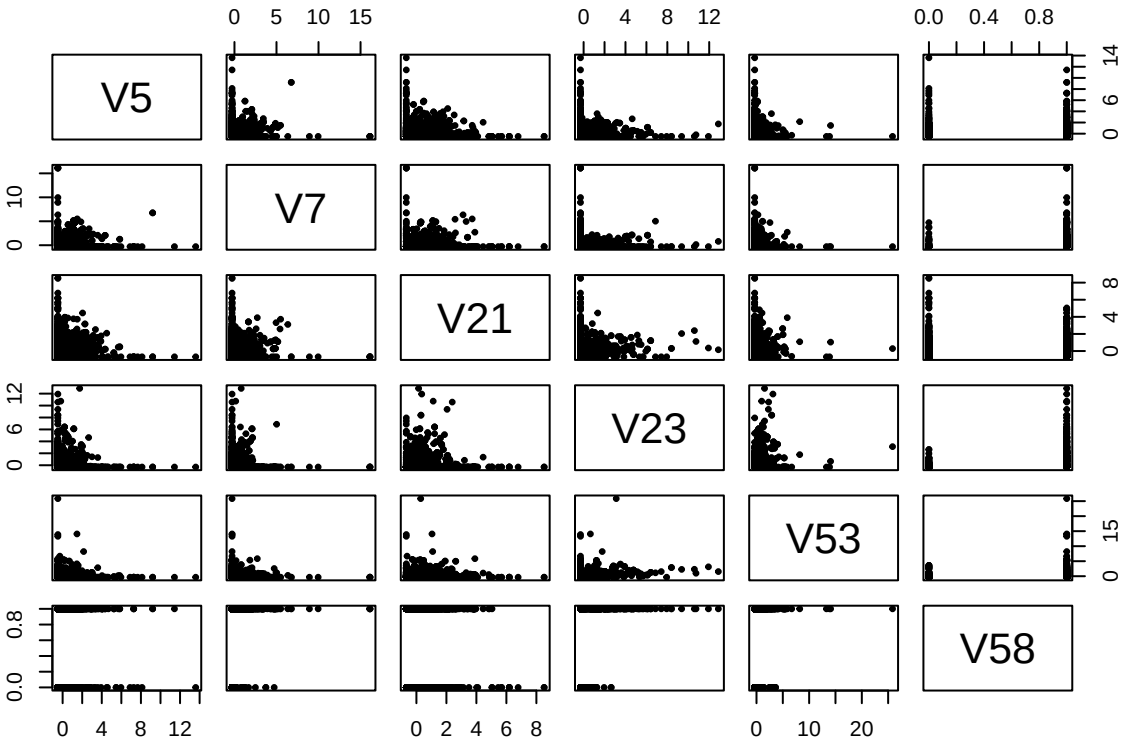
#(a)

```
#train
corr_train1 <- cor(train_final[1:57], train_final[58])
check <- ifelse(abs(corr_train1) > 0.3, corr_train1, 0)
get <- check[check != 0, ]
get
```

```
##        V5        V7       V21       V23       V53
```

```
## 0.3084844 0.3269769 0.3880512 0.3401085 0.3374699
```

```
new_train1 <- subset(train_final, select = c(V5, V7, V21, V23, V53, V58))
pairwise_train1 <- pairs(new_train1, pch = 19, cex = 0.5)
```



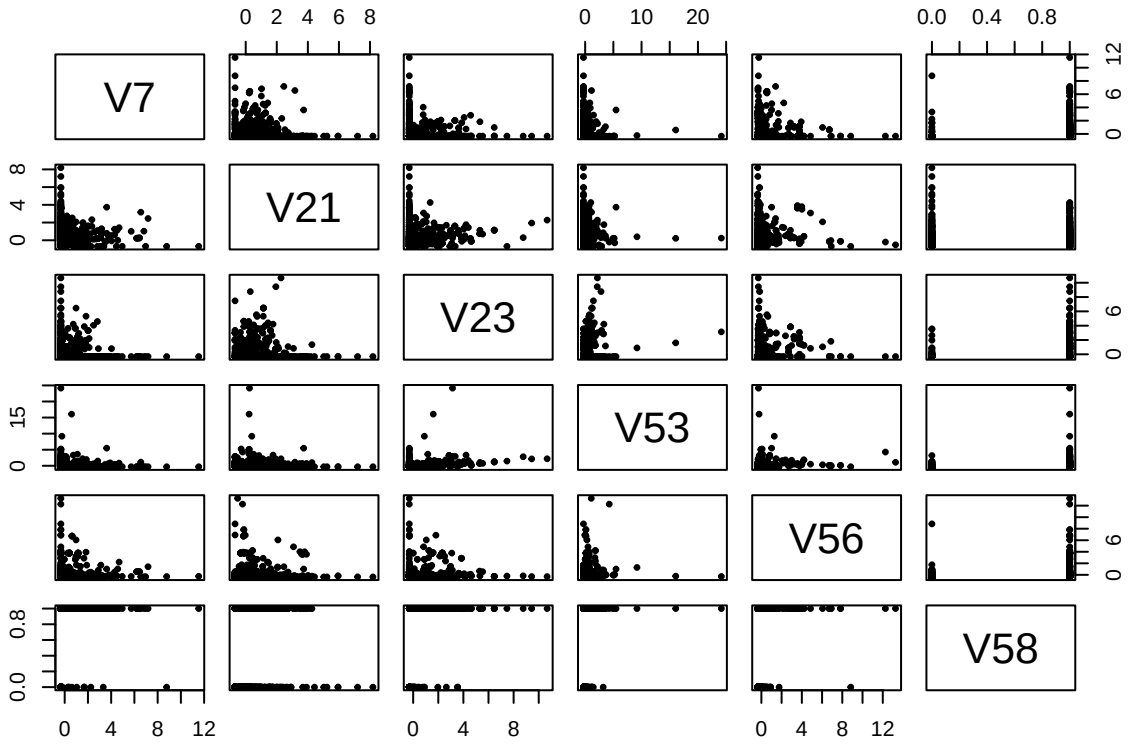

```
pairwise_train1
```

```
## NULL
```

```
corr_test1 <- cor(test_final[1:57], test_final[58])
check <- ifelse(abs(corr_test1) > 0.3, corr_test1, 0)
get <- check[check != 0, ]
get
```

```
##        V7       V21       V23       V53       V56
## 0.3649395 0.3878994 0.3449651 0.3116467 0.3067099
```

```
new_test1 <- subset(test_final, select = c(V7, V21, V23, V53, V56, V58))
pairwise_test1 <- pairs(new_test1, pch = 19, cex = 0.5)
```




```
pairwise_test1
```

```
## NULL
```

#(b)

```
# train
model1_train <- glm(V58~ .,  data = train_final, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred <- predict(model1_train, newdata = train_final, type = "response")

sum <- summary(model1_train)
sum
```

```
##
## Call:
## glm(formula = V58 ~ ., family = binomial, data = train_final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3245  -0.1988  -0.0001   0.0940   3.6053
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.36294    1.76165  -4.180 2.92e-05 ***
## V1           -0.07047    0.08544  -0.825 0.409508
## V2           -0.21268    0.13656  -1.557 0.119379
## V3            0.02573    0.07472   0.344 0.730612
## V4            5.42487    2.63430   2.059 0.039464 *
## V5            0.41029    0.08897   4.611 4.00e-06 ***
## V6            0.08488    0.05780   1.469 0.141965
```

```
## V7           1.30763    0.19827   6.595 4.24e-11 ***
## V8           0.20112    0.07309   2.752 0.005931 **
## V9           0.21642    0.10039   2.156 0.031095 *
## V10          0.05737    0.06090   0.942 0.346145
## V11         -0.19561    0.07523  -2.600 0.009319 **
## V12         -0.03552    0.07302  -0.486 0.626655
## V13         -0.13217    0.11069  -1.194 0.232431
## V14         -0.00339    0.06296  -0.054 0.957058
## V15          0.31084    0.23239   1.338 0.181023
## V16          1.10038    0.16449   6.690 2.24e-11 ***
## V17          0.59641    0.13999   4.260 2.04e-05 ***
## V18         -0.02993    0.08391  -0.357 0.721327
## V19          0.15357    0.07781   1.974 0.048423 *
## V20          1.80199    0.50899   3.540 0.000400 ***
## V21          0.49973    0.08500   5.879 4.13e-09 ***
## V22          0.10473    0.15871   0.660 0.509332
## V23          1.17267    0.24101   4.866 1.14e-06 ***
## V24          0.09945    0.06169   1.612 0.106930
## V25         -3.27164    0.58150  -5.626 1.84e-08 ***
## V26         -0.44855    0.39100  -1.147 0.251312
## V27        -18.55268    3.80185  -4.880 1.06e-06 ***
## V28          0.24526    0.17081   1.436 0.151031
## V29         -2.42887    1.66214  -1.461 0.143936
## V30          0.01145    0.09666   0.118 0.905705
## V31         -0.08296    0.25709  -0.323 0.746941
## V32         -0.37441    0.95348  -0.393 0.694553
## V33         -0.46280    0.24665  -1.876 0.060610 .
## V34          0.85386    1.01167   0.844 0.398662
## V35         -0.61202    0.35339  -1.732 0.083302 .
## V36          0.07618    0.16958   0.449 0.653264
## V37         -0.26049    0.14890  -1.749 0.080214 .
## V38         -0.15147    0.12133  -1.248 0.211871
## V39         -0.02633    0.15297  -0.172 0.863349
## V40         -0.15745    0.17675  -0.891 0.373028
## V41        -18.56408   12.22870  -1.518 0.128996
## V42         -1.69535    0.58310  -2.907 0.003644 **
## V43         -0.45417    0.23919  -1.899 0.057599 .
## V44         -0.73394    0.35711  -2.055 0.039857 *
## V45         -0.88579    0.17727  -4.997 5.83e-07 ***
## V46         -1.08493    0.25513  -4.252 2.11e-05 ***
## V47         -0.64235    0.32519  -1.975 0.048234 *
## V48         -0.50262    0.38329  -1.311 0.189745
## V49         -0.20714    0.10111  -2.049 0.040502 *
## V50          0.04754    0.06007   0.791 0.428765
## V51         -0.06586    0.12898  -0.511 0.609646
## V52          0.24800    0.05813   4.266 1.99e-05 ***
## V53          1.01664    0.16220   6.268 3.66e-10 ***
## V54          0.59058    0.33572   1.759 0.078551 .
## V55         -0.56200    0.22976  -2.446 0.014445 *
## V56          1.08271    0.29373   3.686 0.000228 ***
## V57          0.61655    0.14118   4.367 1.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4121.0  on 3066  degrees of freedom
## Residual deviance: 1157.4  on 3009  degrees of freedom
## AIC: 1273.4
##
## Number of Fisher Scoring iterations: 13
```

```
prob <- data.frame(sum$coefficients[,4])
prob <- prob %>% filter(sum.coefficients...4. < 0.01)
prob
```

```
##             sum.coefficients...4.
## (Intercept)          2.920602e-05
## V5                   4.000834e-06
## V7                   4.244434e-11
## V8                   5.930998e-03
## V11                  9.319500e-03
## V16                  2.236342e-11
## V17                  2.040443e-05
## V20                  3.996402e-04
## V21                  4.132772e-09
## V23                  1.140340e-06
## V25                  1.841738e-08
## V27                  1.061313e-06
## V42                  3.643685e-03
## V45                  5.831362e-07
## V46                  2.113996e-05
## V52                  1.989311e-05
## V53                  3.656660e-10
## V56                  2.277326e-04
## V57                  1.259030e-05
```

```
# Evaluate the model on the train data
accuracy <- mean((pred > 0.5) == train_final$V58)
accuracy
```

```
## [1] 0.9282687
```

```
# Calculate the classification error
error <- 1 - accuracy
error
```

```
## [1] 0.07173133
```

V5, V7, V8, V11, V16, V17, V20, V21, V23, V25, V27, V42, V45, V46, V52, V53, V56, V57 appear to be statistically significant.

Those has a sum.coefficients...4. less than 0.01 are statistically significant.

```
# test
model1_test <- glm(V58~ ., data = test_final, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred <- predict(model1_test, newdata = test_final, type = "response")

# Evaluate the model on the test data
accuracy <- mean((pred > 0.5) == test_final$V58)
```

```
accuracy
```

```
## [1] 0.9478488
```

```
# Calculate the classification error
error <- 1 - accuracy
error
```

```
## [1] 0.05215124
```

#(c)

```
#linear
lda.fit=lda(V58 ~ ., data=train_final)
```

```
#train
lda.pred=predict(lda.fit, train_final)
lda.class=lda.pred$class
true_value=train_final$V58
table(lda.class,true_value)
```

```
##          true_value
## lda.class    0    1
##         0 1770  233
##         1   79  985
```

```
test_error_LDA=mean(lda.class!=true_value)
test_error_LDA
```

```
## [1] 0.1017281
```

```
#test
lda.pred=predict(lda.fit, test_final)
lda.class=lda.pred$class
true_value=test_final$V58
table(lda.class,true_value)
```

```
##          true_value
## lda.class   0   1
##         0 873 115
##         1  43 503
```

```
test_error_LDA=mean(lda.class!=true_value)
test_error_LDA
```

```
## [1] 0.1029987
```

```
#quadratic
qda.fit=qda(V58 ~ ., data=train_final)
```

```
qda.pred=predict(qda.fit, train_final)
qda.class=qda.pred$class
true_value=train_final$V58
table(qda.class,true_value)
```

```
##          true_value
## qda.class    0    1
##         0 1369   68
##         1  480 1150
```

```
test_error_LDA=mean(qda.class!=true_value)
test_error_LDA
```

```
## [1] 0.1786762
```

```
#test
qda.pred=predict(qda.fit, test_final)
qda.class=qda.pred$class
true_value=test_final$V58
table(qda.class,true_value)
```

```
##          true_value
## qda.class   0   1
##         0 673  25
##         1 243 593
```

```
test_error_QDA=mean(qda.class!=true_value)
test_error_QDA
```

```
## [1] 0.1747066
```

#(d)

```
#linear
svm.model <- svm(V58~ ., data = train_final, cost = 1, kernel ="linear", type="C-classification")
summary(svm.model)
```

```
##
## Call:
## svm(formula = V58 ~ ., data = train_final, cost = 1, kernel = "linear",
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  621
##
##  ( 315 306 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#train
#obtain my confusion matrix:
svm.pred=predict(svm.model, train_final)
tab <- table(svm.pred, train_final$V58)
#And the classification error as:
mean(svm.pred != train_final$V58)
```

```
## [1] 0.06488425
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.model, test_final)
tab <- table(svm.pred, test_final$V58)
#And the classification error as:
mean(svm.pred != test_final$V58)
```

```
## [1] 0.07170795
```

```
#non linear
svm.nmodel <- svm(V58~ ., data = train_final, cost = 1, kernel ="radial", type="C-classification")
summary(svm.nmodel)
```

```
##
## Call:
## svm(formula = V58 ~ ., data = train_final, cost = 1, kernel = "radial",
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  926
##
##  ( 492 434 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#train
#obtain my confusion matrix:
svm.pred=predict(svm.nmodel, train_final)
tab <- table(svm.pred, train_final$V58)
#And the classification error as:
mean(svm.pred != train_final$V58)
```

```
## [1] 0.05151614
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.nmodel, test_final)
tab <- table(svm.pred, test_final$V58)
#And the classification error as:
mean(svm.pred != test_final$V58)
```

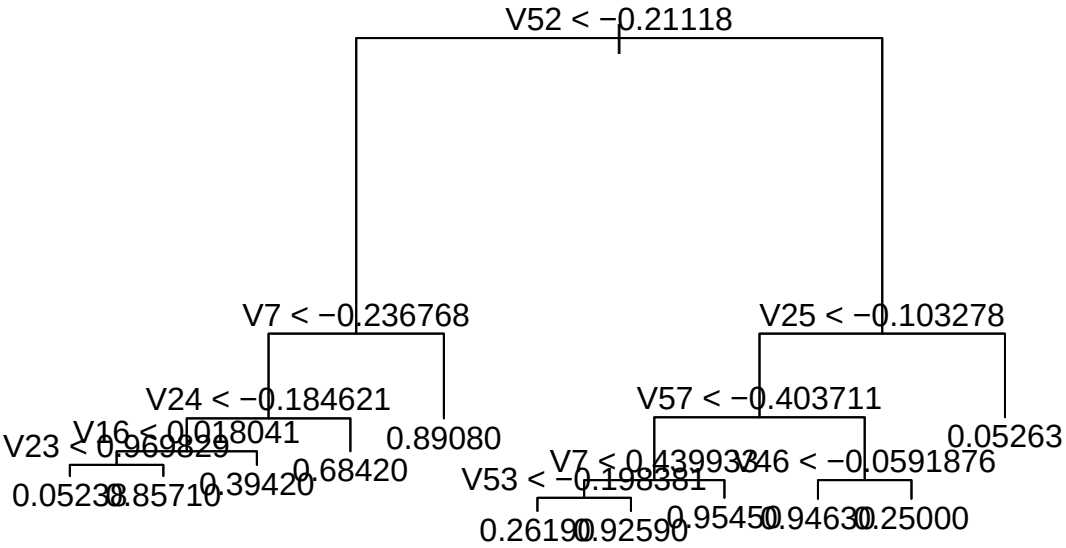
```
## [1] 0.06453716
```

#(e)

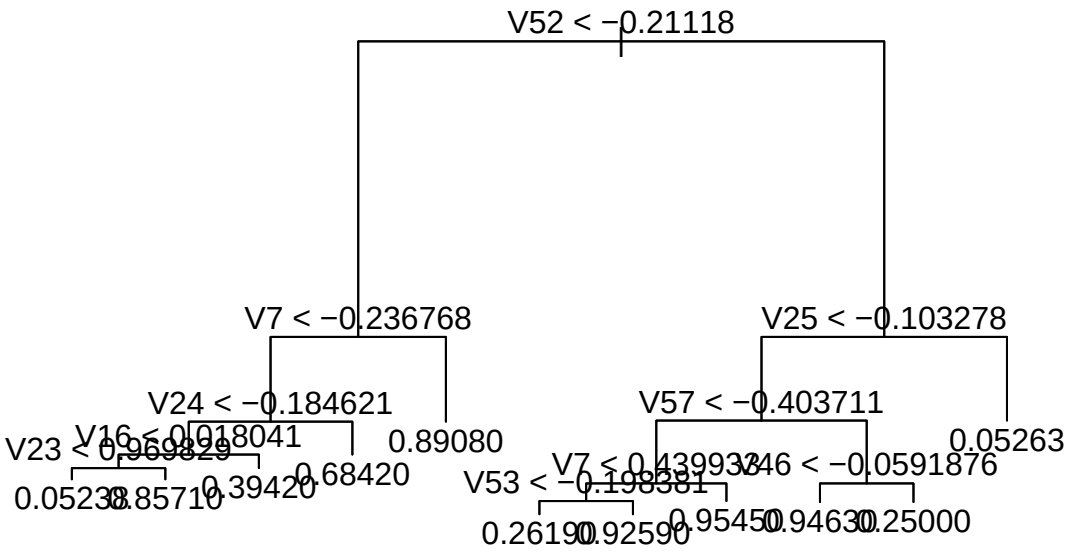
```
tree.boston = tree(V58~., data = train_final)
tree.boston.summary = summary(tree.boston)
tree.boston.summary
```

```
##
## Regression tree:
## tree(formula = V58 ~ ., data = train_final)
## Variables actually used in tree construction:
## [1] "V52" "V7"  "V24" "V16" "V23" "V25" "V57" "V53" "V46"
## Number of terminal nodes:  11
## Residual mean deviance:  0.07431 = 227.1 / 3056
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.95450 -0.05238 -0.05238  0.00000  0.05370  0.94760
```

```
cv.boston = cv.tree(tree.boston, K=10)
cv.size = cv.boston$size[which.min(cv.boston$dev)]
#prune
prune.boston = prune.tree(tree.boston, best=cv.size)
plot(tree.boston)
text(tree.boston, pretty=0)
```




```
plot(prune.boston)
text(prune.boston, pretty = 0)
```




```
#train
#Calculating the test errors
y.test = train_final$V58
yhat.single = predict(tree.boston, newdata = train_final)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = train_final)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.07."
```

```
## [1] "MSE for a single tree: 0.07."
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.01."
```

```
## [1]  "MSE for a pruned tree: 0.01."
```

```
#test
#Calculating the test errors
y.test = test_final$V58
yhat.single = predict(tree.boston, newdata = test_final)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = test_final)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.09."
```

```
## [1] "MSE for a single tree: 0.09."
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.02."
```

```
## [1]  "MSE for a pruned tree: 0.02."
```

Question 2

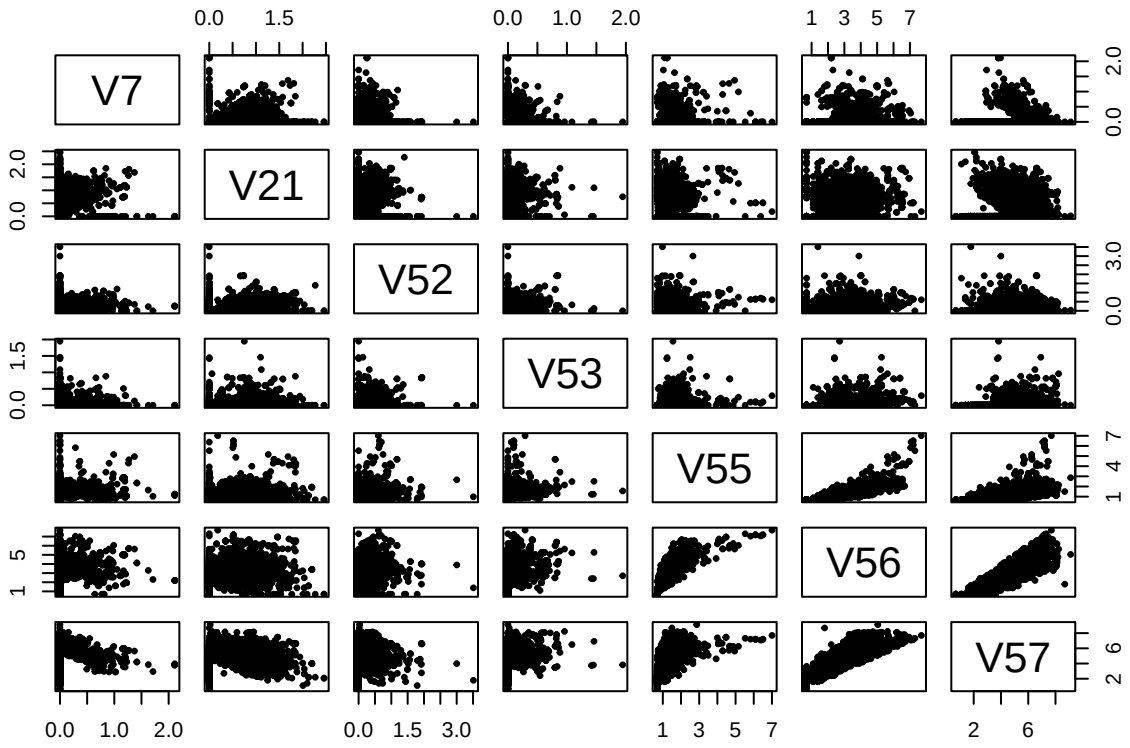
```
#preprocessing
train2 <- log(train[1:57] + 1)
test2 <- log(test[1:57] + 1)
train_final2 <- cbind(train2,train[58])
test_final2 <- cbind(test2,test[58])
```

#(a)

```
#train
corr_train2 <- cor(train_final2[1:57], train_final2[58])
check2 <- ifelse(abs(corr_train2) > 0.4, corr_train2, 0)
get2 <- check2[check2 != 0, ]
get2
```

```
##        V7       V21       V52       V53       V55       V56       V57
## 0.4150253 0.4753571 0.4783242 0.4271903 0.4175907 0.5044962 0.4386707
```

```
new_train2 <- subset(train_final2, select = c(V7, V21, V52, V53, V55, V56, V57))
pairwise_train2 <- pairs(new_train2, pch = 19, cex = 0.5)
```



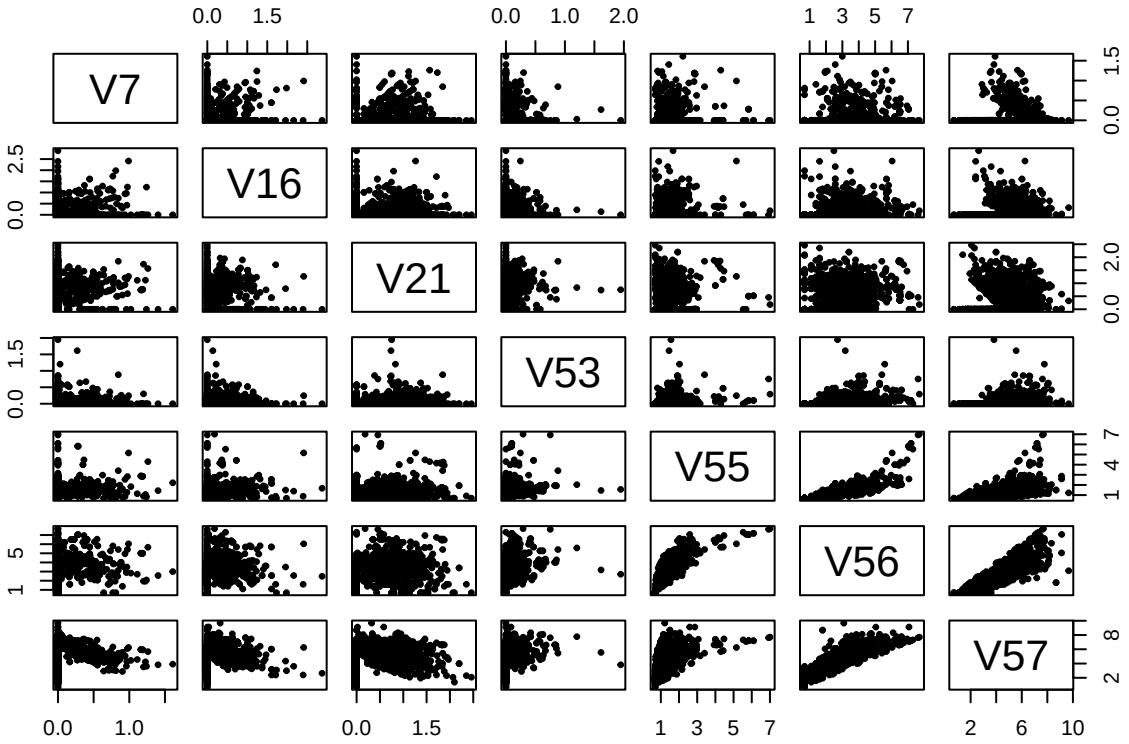

```
pairwise_train2
```

```
## NULL
```

```
corr_test2 <- cor(test_final2[1:57], test_final2[58])
check2 <- ifelse(abs(corr_test2) > 0.4, corr_test2, 0)
get2 <- check2[check2 != 0, ]
get2
```

```
##        V7       V16       V21       V52       V53       V55       V56       V57
## 0.4227755 0.4095069 0.4810112 0.4632664 0.4364713 0.4519084 0.5324709 0.4508028
```

```
new_test2 <- subset(test_final2, select = c(V7, V16, V21, V53, V55, V56, V57))
pairwise_test2 <- pairs(new_test2, pch = 19, cex = 0.5)
```




```
pairwise_test2
```

```
## NULL
```

#(b)

```
model2 <- glm(V58~ .,  data = train_final2, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
sum2 <- summary(model2)
sum2
```

```
##
## Call:
## glm(formula = V58 ~ ., family = binomial, data = train_final2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0831  -0.1646  -0.0010   0.0738   3.7853
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.55361    0.47536 -11.683  < 2e-16 ***
## V1           -0.50525    0.52078  -0.970 0.331955
## V2           -0.48375    0.41287  -1.172 0.241325
## V3           -0.34268    0.32461  -1.056 0.291122
## V4            2.49036    2.49963   0.996 0.319109
## V5            1.68052    0.26735   6.286 3.26e-10 ***
## V6            0.49007    0.49976   0.981 0.326779
```

```
## V7           3.81919    0.63656   6.000 1.98e-09 ***
## V8           1.11891    0.39254   2.850 0.004366 **
## V9           0.22162    0.61448   0.361 0.718349
## V10          0.20794    0.26664   0.780 0.435466
## V11         -1.73051    0.64790  -2.671 0.007563 **
## V12         -0.13019    0.21705  -0.600 0.548628
## V13         -1.47819    0.59699  -2.476 0.013284 *
## V14          0.49815    0.49244   1.012 0.311724
## V15          2.35454    1.31509   1.790 0.073389 .
## V16          2.00188    0.30550   6.553 5.64e-11 ***
## V17          2.00033    0.49917   4.007 6.14e-05 ***
## V18         -0.62599    0.34041  -1.839 0.065927 .
## V19          0.04966    0.17069   0.291 0.771075
## V20          4.74708    1.75988   2.697 0.006989 **
## V21          0.92793    0.20837   4.453 8.46e-06 ***
## V22          0.19783    0.59582   0.332 0.739860
## V23          3.39784    0.89163   3.811 0.000139 ***
## V24          1.27695    0.41124   3.105 0.001902 **
## V25         -3.97126    0.60152  -6.602 4.06e-11 ***
## V26         -0.43395    0.74531  -0.582 0.560401
## V27         -5.92242    1.42772  -4.148 3.35e-05 ***
## V28          1.27690    0.58913   2.167 0.030202 *
## V29         -5.52545    3.47037  -1.592 0.111344
## V30         -0.08833    0.47636  -0.185 0.852892
## V31         -1.17924    2.44793  -0.482 0.629997
## V32         -4.26131    4.43665  -0.960 0.336814
## V33         -1.44590    0.73243  -1.974 0.048368 *
## V34          0.86735    4.05419   0.214 0.830595
## V35         -2.60252    1.20495  -2.160 0.030784 *
## V36          0.44061    0.70994   0.621 0.534840
## V37         -1.55260    0.59961  -2.589 0.009615 **
## V38         -1.10219    1.36375  -0.808 0.418971
## V39          0.09940    0.80741   0.123 0.902025
## V40         -1.66152    1.14748  -1.448 0.147622
## V41        -45.30209   35.39198  -1.280 0.200542
## V42         -4.12654    1.24565  -3.313 0.000924 ***
## V43         -5.08561    1.94170  -2.619 0.008815 **
## V44         -2.90440    1.49695  -1.940 0.052354 .
## V45         -2.02986    0.41499  -4.891 1.00e-06 ***
## V46         -2.21581    0.52201  -4.245 2.19e-05 ***
## V47         -7.41904    4.88356  -1.519 0.128715
## V48         -2.02099    1.39842  -1.445 0.148405
## V49         -1.58851    0.79263  -2.004 0.045059 *
## V50         -0.01172    0.62116  -0.019 0.984945
## V51         -3.40426    2.64864  -1.285 0.198693
## V52          2.24783    0.29972   7.500 6.39e-14 ***
## V53          4.93003    0.88667   5.560 2.70e-08 ***
## V54         -0.01276    2.13277  -0.006 0.995225
## V55          0.57047    0.33492   1.703 0.088513 .
## V56          0.09317    0.19497   0.478 0.632744
## V57          0.75138    0.13167   5.707 1.15e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4121.01  on 3066  degrees of freedom
## Residual deviance:  930.67  on 3009  degrees of freedom
## AIC: 1046.7
##
## Number of Fisher Scoring iterations: 12
```

```
prob2 <- data.frame(sum2$coefficients[,4])
prob2 <- prob2 %>% filter(sum2.coefficients...4. < 0.01)
prob2
```

```
##             sum2.coefficients...4.
## (Intercept)           1.556959e-31
## V5                    3.259836e-10
## V7                    1.976092e-09
## V8                    4.365835e-03
## V11                   7.563353e-03
## V16                   5.643374e-11
## V17                   6.140670e-05
## V20                   6.988717e-03
## V21                   8.456029e-06
## V23                   1.385080e-04
## V24                   1.902223e-03
## V25                   4.056767e-11
## V27                   3.351604e-05
## V37                   9.615473e-03
## V42                   9.237874e-04
## V43                   8.814615e-03
## V45                   1.001393e-06
## V46                   2.188335e-05
## V52                   6.393530e-14
## V53                   2.695659e-08
## V57                   1.152015e-08
```

```
## train
pred2 <- predict(model2, newdata = train_final2, type = "response")

# Evaluate the model on the test data
accuracy2 <- mean((pred2 > 0.5) == train_final2$V58)
accuracy2
```

```
## [1] 0.9422889
```

```
# Calculate the classification error
error2 <- 1 - accuracy2
error2
```

```
## [1] 0.05771112
```

V5 V7 V8 V11 V16 V17 V20 V21 V23 V24 V25 V27 V37 V42 V43 V45 V46 V52 V53 appear to be statistically significant. Those has a sum.coefficients...4. less than 0.01 are statistically significant. VW50

```
# test
pred2 <- predict(model2, newdata = test_final2, type = "response")

# Evaluate the model on the test data
```

```r
accuracy2 <- mean((pred2 > 0.5) == test_final2$V58)
accuracy2
```

```
## [1] 0.9432855
```

```r
# Calculate the classification error
error2 <- 1 - accuracy2
error2
```

```
## [1] 0.05671447
```

#(c)

```r
#linear
lda2.fit=lda(V58 ~ ., data=train_final2)
```

```r
#train
lda2.pred=predict(lda2.fit, train_final2)
lda2.class=lda2.pred$class
true_value=train_final2$V58
table(lda2.class,true_value)
```

```
##           true_value
## lda2.class    0    1
##          0 1795  131
##          1   54 1087
```

```r
test_error_LDA=mean(lda2.class!=true_value)
test_error_LDA
```

```
## [1] 0.06031953
```

```r
#test
lda2.pred=predict(lda2.fit, test_final2)
lda2.class=lda2.pred$class
true_value=test_final2$V58
table(lda2.class,true_value)
```

```
##           true_value
## lda2.class   0   1
##          0 885  69
##          1  31 549
```

```r
test_error_LDA=mean(lda2.class!=true_value)
test_error_LDA
```

```
## [1] 0.06518905
```

```r
#quadratic
qda2.fit=qda(V58 ~ ., data=train_final2)
```

```r
#train
qda2.pred=predict(qda2.fit, train_final2)
qda2.class=qda2.pred$class
true_value=train_final2$V58
table(qda2.class,true_value)
```

```
##           true_value
## qda2.class    0    1
##          0 1433   71
```

```
##          1  416 1147
```

```
test_error_QDA=mean(qda2.class!=true_value)
test_error_QDA
```

```
## [1] 0.1587871
```

```
#test
qda2.pred=predict(qda2.fit, test_final2)
qda2.class=qda2.pred$class
true_value=test_final2$V58
table(qda2.class,true_value)
```

```
##           true_value
## qda2.class   0   1
##          0 702  27
##          1 214 591
```

```
test_error_QDA=mean(qda2.class!=true_value)
test_error_QDA
```

```
## [1] 0.1571056
```

#(d)

```
#linear
svm.model <- svm(V58 ~ ., data = train_final2, cost = 1, kernel ="linear", type="C-classification")
summary(svm.model)
```

```
##
## Call:
## svm(formula = V58 ~ ., data = train_final2, cost = 1, kernel = "linear",
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  490
##
##  ( 252 238 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#train
#obtain my confusion matrix:
svm.pred=predict(svm.model, train_final2)
tab <- table(svm.pred, train_final2$V58)
#And the classification error as:
mean(svm.pred != train_final2$V58)
```

```
## [1] 0.05412455
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.model, test_final2)
tab <- table(svm.pred, test_final2$V58)
#And the classification error as:
mean(svm.pred != test_final2$V58)
```

```
## [1] 0.05149935
```

```
#non linear
svm.nmodel <- svm(V58 ~ ., data = train_final2, cost = 1, kernel ="radial", type="C-classification")
summary(svm.nmodel)
```

```
##
## Call:
## svm(formula = V58 ~ ., data = train_final2, cost = 1, kernel = "radial",
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  786
##
##  ( 418 368 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.nmodel, train_final2)
tab <- table(svm.pred, train_final2$V58)
#And the classification error as:
mean(svm.pred != train_final2$V58)
```

```
## [1] 0.03880013
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.nmodel, test_final2)
tab <- table(svm.pred, test_final2$V58)
#And the classification error as:
mean(svm.pred != test_final2$V58)
```

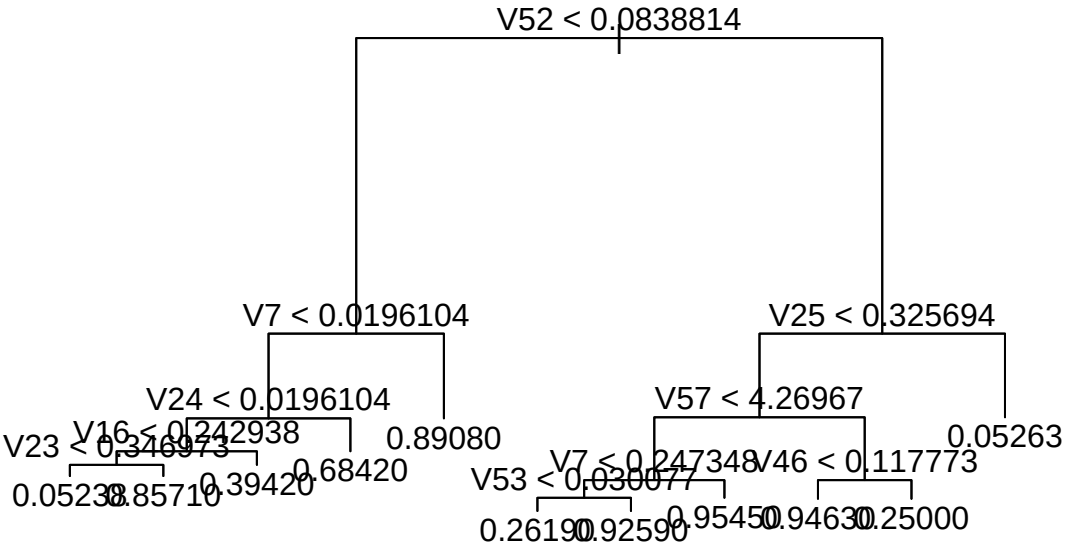
```
## [1] 0.04498044
```

#(e)

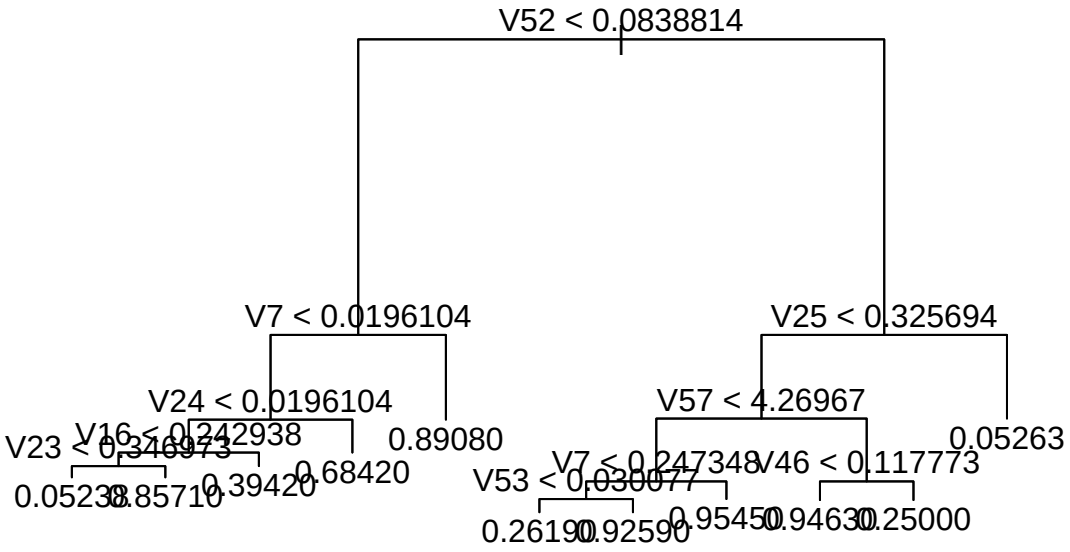
```
tree.boston = tree(V58 ~ ., data = train_final2)
tree.boston.summary = summary(tree.boston)
tree.boston.summary
```

```
##
## Regression tree:
## tree(formula = V58 ~ ., data = train_final2)
## Variables actually used in tree construction:
## [1] "V52" "V7"  "V24" "V16" "V23" "V25" "V57" "V53" "V46"
## Number of terminal nodes:  11
## Residual mean deviance:  0.07431 = 227.1 / 3056
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.95450 -0.05238 -0.05238  0.00000  0.05370  0.94760
```

```
cv.boston = cv.tree(tree.boston, K=10)
cv.size = cv.boston$size[which.min(cv.boston$dev)]
#prune
prune.boston = prune.tree(tree.boston, best=cv.size)
plot(tree.boston)
text(tree.boston, pretty=0)
```




```
plot(prune.boston)
text(prune.boston, pretty = 0)
```




```
#train
y.test = train_final2$V58
yhat.single = predict(tree.boston, newdata = train_final2)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = train_final2)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.07."
```

```
## [1]  "MSE for a single tree: 0.07."
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.01."
```

```
## [1]  "MSE for a pruned tree: 0.01."
```

```
#test
y.test = test_final2$V58
yhat.single = predict(tree.boston, newdata = test_final2)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = test_final2)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.08."
```

```
## [1]  "MSE for a single tree: 0.08."
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.01."
```

```
## [1]  "MSE for a pruned tree: 0.01.
```

Question 3

```
#preprocessing
train3 <- ifelse(train[1:57] >0, 1, 0)
test3 <- ifelse(test[1:57] >0, 1, 0)
```

```
train_final3 <- cbind(train3,train[58])
test_final3 <- cbind(test3,test[58])
```

#(a)

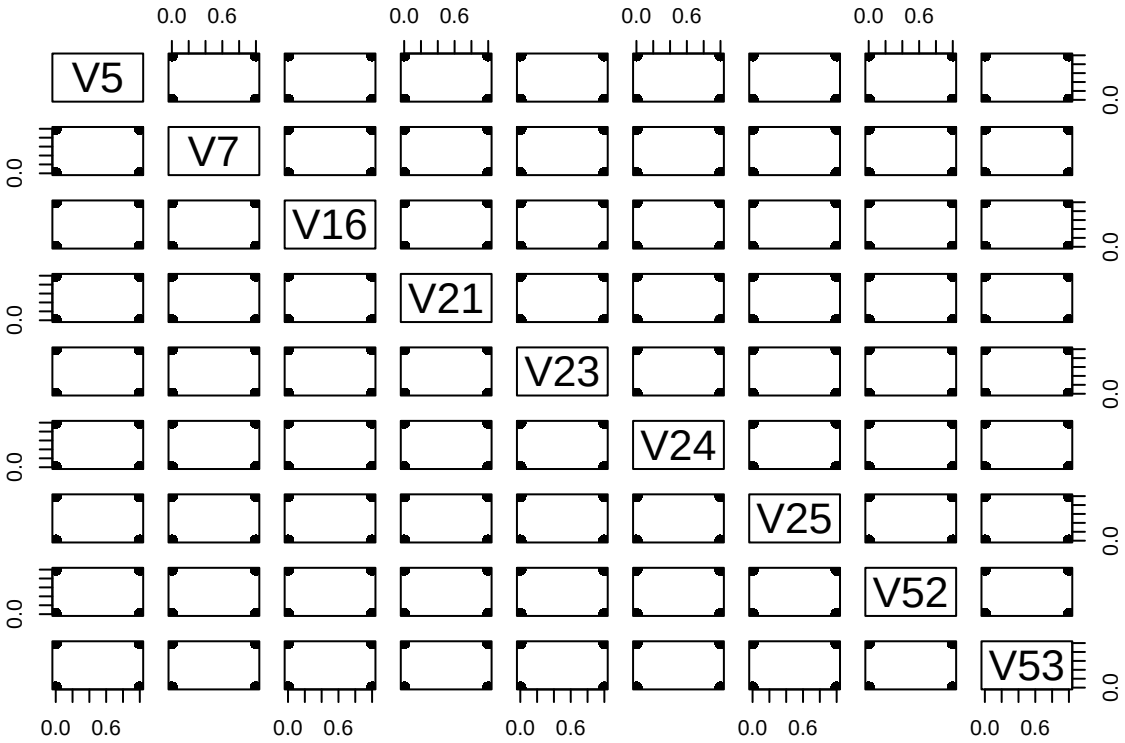
```
#train
corr_train3 <- cor(train_final3[1:57], train_final3[58])
```

```
## Warning in cor(train_final3[1:57], train_final3[58]): the standard deviation is
## zero
```

```
check3 <- ifelse(abs(corr_train3) > 0.4, corr_train3, 0)
get3 <- check3[check3 != 0, ]
get3
```

```
##         V5         V7        V16        V21        V23        V24        V25
##  0.4356354  0.5471629  0.5080070  0.4631326  0.4247405  0.4901018 -0.4031961
##        V52        V53       <NA>       <NA>       <NA>
##  0.5445165  0.5379247         NA         NA         NA
```

```
new_train3 <- subset(train_final3, select = c(V5, V7, V16, V21, V23, V24, V25, V52, V53))
pairwise_train3 <- pairs(new_train3, pch = 19, cex = 1)
```




```
pairwise_train3
```

```
## NULL
```

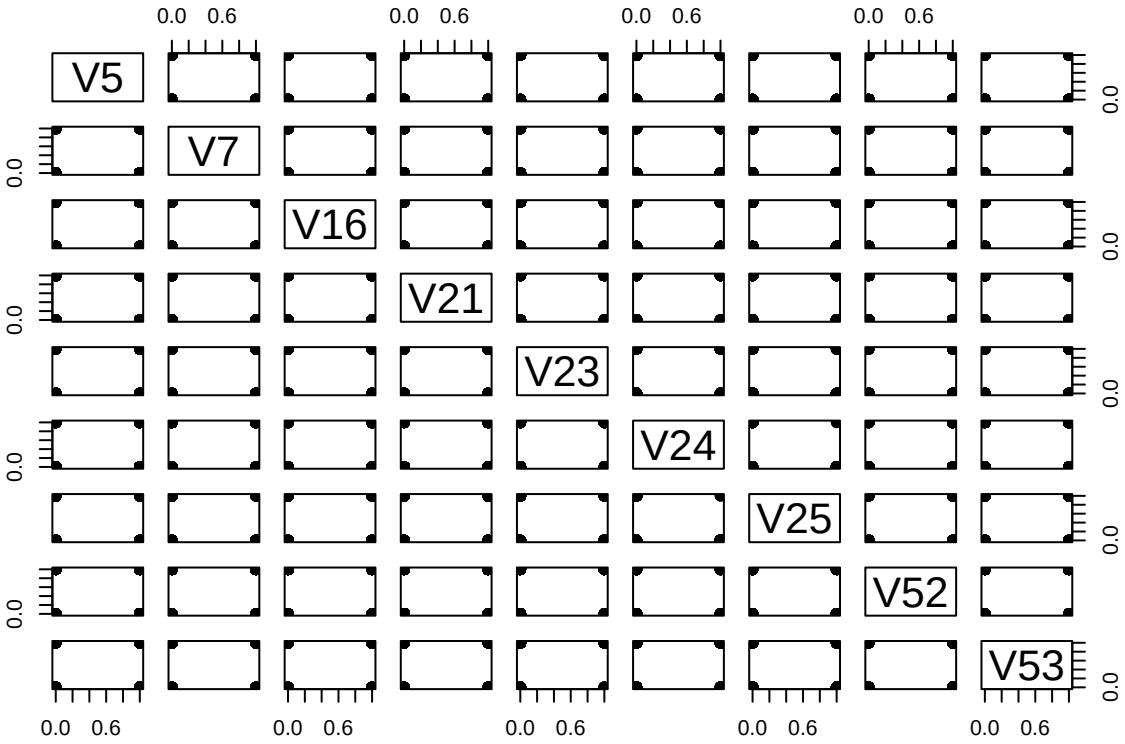
```
#test
corr_test3 <- cor(test_final3[1:57], test_final3[58])
```

```
## Warning in cor(test_final3[1:57], test_final3[58]): the standard deviation is
## zero
```

```
check3 <- ifelse(abs(corr_test3) > 0.4, corr_test3, 0)
get3 <- check3[check3 != 0, ]
get3
```

```
##         V5         V7        V16        V21        V23        V24        V25
##  0.4384555  0.5457793  0.4973257  0.4669913  0.4109408  0.4765877 -0.4069320
##        V52        V53       <NA>       <NA>       <NA>
##  0.5369301  0.5463490         NA         NA         NA
```

```
new_test3 <- subset(test_final3, select = c(V5, V7, V16, V21, V23, V24, V25, V52, V53))
pairwise_test3 <- pairs(new_test3, pch = 19, cex = 1)
```




```
pairwise_test3
```

```
## NULL
```

#(b)

```
# train
model3 <- glm(V58~ .,  data = train_final3, family = binomial)
sum3 <- summary(model3)
sum2
```

```
##
## Call:
## glm(formula = V58 ~ ., family = binomial, data = train_final2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0831  -0.1646  -0.0010   0.0738   3.7853
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  -5.55361    0.47536 -11.683  < 2e-16 ***
## V1           -0.50525    0.52078  -0.970 0.331955
## V2           -0.48375    0.41287  -1.172 0.241325
## V3           -0.34268    0.32461  -1.056 0.291122
## V4            2.49036    2.49963   0.996 0.319109
## V5            1.68052    0.26735   6.286 3.26e-10 ***
## V6            0.49007    0.49976   0.981 0.326779
## V7            3.81919    0.63656   6.000 1.98e-09 ***
## V8            1.11891    0.39254   2.850 0.004366 **
## V9            0.22162    0.61448   0.361 0.718349
## V10           0.20794    0.26664   0.780 0.435466
## V11          -1.73051    0.64790  -2.671 0.007563 **
## V12          -0.13019    0.21705  -0.600 0.548628
## V13          -1.47819    0.59699  -2.476 0.013284 *
## V14           0.49815    0.49244   1.012 0.311724
## V15           2.35454    1.31509   1.790 0.073389 .
## V16           2.00188    0.30550   6.553 5.64e-11 ***
## V17           2.00033    0.49917   4.007 6.14e-05 ***
## V18          -0.62599    0.34041  -1.839 0.065927 .
## V19           0.04966    0.17069   0.291 0.771075
## V20           4.74708    1.75988   2.697 0.006989 **
## V21           0.92793    0.20837   4.453 8.46e-06 ***
## V22           0.19783    0.59582   0.332 0.739860
## V23           3.39784    0.89163   3.811 0.000139 ***
## V24           1.27695    0.41124   3.105 0.001902 **
## V25          -3.97126    0.60152  -6.602 4.06e-11 ***
## V26          -0.43395    0.74531  -0.582 0.560401
## V27          -5.92242    1.42772  -4.148 3.35e-05 ***
## V28           1.27690    0.58913   2.167 0.030202 *
## V29          -5.52545    3.47037  -1.592 0.111344
## V30          -0.08833    0.47636  -0.185 0.852892
## V31          -1.17924    2.44793  -0.482 0.629997
## V32          -4.26131    4.43665  -0.960 0.336814
## V33          -1.44590    0.73243  -1.974 0.048368 *
## V34           0.86735    4.05419   0.214 0.830595
## V35          -2.60252    1.20495  -2.160 0.030784 *
## V36           0.44061    0.70994   0.621 0.534840
## V37          -1.55260    0.59961  -2.589 0.009615 **
## V38          -1.10219    1.36375  -0.808 0.418971
## V39           0.09940    0.80741   0.123 0.902025
## V40          -1.66152    1.14748  -1.448 0.147622
## V41         -45.30209   35.39198  -1.280 0.200542
## V42          -4.12654    1.24565  -3.313 0.000924 ***
## V43          -5.08561    1.94170  -2.619 0.008815 **
## V44          -2.90440    1.49695  -1.940 0.052354 .
## V45          -2.02986    0.41499  -4.891 1.00e-06 ***
## V46          -2.21581    0.52201  -4.245 2.19e-05 ***
## V47          -7.41904    4.88356  -1.519 0.128715
## V48          -2.02099    1.39842  -1.445 0.148405
## V49          -1.58851    0.79263  -2.004 0.045059 *
## V50          -0.01172    0.62116  -0.019 0.984945
## V51          -3.40426    2.64864  -1.285 0.198693
## V52           2.24783    0.29972   7.500 6.39e-14 ***
## V53           4.93003    0.88667   5.560 2.70e-08 ***
```

```
## V54          -0.01276    2.13277  -0.006 0.995225
## V55           0.57047    0.33492   1.703 0.088513 .
## V56           0.09317    0.19497   0.478 0.632744
## V57           0.75138    0.13167   5.707 1.15e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4121.01  on 3066  degrees of freedom
## Residual deviance:  930.67  on 3009  degrees of freedom
## AIC: 1046.7
##
## Number of Fisher Scoring iterations: 12
```

```
prob3 <- data.frame(sum2$coefficients[,4])
prob3 <- prob3 %>% filter(sum2.coefficients...4. < 0.01)
prob3
```

```
##             sum2.coefficients...4.
## (Intercept)           1.556959e-31
## V5                    3.259836e-10
## V7                    1.976092e-09
## V8                    4.365835e-03
## V11                   7.563353e-03
## V16                   5.643374e-11
## V17                   6.140670e-05
## V20                   6.988717e-03
## V21                   8.456029e-06
## V23                   1.385080e-04
## V24                   1.902223e-03
## V25                   4.056767e-11
## V27                   3.351604e-05
## V37                   9.615473e-03
## V42                   9.237874e-04
## V43                   8.814615e-03
## V45                   1.001393e-06
## V46                   2.188335e-05
## V52                   6.393530e-14
## V53                   2.695659e-08
## V57                   1.152015e-08
```

```
# train
pred3 <- predict(model3, newdata = train_final3, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
accuracy3 <- mean((pred3 > 0.5) == train_final3$V58)
accuracy3
```

```
## [1] 0.942941
```

```
# Calculate the classification error
error3 <- 1 - accuracy3
error3
```

```
## [1] 0.05705902
```

V5 V7 V8 V11 V16 V17 V20 V21 V23 V24 V25 V27 V37 V42 V43 V45 V46 V52 V53 V57 appear to be statistically significant. Those has a sum.coefficients. . . 4. less than 0.01 are statistically significant. VW50

```
# test
pred3 <- predict(model3, newdata = test_final3, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
# Evaluate the model on the test data
accuracy3 <- mean((pred3 > 0.5) == test_final3$V58)
accuracy3
```

```
## [1] 0.9191656
```

```
# Calculate the classification error
error3 <- 1 - accuracy3
error3
```

```
## [1] 0.08083442
```

#(c) #(d)

```
#linear
svm.model <- svm(V58 ~ ., data = train_final3, cost = 1, kernel ="linear", type="C-classification")
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'V55' and 'V56' and 'V57' constant. Cannot scale data.
```

```
summary(svm.model)
```

```
##
## Call:
## svm(formula = V58 ~ ., data = train_final3, cost = 1, kernel = "linear",
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  560
##
##  ( 280 280 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#train
#obtain my confusion matrix:
svm.pred=predict(svm.model, train_final3)
tab <- table(svm.pred, train_final3$V58)
#And the classification error as:
mean(svm.pred != train_final3$V58)
```

```
## [1] 0.06031953
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.model, test_final3)
tab <- table(svm.pred, test_final3$V58)
#And the classification error as:
mean(svm.pred != test_final3$V58)
```

```
## [1] 0.07431551
```

```
#non linear
svm.nmodel <- svm(V58 ~ ., data = train_final3, cost = 1, kernel ="radial", type="C-classification")
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'V55' and 'V56' and 'V57' constant. Cannot scale data.
```

```
summary(svm.nmodel)
```

```
##
## Call:
## svm(formula = V58 ~ ., data = train_final3, cost = 1, kernel = "radial",
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  814
##
##  ( 411 403 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
#train
#obtain my confusion matrix:
svm.pred=predict(svm.nmodel, train_final3)
tab <- table(svm.pred, train_final3$V58)
#And the classification error as:
mean(svm.pred != train_final3$V58)
```

```
## [1] 0.06162374
```

```
#test
#obtain my confusion matrix:
svm.pred=predict(svm.nmodel, test_final3)
tab <- table(svm.pred, test_final3$V58)
#And the classification error as:
mean(svm.pred != test_final3$V58)
```

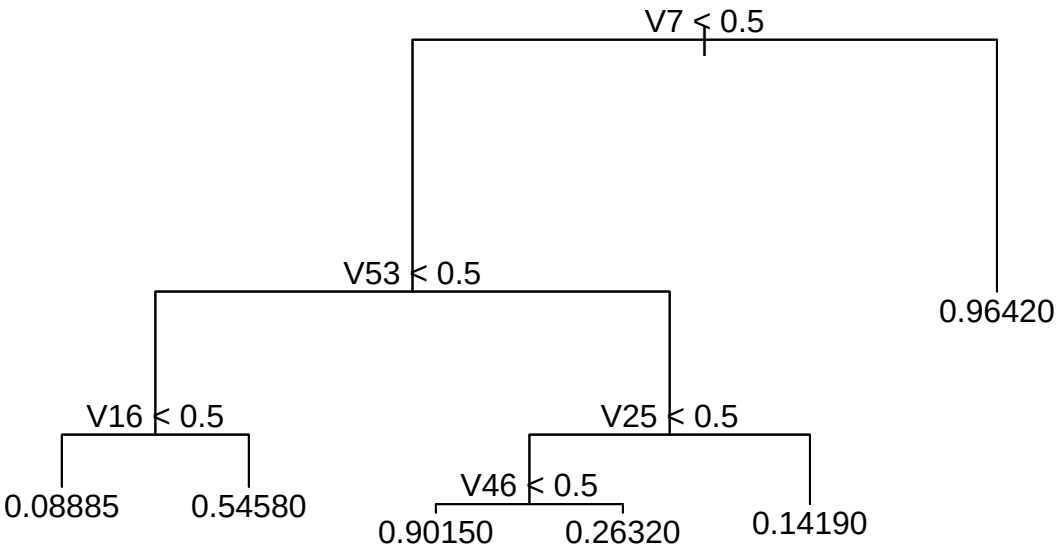
```
## [1] 0.0756193
```

#(e)

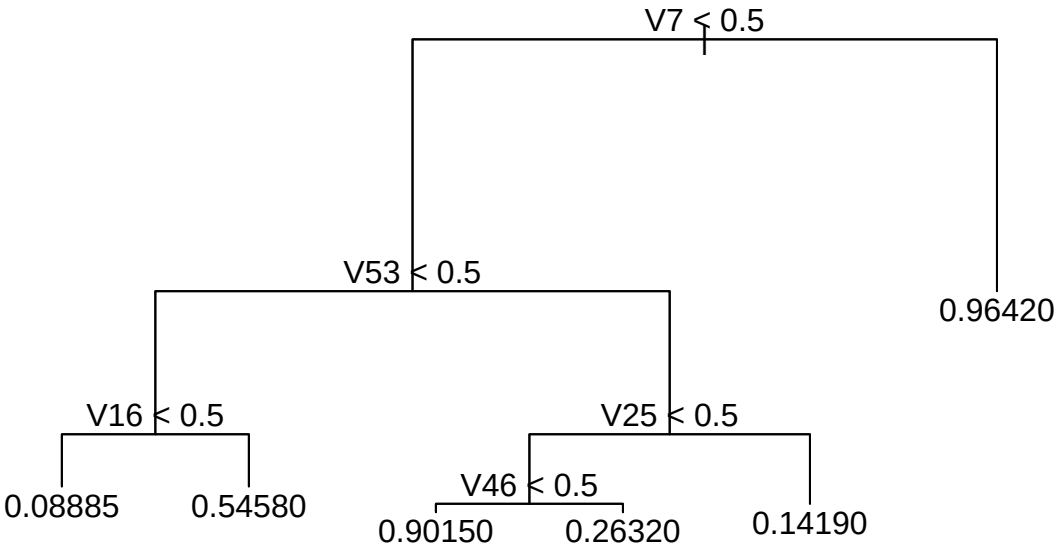
```
tree.boston = tree(V58 ~ ., data = train_final3)
tree.boston.summary = summary(tree.boston)
tree.boston.summary
```

```
##
## Regression tree:
## tree(formula = V58 ~ ., data = train_final3)
## Variables actually used in tree construction:
## [1] "V7"  "V53" "V16" "V25" "V46"
## Number of terminal nodes:  6
## Residual mean deviance:  0.09013 = 275.9 / 3061
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.96420 -0.08885 -0.08885  0.00000  0.03578  0.91120
```

```
cv.boston = cv.tree(tree.boston, K=10)
cv.size = cv.boston$size[which.min(cv.boston$dev)]
#prune
prune.boston = prune.tree(tree.boston, best=cv.size)
plot(tree.boston)
text(tree.boston, pretty=0)
```




```
plot(prune.boston)
text(prune.boston, pretty = 0)
```




```
#train
y.test = train_final3$V58
yhat.single = predict(tree.boston, newdata = train_final3)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = train_final3)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.09."
```

```
## [1]  "MSE for a single tree: 0.09"
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.01."
```

```
## [1]  "MSE for a pruned tree: 0.01"
```

```
#Calculating the test errors
y.test = test_final3$V58
yhat.single = predict(tree.boston, newdata = test_final3)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = test_final3)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.09."
```

```
## [1]  "MSE for a single tree: 0.09."
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.01."
```

```
## [1]  "MSE for a pruned tree: 0.01."
```

Report classification errors using different methods and different preprocessed data in a table, and comment on the different performances

```
mat1.data <- c(0.0717, 0.0521, 0.1017, 0.1030, 0.1787, 0.1747, 0.0649, 0.0717, 0.0515, 0.0645, 0.0100,
mat1 <- matrix(mat1.data, nrow=3, ncol=12,byrow=TRUE)
```

```
colnames(mat1) <- c('LRTrain', 'LRTest', 'LDTrain', 'LDTest', 'QDTrain', 'QDTest', 'LSVMTrain', 'LSVMTes
rownames(mat1) <- c('STD', 'Log', 'Discret')
table <- as.table(mat1)
table
```

```
##         LRTrain LRTest LDTrain LDTest QDTrain QDTest LSVMTrain LSVMTest
## STD      0.0717 0.0521  0.1017 0.1030  0.1787 0.1747    0.0649   0.0717
## Log      0.0577 0.0567  0.0603 0.0652  0.1588 0.1571    0.0541   0.0515
## Discret  0.0571 0.0808                                  0.0603   0.0743
##         NLSVMTrain NLSVMTest TBTrain TBTest
## STD         0.0515    0.0645  0.0100 0.0200
## Log         0.0389    0.0450  0.0100 0.0100
## Discret     0.0616    0.0756  0.0100 0.0100
```

For all three different preprocessed data, tree-based classifiers perform the best for both training and test datasets. For standardized data and the log-transformed data, quadratic discriminant analysis methods perform the worst for both training and test datasets. For Discretized data, logistic regression model has the worst performance for the test data, while nonlinear support vector machine classifiers has the worst performance for the training data.

design a classifier with test error rate as small as possible

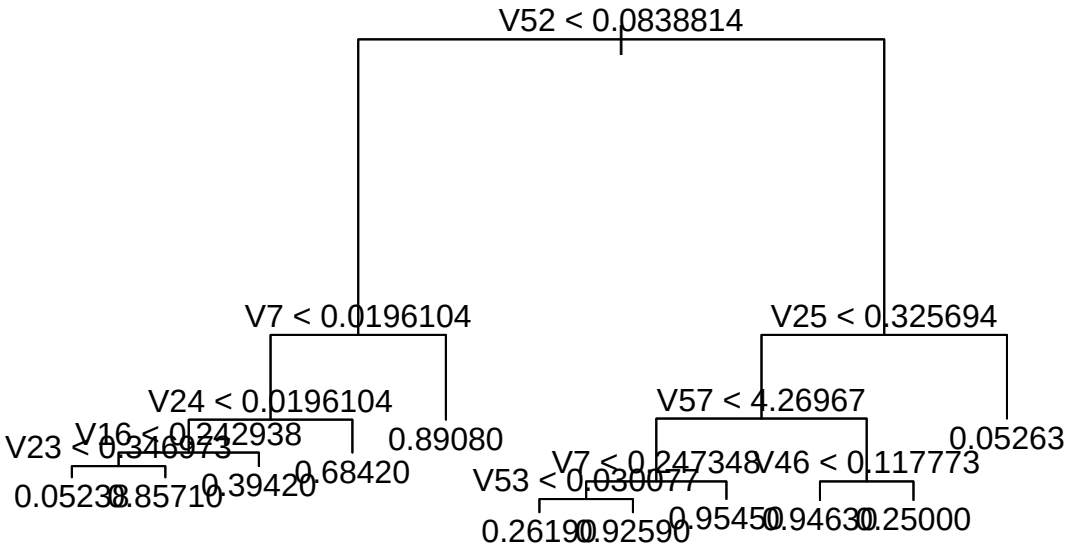
```
#Transform the features using Log(xij + 1) first
train_model <- log(train[1:57] + 1)
test_model <- log(test[1:57] + 1)
train_model <- cbind(train_model,train[58])
test_model <- cbind(test_model,test[58])

#Apply tree-based classifiers to the data
tree.boston = tree(V58 ~ ., data = train_model)
tree.boston.summary = summary(tree.boston)
cv.boston = cv.tree(tree.boston, K=10)
cv.size = cv.boston$size[which.min(cv.boston$dev)]


#prune
prune.boston = prune.tree(tree.boston, best=cv.size)
summary(prune.boston)
```

```
##
## Regression tree:
## tree(formula = V58 ~ ., data = train_model)
## Variables actually used in tree construction:
## [1] "V52" "V7"  "V24" "V16" "V23" "V25" "V57" "V53" "V46"
## Number of terminal nodes:  11
## Residual mean deviance:  0.07431 = 227.1 / 3056
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.95450 -0.05238 -0.05238  0.00000  0.05370  0.94760
```

```
plot(tree.boston)
text(tree.boston, pretty=0)
```




```
#Calculating the test errors
y.test = test_model$V58
yhat.single = predict(tree.boston, newdata = test_model)
mse.single = mean((yhat.single - y.test)^2)
yhat.prune = predict(prune.boston, newdata = test_model)
mse.prune = mean((yhat.prune - y.test)^22)
sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 0.08."
```

```
## [1]  "MSE for a single tree: 0.08."
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 0.01."
```

```
## [1]  "MSE for a pruned tree: 0.01."
```

We recommended preprocessed the data by transforming the features using Log(xij + 1) first, then apply a decision tree model. Based on the pruning result, we found that the variables were actually used in tree construction are “V52” “V7” “V24” “V16” “V23” “V25” “V57” “V53” “V46”, and there are only 11 nodes will achieve the best performace with accuracy 99%.