CA2 Ionosphere Data

Robert Dowd

1. Ionosphere Data:
   Load the mlbench package and load in the Ionosphere dataset from this package. Use help and str to understand the data that was collected on radar data at hospital in Labrador.

set.seed(153)
(a)
Have a look at the data, are the any variables to remove before running any models?

```
#Libraries
library(rpart) #to run decision tree
library(rpart.plot) #to plot decision tree

## Warning: package 'rpart.plot' was built under R version 4.2.3

library(gbm)

## Loaded gbm 2.2.2

## This version of gbm is no longer under development. Consider transitioning
to gbm3, https://github.com/gbm-developers/gbm3

library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

library(ipred)
library(caret) #to get the confusion matrix

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice

library(pROC) #to get the ROC Curve

## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

#install.packages('mlbench')
library(mlbench) #to get the Ionosphere dataset

## Warning: package 'mlbench' was built under R version 4.2.3

library(e1071) #to use svm
data(Ionosphere)

head(Ionosphere)

##   V1 V2      V3       V4       V5       V6       V7       V8      V9
V10
## 1  1  0 0.99539 -0.05889  0.85243  0.02306  0.83398 -0.37708 1.00000
0.03760
## 2  1  0 1.00000 -0.18829  0.93035 -0.36156 -0.10868 -0.93597 1.00000 -
0.04549
## 3  1  0 1.00000 -0.03365  1.00000  0.00485  1.00000 -0.12062 0.88965
0.01198
## 4  1  0 1.00000 -0.45161  1.00000  1.00000  0.71216 -1.00000 0.00000
0.00000
## 5  1  0 1.00000 -0.02401  0.94140  0.06531  0.92106 -0.23255 0.77152 -
0.16399
## 6  1  0 0.02337 -0.00592 -0.09924 -0.11949 -0.00763 -0.11824 0.14706
0.06637
##       V11      V12     V13      V14      V15      V16      V17      V18
## 1 0.85243 -0.17755 0.59755 -0.44945  0.60536 -0.38223  0.84356 -0.38542
## 2 0.50874 -0.67743 0.34432 -0.69707 -0.51685 -0.97515  0.05499 -0.62237
## 3 0.73082  0.05346 0.85443  0.00827  0.54591  0.00299  0.83775 -0.13644
## 4 0.00000  0.00000 0.00000  0.00000 -1.00000  0.14516  0.54094 -0.39330
## 5 0.52798 -0.20275 0.56409 -0.00712  0.34395 -0.27457  0.52940 -0.21780
## 6 0.03786 -0.06302 0.00000  0.00000 -0.04572 -0.15540 -0.00343 -0.10196
##         V19      V20      V21      V22      V23      V24      V25      V26
## 1  0.58212 -0.32192  0.56971 -0.29674  0.36946 -0.47357  0.56811 -0.51171
## 2  0.33109 -1.00000 -0.13151 -0.45300 -0.18056 -0.35734 -0.20332 -0.26569
## 3  0.75535 -0.08540  0.70887 -0.27502  0.43385 -0.12062  0.57528 -0.40220
## 4 -1.00000 -0.54467 -0.69975  1.00000  0.00000  0.00000  1.00000  0.90695
## 5  0.45107 -0.17813  0.05982 -0.35575  0.02309 -0.52879  0.03286 -0.65158
## 6 -0.11575 -0.05414  0.01838  0.03669  0.01519  0.00888  0.03513 -0.01535
##         V27      V28      V29      V30      V31      V32      V33      V34
Class
## 1  0.41078 -0.46168  0.21266 -0.34090  0.42267 -0.54487  0.18641 -0.45300
good
## 2 -0.20468 -0.18401 -0.19040 -0.11593 -0.16626 -0.06288 -0.13738 -0.02447
bad
```

```
## 3  0.58984 -0.22145  0.43100 -0.17365  0.60436 -0.24180  0.56045 -0.38238
good
## 4  0.51613  1.00000  1.00000 -0.20099  0.25682  1.00000 -0.32382  1.00000
bad
## 5  0.13290 -0.53206  0.02431 -0.62197 -0.05707 -0.59573 -0.04608 -0.65697
good
## 6 -0.03240  0.09223 -0.07859  0.00732  0.00000  0.00000 -0.00039  0.12011
bad
```

```
str(Ionosphere)
```

```
## 'data.frame':    351 obs. of  35 variables:
##  $ V1   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...
##  $ V2   : Factor w/ 1 level "0": 1 1 1 1 1 1 1 1 1 1 ...
##  $ V3   : num  0.995 1 1 1 1 ...
##  $ V4   : num  -0.0589 -0.1883 -0.0336 -0.4516 -0.024 ...
##  $ V5   : num  0.852 0.93 1 1 0.941 ...
##  $ V6   : num  0.02306 -0.36156 0.00485 1 0.06531 ...
##  $ V7   : num  0.834 -0.109 1 0.712 0.921 ...
##  $ V8   : num  -0.377 -0.936 -0.121 -1 -0.233 ...
##  $ V9   : num  1 1 0.89 0 0.772 ...
##  $ V10  : num  0.0376 -0.0455 0.012 0 -0.164 ...
##  $ V11  : num  0.852 0.509 0.731 0 0.528 ...
##  $ V12  : num  -0.1776 -0.6774 0.0535 0 -0.2028 ...
##  $ V13  : num  0.598 0.344 0.854 0 0.564 ...
##  $ V14  : num  -0.44945 -0.69707 0.00827 0 -0.00712 ...
##  $ V15  : num  0.605 -0.517 0.546 -1 0.344 ...
##  $ V16  : num  -0.38223 -0.97515 0.00299 0.14516 -0.27457 ...
##  $ V17  : num  0.844 0.055 0.838 0.541 0.529 ...
##  $ V18  : num  -0.385 -0.622 -0.136 -0.393 -0.218 ...
##  $ V19  : num  0.582 0.331 0.755 -1 0.451 ...
##  $ V20  : num  -0.3219 -1 -0.0854 -0.5447 -0.1781 ...
##  $ V21  : num  0.5697 -0.1315 0.7089 -0.6997 0.0598 ...
##  $ V22  : num  -0.297 -0.453 -0.275 1 -0.356 ...
##  $ V23  : num  0.3695 -0.1806 0.4339 0 0.0231 ...
##  $ V24  : num  -0.474 -0.357 -0.121 0 -0.529 ...
##  $ V25  : num  0.5681 -0.2033 0.5753 1 0.0329 ...
##  $ V26  : num  -0.512 -0.266 -0.402 0.907 -0.652 ...
##  $ V27  : num  0.411 -0.205 0.59 0.516 0.133 ...
##  $ V28  : num  -0.462 -0.184 -0.221 1 -0.532 ...
##  $ V29  : num  0.2127 -0.1904 0.431 1 0.0243 ...
##  $ V30  : num  -0.341 -0.116 -0.174 -0.201 -0.622 ...
##  $ V31  : num  0.4227 -0.1663 0.6044 0.2568 -0.0571 ...
##  $ V32  : num  -0.5449 -0.0629 -0.2418 1 -0.5957 ...
##  $ V33  : num  0.1864 -0.1374 0.5605 -0.3238 -0.0461 ...
##  $ V34  : num  -0.453 -0.0245 -0.3824 1 -0.657 ...
##  $ Class: Factor w/ 2 levels "bad","good": 2 1 2 1 2 1 2 1 2 1 ...
```

```
help(Ionosphere)
```

```
#Creating new data frame so we do not over write the original data frame
Ionosphere_Data <- Ionosphere
variance <- apply(Ionosphere_Data[ , -ncol(Ionosphere_Data)], 2, var)
variance

##         V1         V2         V3         V4         V5         V6
V7
## 0.09681726 0.00000000 0.24771345 0.19486466 0.27025599 0.21234597
0.24270774
##         V8         V9        V10        V11        V12        V13
V14
## 0.27118045 0.25711545 0.23411168 0.31752815 0.24484431 0.38711557
0.24489893
##        V15        V16        V17        V18        V19        V20
V21
## 0.42618418 0.21010367 0.38194916 0.24677247 0.39221012 0.26943999
0.37189058
##        V22        V23        V24        V25        V26        V27
V28
## 0.26849589 0.36453518 0.27821017 0.33460543 0.25856666 0.26646726
0.30252777
##        V29        V30        V31        V32        V33        V34
## 0.33164418 0.25803769 0.32659324 0.26375862 0.27317700 0.21933975

#Excluding V2 as it has no variance
Ionosphere_Data$V2 <- NULL
#Converting V1 to numeric
Ionosphere_Data$V1 <- as.numeric(Ionosphere_Data$V1)
head(Ionosphere_Data)

##   V1      V3       V4       V5       V6       V7       V8      V9      V10
## 1  2 0.99539 -0.05889  0.85243  0.02306  0.83398 -0.37708 1.00000  0.03760
## 2  2 1.00000 -0.18829  0.93035 -0.36156 -0.10868 -0.93597 1.00000 -0.04549
## 3  2 1.00000 -0.03365  1.00000  0.00485  1.00000 -0.12062 0.88965  0.01198
## 4  2 1.00000 -0.45161  1.00000  1.00000  0.71216 -1.00000 0.00000  0.00000
## 5  2 1.00000 -0.02401  0.94140  0.06531  0.92106 -0.23255 0.77152 -0.16399
## 6  2 0.02337 -0.00592 -0.09924 -0.11949 -0.00763 -0.11824 0.14706  0.06637
##       V11      V12     V13      V14      V15      V16      V17      V18
## 1 0.85243 -0.17755 0.59755 -0.44945  0.60536 -0.38223  0.84356 -0.38542
## 2 0.50874 -0.67743 0.34432 -0.69707 -0.51685 -0.97515  0.05499 -0.62237
## 3 0.73082  0.05346 0.85443  0.00827  0.54591  0.00299  0.83775 -0.13644
## 4 0.00000  0.00000 0.00000  0.00000 -1.00000  0.14516  0.54094 -0.39330
## 5 0.52798 -0.20275 0.56409 -0.00712  0.34395 -0.27457  0.52940 -0.21780
## 6 0.03786 -0.06302 0.00000  0.00000 -0.04572 -0.15540 -0.00343 -0.10196
##        V19      V20      V21      V22      V23      V24      V25      V26
## 1  0.58212 -0.32192  0.56971 -0.29674  0.36946 -0.47357  0.56811 -0.51171
## 2  0.33109 -1.00000 -0.13151 -0.45300 -0.18056 -0.35734 -0.20332 -0.26569
## 3  0.75535 -0.08540  0.70887 -0.27502  0.43385 -0.12062  0.57528 -0.40220
## 4 -1.00000 -0.54467 -0.69975  1.00000  0.00000  0.00000  1.00000  0.90695
## 5  0.45107 -0.17813  0.05982 -0.35575  0.02309 -0.52879  0.03286 -0.65158
## 6 -0.11575 -0.05414  0.01838  0.03669  0.01519  0.00888  0.03513 -0.01535
##        V27      V28      V29      V30      V31      V32      V33      V34
Class
## 1  0.41078 -0.46168  0.21266 -0.34090  0.42267 -0.54487  0.18641 -0.45300
good
## 2 -0.20468 -0.18401 -0.19040 -0.11593 -0.16626 -0.06288 -0.13738 -0.02447
bad
## 3  0.58984 -0.22145  0.43100 -0.17365  0.60436 -0.24180  0.56045 -0.38238
good
## 4  0.51613  1.00000  1.00000 -0.20099  0.25682  1.00000 -0.32382  1.00000
bad
## 5  0.13290 -0.53206  0.02431 -0.62197 -0.05707 -0.59573 -0.04608 -0.65697
good
## 6 -0.03240  0.09223 -0.07859  0.00732  0.00000  0.00000 -0.00039  0.12011
bad
```

```
str(Ionosphere_Data)
```

```
## 'data.frame':    351 obs. of  34 variables:
##  $ V1   : num  2 2 2 2 2 2 2 1 2 2 ...
##  $ V3   : num  0.995 1 1 1 1 ...
##  $ V4   : num  -0.0589 -0.1883 -0.0336 -0.4516 -0.024 ...
##  $ V5   : num  0.852 0.93 1 1 0.941 ...
##  $ V6   : num  0.02306 -0.36156 0.00485 1 0.06531 ...
##  $ V7   : num  0.834 -0.109 1 0.712 0.921 ...
##  $ V8   : num  -0.377 -0.936 -0.121 -1 -0.233 ...
##  $ V9   : num  1 1 0.89 0 0.772 ...
##  $ V10  : num  0.0376 -0.0455 0.012 0 -0.164 ...
##  $ V11  : num  0.852 0.509 0.731 0 0.528 ...
##  $ V12  : num  -0.1776 -0.6774 0.0535 0 -0.2028 ...
##  $ V13  : num  0.598 0.344 0.854 0 0.564 ...
##  $ V14  : num  -0.44945 -0.69707 0.00827 0 -0.00712 ...
##  $ V15  : num  0.605 -0.517 0.546 -1 0.344 ...
##  $ V16  : num  -0.38223 -0.97515 0.00299 0.14516 -0.27457 ...
##  $ V17  : num  0.844 0.055 0.838 0.541 0.529 ...
##  $ V18  : num  -0.385 -0.622 -0.136 -0.393 -0.218 ...
##  $ V19  : num  0.582 0.331 0.755 -1 0.451 ...
##  $ V20  : num  -0.3219 -1 -0.0854 -0.5447 -0.1781 ...
##  $ V21  : num  0.5697 -0.1315 0.7089 -0.6997 0.0598 ...
##  $ V22  : num  -0.297 -0.453 -0.275 1 -0.356 ...
##  $ V23  : num  0.3695 -0.1806 0.4339 0 0.0231 ...
##  $ V24  : num  -0.474 -0.357 -0.121 0 -0.529 ...
##  $ V25  : num  0.5681 -0.2033 0.5753 1 0.0329 ...
##  $ V26  : num  -0.512 -0.266 -0.402 0.907 -0.652 ...
##  $ V27  : num  0.411 -0.205 0.59 0.516 0.133 ...
##  $ V28  : num  -0.462 -0.184 -0.221 1 -0.532 ...
##  $ V29  : num  0.2127 -0.1904 0.431 1 0.0243 ...
##  $ V30  : num  -0.341 -0.116 -0.174 -0.201 -0.622 ...
##  $ V31  : num  0.4227 -0.1663 0.6044 0.2568 -0.0571 ...
##  $ V32  : num  -0.5449 -0.0629 -0.2418 1 -0.5957 ...
##  $ V33  : num  0.1864 -0.1374 0.5605 -0.3238 -0.0461 ...
##  $ V34  : num  -0.453 -0.0245 -0.3824 1 -0.657 ...
##  $ Class: Factor w/ 2 levels "bad","good": 2 1 2 1 2 1 2 1 2 1 ...
```

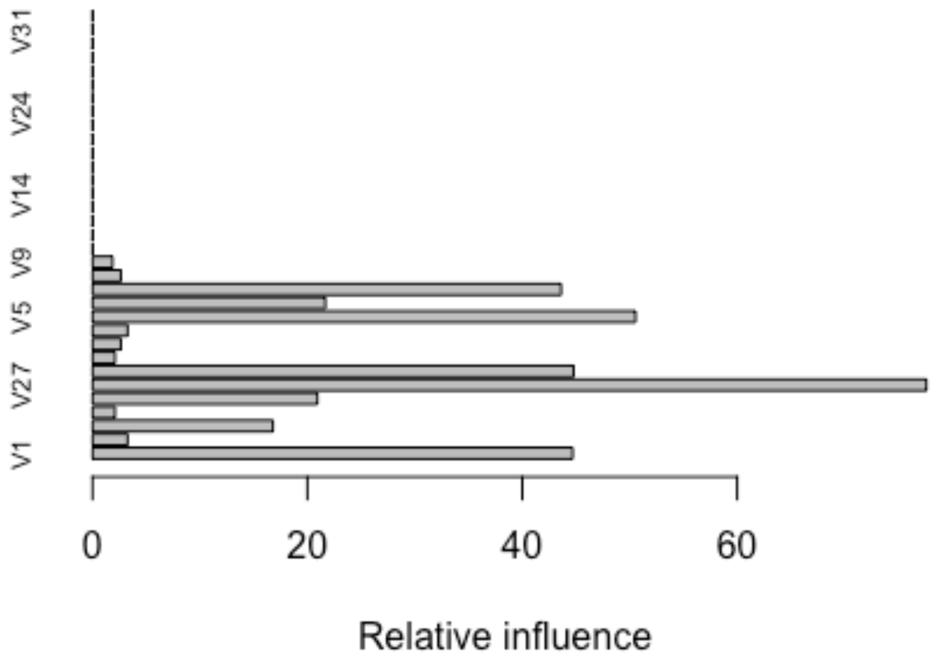
(b) Create a training and test dataset (70:30)

```
set.seed(153)
n_train <- round(0.7 * nrow(Ionosphere_Data))

#Create a vector of indices which is an 80% random sample
train_indices <- sample(1:nrow(Ionosphere_Data), n_train)

#Subset the credit data frame to training indices only
Ionosphere_train <- Ionosphere_Data[train_indices, ]

#Exclude the training indices to create the test set
Ionosphere_test <- Ionosphere_Data[-train_indices, ]
```

Decision Trees:
(c)
Create a decision tree for the train data with Class as the response and all of the other variables bar the variables discussed in (a) as predictors.
Is this a classification tree or a regression tree?

This is a classification tree.

```
model <- rpart(formula = Class ~ .,
               data = Ionosphere_train,
               method = "class")
print(model)

## n= 246
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 246 87 good (0.35365854 0.64634146)
##    2) V5< 0.145975 51  1 bad (0.98039216 0.01960784) *
##    3) V5>=0.145975 195 37 good (0.18974359 0.81025641)
##      6) V27>=0.999945 35  7 bad (0.80000000 0.20000000)
##       12) V21< 0.99242 15  0 bad (1.00000000 0.00000000) *
##       13) V21>=0.99242 20  7 bad (0.65000000 0.35000000)
##         26) V4< -0.517605 8  0 bad (1.00000000 0.00000000) *
##         27) V4>=-0.517605 12  5 good (0.41666667 0.58333333) *
##      7) V27< 0.999945 160  9 good (0.05625000 0.94375000) *

#Calculate variable importance in the model
varImp(model)

##       Overall
## V1  44.687839
## V16  3.266667
## V18 16.789575
## V21  2.100000
## V22 20.933029
## V27 77.614419
## V3  44.770345
## V33  2.100000
## V34  2.638462
## V4   3.266667
## V5  50.543656
## V6  21.670945
## V7  43.609879
## V8   2.638462
## V9   1.866667
## V10  0.000000
## V11  0.000000
## V12  0.000000
## V13  0.000000
## V14  0.000000
## V15  0.000000
## V17  0.000000
## V19  0.000000
## V20  0.000000
## V23  0.000000
## V24  0.000000
## V25  0.000000
## V26  0.000000
## V28  0.000000
## V29  0.000000
## V30  0.000000
## V31  0.000000
## V32  0.000000

#V10 to V32 are of low importance in this model
var_info=varImp(model)
barplot(var_info$Overall,horiz =TRUE, names.arg =
row.names(var_info),xlab="Relative influence",cex.names=0.7)
```



(d) Create a diagram of the decision tree created in (c).
Interpret the tree diagram.
Is this a useful visualisation for the data?

Yes this visualisation for the data is useful because we can clearly see what is classified bad or good from this tree diagram.

If V5 is less than 0.15 it is BAD.

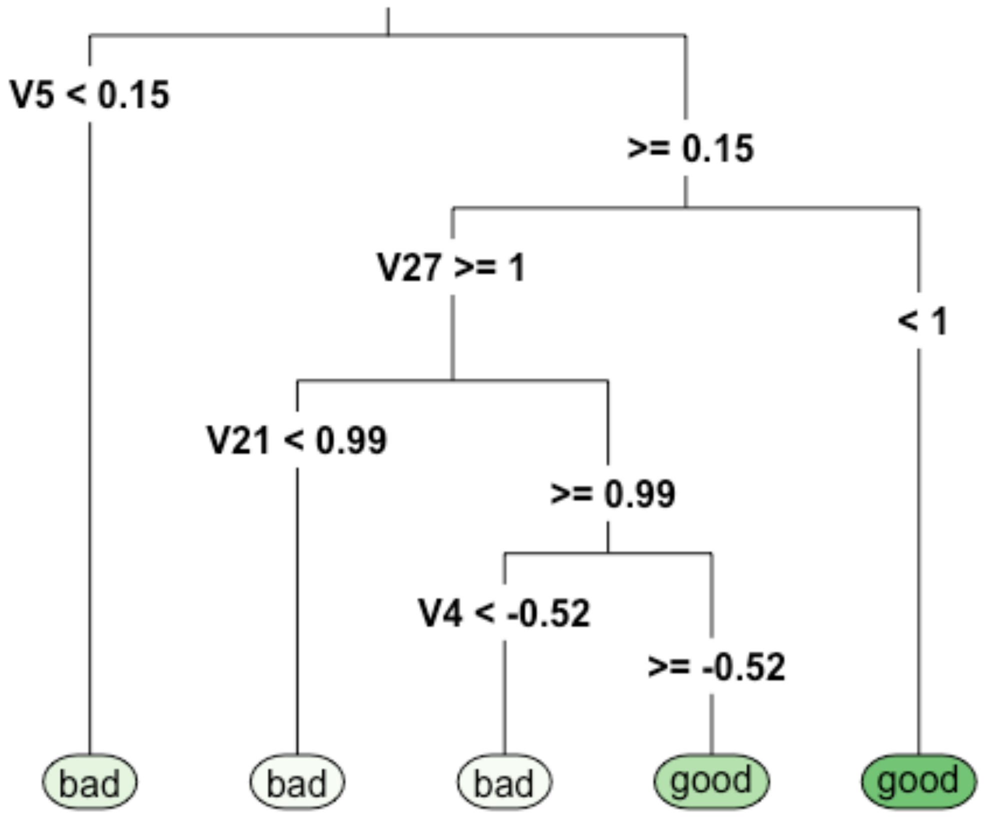
If V5 is greater than or equal to 0.15 AND V27 is greater than or equal to 1 AND V21 is less than 0.99 it is BAD.

If V5 is greater than or equal to 0.15 AND V27 is greater than or equal to 1 AND V21 is less greater than or equal 0.99 AND V4 is less than -0.52 it is BAD.

If V5 is greater than or equal to 0.15 AND V27 is greater than or equal to 1 AND V21 is less greater than or equal 0.99 AND V4 is greater than or equal to -0.52 it is GOOD.

If V5 is greater than or equal to 0.15 AND V27 is less than 1 it is GOOD.

```
#Diagram of Classification Decision Tree
fit <- rpart.plot(x = model,
           type = 3, extra = 0,
           box.palette = "Greens",
           fallen.leaves = TRUE)
```



```
fit

## $obj
## n= 246
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 246 87 good (0.35365854 0.64634146)
##    2) V5< 0.145975 51  1 bad (0.98039216 0.01960784) *
##    3) V5>=0.145975 195 37 good (0.18974359 0.81025641)
##      6) V27>=0.999945 35  7 bad (0.80000000 0.20000000)
##       12) V21< 0.99242 15  0 bad (1.00000000 0.00000000) *
##       13) V21>=0.99242 20  7 bad (0.65000000 0.35000000)
##         26) V4< -0.517605 8  0 bad (1.00000000 0.00000000) *
##         27) V4>=-0.517605 12  5 good (0.41666667 0.58333333) *
##      7) V27< 0.999945 160  9 good (0.05625000 0.94375000) *
##
## $snipped.nodes
## NULL
```

```
##
## $xlim
## [1] 0 1
##
## $ylim
## [1] 0 1
##
## $x
## [1] 0.38815132 0.07604896 0.70025368 0.45599966 0.29316364 0.61883568
0.51027833
## [8] 0.72739302 0.94450771
##
## $y
## [1] 1.03455806 0.01818845 0.80356496 0.57257187 0.01818845 0.34157878
0.01818845
## [8] 0.01818845 0.01818845
##
## $branch.x
##        [,1]       [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## x 0.3881513 0.07604896 0.7002537 0.4559997 0.2931636 0.6188357 0.5102783
##          NA 0.07604896 0.7002537 0.4559997 0.2931636 0.6188357 0.5102783
##          NA 0.38815132 0.3881513 0.7002537 0.4559997 0.4559997 0.6188357
##        [,8]      [,9]
## x 0.7273930 0.9445077
##   0.7273930 0.9445077
##   0.6188357 0.7002537
##
## $branch.y
##       [,1]       [,2]     [,3]      [,4]       [,5]      [,6]       [,7]
## y 1.034558 0.07187497 0.803565 0.5725719 0.07187497 0.3415788 0.07187497
##         NA 1.03455806 1.034558 0.8035650 0.57257187 0.5725719 0.34157878
##         NA 1.03455806 1.034558 0.8035650 0.57257187 0.5725719 0.34157878
##         [,8]       [,9]
## y 0.07187497 0.07187497
##   0.34157878 0.80356496
##   0.34157878 0.80356496
##
## $labs
## [1] NA     "bad"  NA     NA     "bad"  NA     "bad"  "good" "good"
##
## $cex
## [1] 1
##
## $boxes
## $boxes$x1
## [1]         NA 0.02710725         NA         NA 0.24422194         NA
0.46133663
## [8] 0.66757008 0.88468476
##
## $boxes$y1
## [1]           NA 0.0002929424           NA           NA 0.0002929424
## [6]           NA 0.0002929424 0.0002929424 0.0002929424
##
## $boxes$x2
## [1]        NA 0.1249907        NA        NA 0.3421053        NA 0.5592200
## [8] 0.7872160 1.0043307
##
## $boxes$y2
## [1]         NA 0.07187497         NA         NA 0.07187497         NA
0.07187497
## [8] 0.07187497 0.07187497
##
##
## $split.labs
## [1] ""
##
## $split.cex
## [1] 1 1 1 1 1 1 1 1 1
##
## $split.box
## $split.box$x1
## [1] -0.01151877          NA  0.37332324  0.19471468          NA
0.41619524
## [7]          NA          NA          NA
##
## $split.box$y1
## [1] 0.9200268        NA 0.6890337 0.4580406        NA 0.2270475        NA
## [8]        NA        NA
##
## $split.box$x2
## [1] 0.1636167        NA 0.5386761 0.3916126        NA 0.6043614        NA
## [8]        NA        NA
##
## $split.box$y2
## [1] 0.9916088        NA 0.7606157 0.5296227        NA 0.2986296        NA
## [8]        NA        NA
```

(e) Using print function or otherwise, answer the following about the decision tree created in part (c):
How many terminal nodes are there?

There are 5 terminal nodes.

What is the minimum number of observations in these terminal nodes?

V5 < 0.145975, 1 bad.
V21 < 0.99242, 0 bad.
V4 < -0.517605, 0 bad.
V4 >= -0.517605, 5 good.
V27 < 0.999945, 9 good.

```
print(model)

## n= 246
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 246 87 good (0.35365854 0.64634146)
##    2) V5< 0.145975 51  1 bad (0.98039216 0.01960784) *
##    3) V5>=0.145975 195 37 good (0.18974359 0.81025641)
##      6) V27>=0.999945 35  7 bad (0.80000000 0.20000000)
##       12) V21< 0.99242 15  0 bad (1.00000000 0.00000000) *
##       13) V21>=0.99242 20  7 bad (0.65000000 0.35000000)
##         26) V4< -0.517605 8  0 bad (1.00000000 0.00000000) *
##         27) V4>=-0.517605 12  5 good (0.41666667 0.58333333) *
##      7) V27< 0.999945 160  9 good (0.05625000 0.94375000) *

#There are 5 terminal nodes
```

(f) Use the predict function to find the predicted values for the test dataset. Create a confusion matrix.
What is the accuracy of this model?

Accuracy : 0.8667
Sensitivity : 0.9394
This indicates that it can correctly identify 93.94% of 'good' radar.
Correctly predicted 62 true positives.
Specificity : 0.7436
This indicates that it can correctly identify 74.36% of 'bad' radar.
Correctly predicted 29 true negatives.

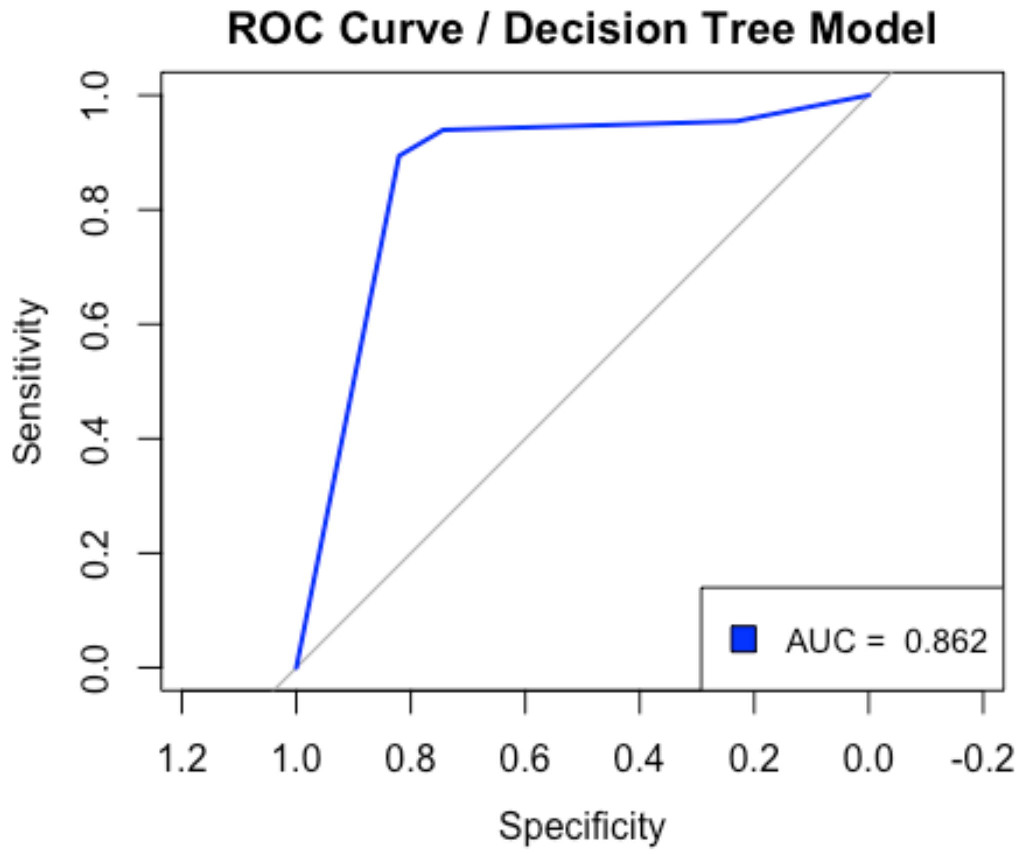
95% of the time our true accuracy should lie between 0.7864, 0.9251.

```
#Generate predicted classes for test data using the model
class_pred <- predict(object = model,  newdata = Ionosphere_test, type =
                            "class")
#Create confusion matrix with positive set to good
confusionMatrix(data = class_pred,  reference = Ionosphere_test$Class,
positive =
                      "good")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction bad good
##       bad   29    4
##       good  10   62
##
##                Accuracy : 0.8667
##                  95% CI : (0.7864, 0.9251)
##     No Information Rate : 0.6286
##      P-Value [Acc > NIR] : 5.206e-08
##
##                    Kappa : 0.7052
##
##   Mcnemar's Test P-Value : 0.1814
##
##              Sensitivity : 0.9394
##              Specificity : 0.7436
##           Pos Pred Value : 0.8611
##           Neg Pred Value : 0.8788
##               Prevalence : 0.6286
##           Detection Rate : 0.5905
##     Detection Prevalence : 0.6857
##        Balanced Accuracy : 0.8415
##
##         'Positive' Class : good
##

#Accuracy : 0.8667
#Sensitivity : 0.9394
#Specificity : 0.7436
```

(g) Create a ROC plot to show the sensitivity vs specificity of the model. Find the Area Under the Curve (AUC). Interpret this.

Area under the curve: 0.8619
This plot allows us to visually see the Accuracy, Sensitivity and Specificity as discussed in part (f).

```
#Generate predicted probabilities for bad, good using the model
levels(Ionosphere_Data$Class)

## [1] "bad"  "good"

tree_preds_prob <- predict(model, Ionosphere_test,
  type="prob")[, 2] #Probability of predicting 'good'
ROC <- roc(Ionosphere_test$Class, tree_preds_prob,
                               levels=c("bad","good"))

## Setting direction: controls < cases

#Plot the ROC curve
plot(ROC, col = 'blue', main = "ROC Curve / Decision Tree Model")
legend("bottomright",fill = c("blue"),
       legend = c(paste("AUC = ",round(ROC$auc,3))),
       cex = 0.9)
```



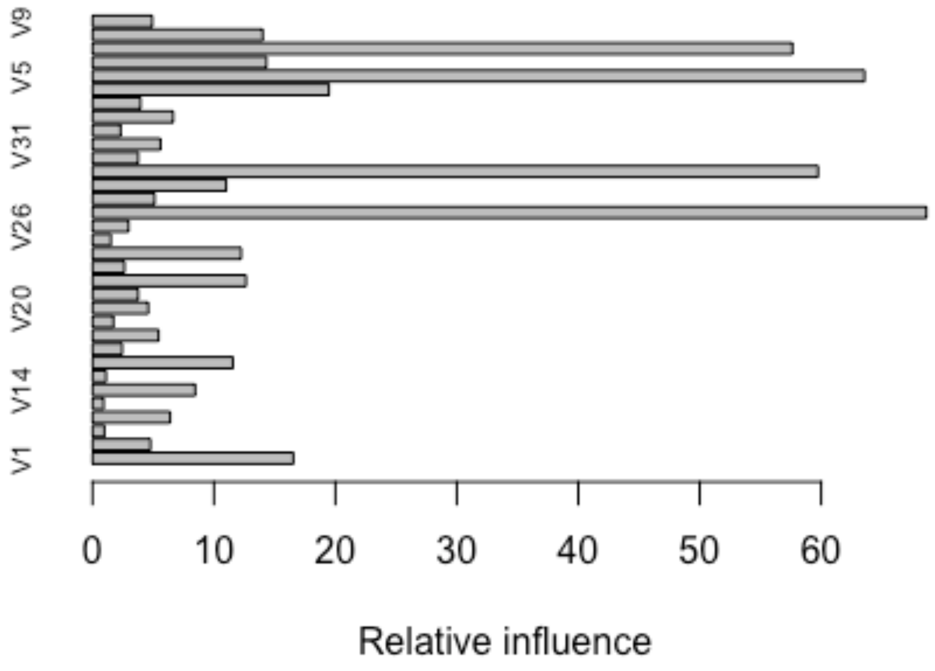
```
#Calculate the area under the curve (AUC)
ROC$auc

## Area under the curve: 0.8619

#Area under the curve: 0.8619
```

Ensemble techniques Trees:
(h)
Use the bagging function on the training data to predict Class.
What are the important variables used in this technique.

The most important variables using this technique are:
V27 with 68.66 relative influence
V5 with 63.56 relative influence
V3 with 59.7 relative influence
V7 with 57.63 relative influence

```
bagged_model <- bagging(formula = Class ~ .,
                        data = Ionosphere_train,
                        coob = TRUE)
varImp(bagged_model)
```

```
##        Overall
## V1  16.5365259
## V10  4.7711169
## V11  0.9741500
## V12  6.3948086
## V13  0.9121732
## V14  8.4550288
## V15  1.1125355
## V16 11.5543027
## V17  2.4426036
## V18  5.4409560
## V19  1.7183892
## V20  4.5875286
## V21  3.7621336
## V22 12.6403113
## V23  2.6329110
## V24 12.2397961
## V25  1.5072044
## V26  2.9723952
## V27 68.6674558
## V28  5.1154657
## V29 11.0084180
## V3  59.7407919
## V30  3.7651233
## V31  5.5978247
## V32  2.3137024
## V33  6.6412216
## V34  3.9560032
## V4  19.4654687
## V5  63.5667844
## V6  14.3277497
## V7  57.6357174
## V8  14.0140722
## V9   4.9237772

print(bagged_model)

##
## Bagging classification trees with 25 bootstrap replications
##
## Call: bagging.data.frame(formula = Class ~ ., data = Ionosphere_train,
##     coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.0813

var_info=varImp(bagged_model)
barplot(var_info$Overall,horiz =TRUE, names.arg =
row.names(var_info),xlab="Relative influence",cex.names=0.7)
```



(i) Use the predict function to find the predicted values for the test dataset using the model in (h).
Create a confusion matrix.
What is the accuracy of this model?

Accuracy : 0.9048
Sensitivity : 0.9545
This indicates that it can correctly identify 95.45% of 'good' radar.
Correctly predicted 63 true positives.
Specificity : 0.8205
This indicates that it can correctly identify 82.05% of 'bad' radar.
Correctly predicted 32 true negatives.

95% of the time our true accuracy should lie between 0.8318, 0.9534.

```
#Generate predicted classes for test data using the model
class_pred <- predict(object = bagged_model,  newdata = Ionosphere_test, type
=
                             "class")
#Create confusion matrix with positive set to good
confusionMatrix(data = class_pred,  reference = Ionosphere_test$Class,
positive =
                      "good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   32    3
##       good   7   63
##
##                Accuracy : 0.9048
##                  95% CI : (0.8318, 0.9534)
##     No Information Rate : 0.6286
##     P-Value [Acc > NIR] : 1.215e-10
##
##                   Kappa : 0.7917
##
##  Mcnemar's Test P-Value : 0.3428
##
##             Sensitivity : 0.9545
##             Specificity : 0.8205
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.9143
##              Prevalence : 0.6286
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.8875
##
##        'Positive' Class : good
##

#Accuracy : 0.9048
#Sensitivity : 0.9545
#Specificity : 0.8205
```

(j) Use the random forest technique on the training data to predict Class. What are the important variables used in this technique.

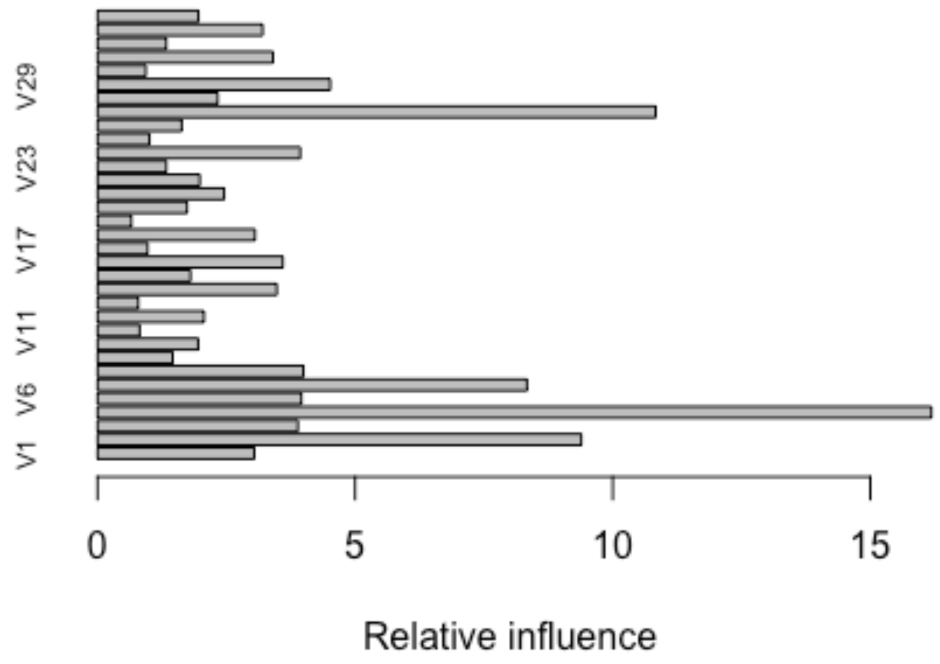
The most important variables using this technique are:
V5 with 16.18 relative influence
V27 with 10.83 relative influence V3 with 9.39 relative influence V7 with 8.33 relative influence

```
rf_model <- randomForest(formula =  Class ~ .,
                         data = Ionosphere_train)
varImp(rf_model)

##        Overall
## V1   3.0423277
## V3   9.3901724
## V4   3.8885327
## V5  16.1861974
## V6   3.9578136
## V7   8.3366238
## V8   3.9967703
## V9   1.4584536
## V10  1.9538362
## V11  0.8181773
## V12  2.0675333
## V13  0.7917586
## V14  3.4795721
## V15  1.8041556
## V16  3.5969873
## V17  0.9732101
## V18  3.0572294
## V19  0.6580511
## V20  1.7380827
## V21  2.4579302
## V22  1.9852248
## V23  1.3359386
## V24  3.9344047
## V25  1.0081049
## V26  1.6343229
## V27 10.8337276
## V28  2.3339952
## V29  4.5173081
## V30  0.9380787
## V31  3.4066585
## V32  1.3350524
## V33  3.2046634
## V34  1.9592511
```

```
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Class ~ ., data = Ionosphere_train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 6.91%
## Confusion matrix:
##      bad good class.error
## bad   77   10  0.11494253
## good   7  152  0.04402516
```

```
var_info=varImp(rf_model)
barplot(var_info$Overall,horiz =TRUE, names.arg =
row.names(var_info),xlab="Relative influence",cex.names=0.7)
```

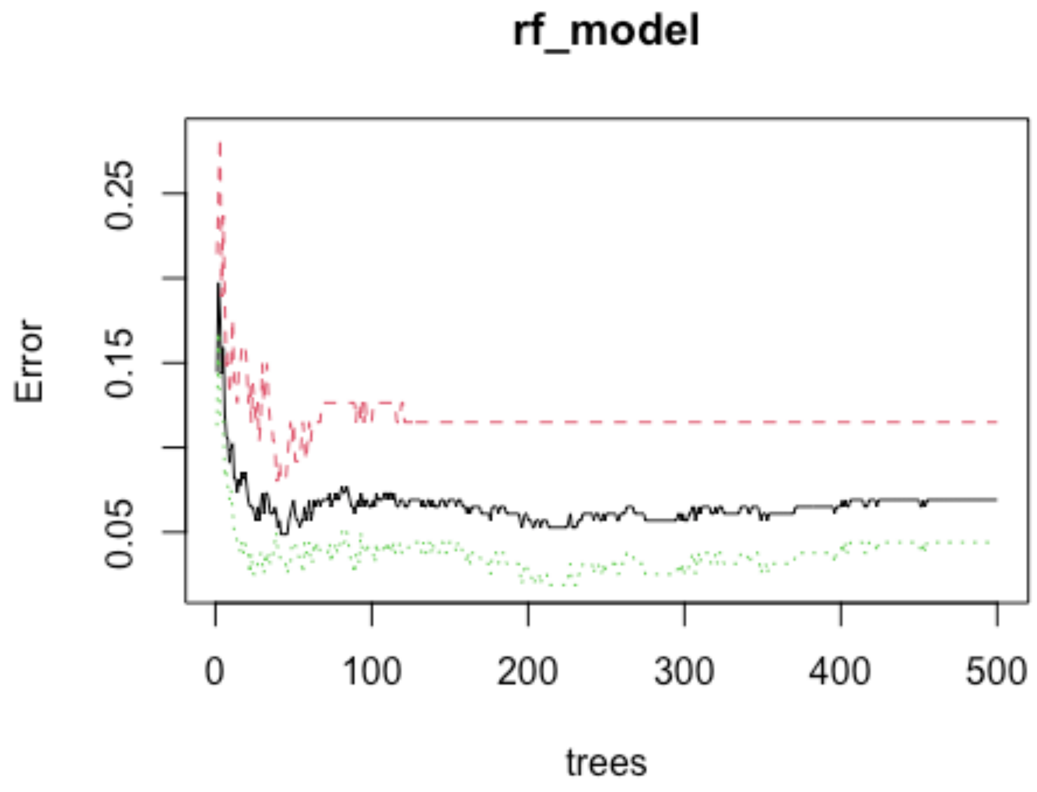


```
#All variables are important using this technique
```

(k) Use the plot function on your output of the randomForest function in (j). What does it tell you?

It looks as though error is the lowest with around 42 trees.
We can confirm this using the “which.min” function and call information from “err.rate” in our model.

```
plot(rf_model)
```



```
which.min(rf_model$err.rate[,1])

## [1] 42
```

(l) Predict the response for the test set and create the confusion matrix and calculate the accuracy.
How does it compare with the model obtained for bagging in part (h)?

Accuracy : 0.9143 Sensitivity : 0.9545
This indicates that it can correctly identify 95.45% of 'good' radar.
Correctly predicted 63 true positives.
Specificity : 0.8462
This indicates that it can correctly identify 84.62% of 'bad' radar.
Correctly predicted 33 true negatives.

95% of the time our true accuracy should lie between 0.8435, 0.9601.

This random forest model is better at predicting true negatives than our bagged model in part (i).

```
#Generate predicted classes for test data using the model
rf_class_pred <- predict(object = rf_model,  newdata = Ionosphere_test, type
=
                        "class")
```

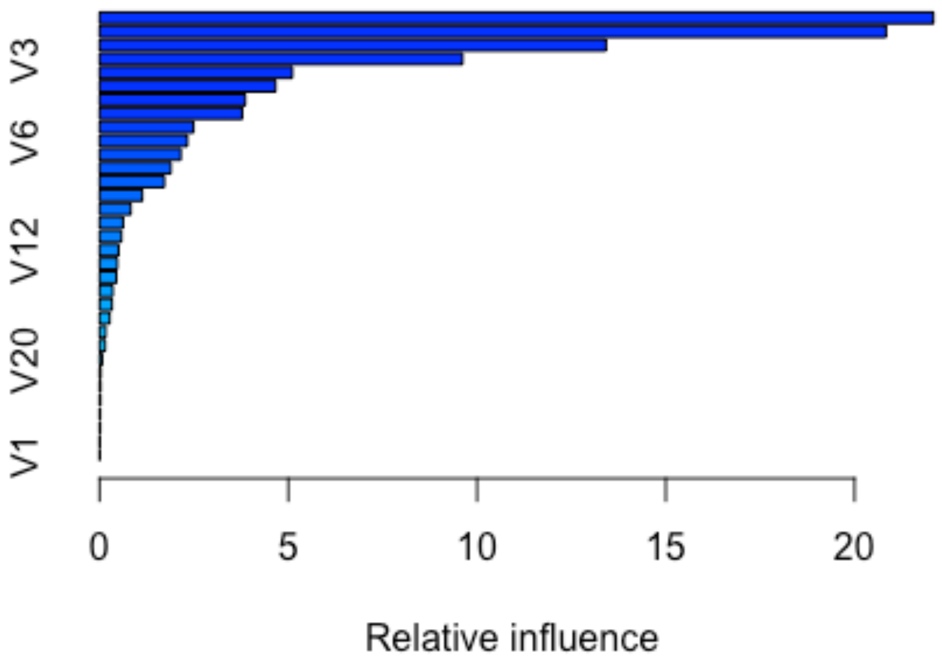
```
#Create confusion matrix with positive set to good
confusionMatrix(data = rf_class_pred,  reference = Ionosphere_test$Class,
positive =
                  "good")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction bad good
##       bad   33    3
##       good   6   63
##
##                Accuracy : 0.9143
##                  95% CI : (0.8435, 0.9601)
##     No Information Rate : 0.6286
##     P-Value [Acc > NIR] : 2.095e-11
##
##                   Kappa : 0.8135
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.9545
##             Specificity : 0.8462
##          Pos Pred Value : 0.9130
##          Neg Pred Value : 0.9167
##              Prevalence : 0.6286
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6571
##       Balanced Accuracy : 0.9003
##
##        'Positive' Class : good
##

#Accuracy : 0.9143
#Sensitivity : 0.9545
#Specificity : 0.8462
```

(m) Perform boosting on the training data to predict Class.
What are the important variables used in this technique.

The most important variables using this technique are: V5 with 22.08 relative influence
V27 with 20.83 relative influence
V7 with 13.4 relative influence
V3 with 9.6 relative influence

```
#Turn off scientific notation
options(scipen = 999)
#Needs to be set up as binary variable
Ionosphere_train_gbm <- Ionosphere_train
Ionosphere_train_gbm$Class<-(as.numeric(Ionosphere_train_gbm$Class)-1)
#head(Ionosphere_train_gbm)
```

```
Ionosphere_test_gbm <- Ionosphere_test
Ionosphere_test_gbm$Class<-(as.numeric(Ionosphere_test_gbm$Class)-1)
#head(Ionosphere_test_gbm)
#Train a 5000-tree GBM model
gbm_model <- gbm(formula = Class ~ ., distribution = "bernoulli", data =
Ionosphere_train_gbm, n.trees = 5000)
summary(gbm_model)
```



```
##      var      rel.inf
## V5    V5 22.0885411274
## V27  V27 20.8331307089
## V7    V7 13.4161592139
## V3    V3  9.6171457960
## V8    V8  5.1082532916
## V24  V24  4.6642740147
## V14  V14  3.8605577844
## V22  V22  3.7787436069
## V29  V29  2.4911078803
## V6    V6  2.3282780575
## V10  V10  2.1525870632
## V16  V16  1.8894027432
## V4    V4  1.7171087640
## V23  V23  1.1343916795
## V26 V26  0.8285797754
## V34 V34  0.6246932480
## V25 V25  0.5820653754
## V12 V12  0.5040808335
## V32 V32  0.4722947465
## V18 V18  0.4476011959
## V33 V33  0.3533848566
## V17 V17  0.3315964370
## V15 V15  0.2701875019
## V21 V21  0.1638976767
## V9   V9  0.1503393341
## V20 V20  0.0713077122
## V30 V30  0.0315728248
## V13 V13  0.0272807635
## V28 V28  0.0246781889
## V11 V11  0.0179858192
## V31 V31  0.0179556921
## V19 V19  0.0008162868
## V1   V1  0.0000000000
```

(n) Predict the response for the test set and create a confusion matrix.
What is the accuracy of this model?

Accuracy : 0.8952
Sensitivity : 0.9697
This indicates that it can correctly identify 96.97% of 'good' radar.
Correctly predicted 64 true positives.
Specificity : 0.7692
This indicates that it can correctly identify 76.92% of 'bad' radar.
Correctly predicted 30 true negatives.

95% of the time our true accuracy should lie between 0.8203, 0.9465.

How does it compare with the bagging and the random forest models?

The accuracy of this boosting model is very similar to the accuracy of the bagged and random forest model as it is close to 90%. This model predicts the true positives better than the other models but is less accurate when predicting true negatives.

```
#Generate predicted classes for test data using the model
gbm_class_pred <- predict(object = gbm_model,  newdata = Ionosphere_test_gbm,
type =
                        "response", n.trees=5000)
#Create confusion matrix with positive set to good
gbm_class_pred <- ifelse(gbm_class_pred < 0.99, 0, 1)
gbm_class_pred <- factor(gbm_class_pred, levels = c(0, 1), labels = c("bad",
"good"))

confusionMatrix(data = gbm_class_pred,  reference = Ionosphere_test$Class,
positive = "good")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   30    2
##       good   9   64
##
##                Accuracy : 0.8952
##                  95% CI : (0.8203, 0.9465)
##     No Information Rate : 0.6286
##     P-Value [Acc > NIR] : 0.0000000006349
##
##                   Kappa : 0.7671
##
##  Mcnemar's Test P-Value : 0.07044
##
##             Sensitivity : 0.9697
##             Specificity : 0.7692
##          Pos Pred Value : 0.8767
##          Neg Pred Value : 0.9375
##              Prevalence : 0.6286
##          Detection Rate : 0.6095
##    Detection Prevalence : 0.6952
##       Balanced Accuracy : 0.8695
##
##        'Positive' Class : good
##

#Accuracy : 0.8952
#Sensitivity : 0.9697
#Specificity : 0.7692
```

(o) Create a ROC plot to show the sensitivity vs specificity of all the models from the previous section.
Find the Area Under the Curve (AUC) for each. Interpret this.

Decision Tree Model
Area under the curve: 0.8619

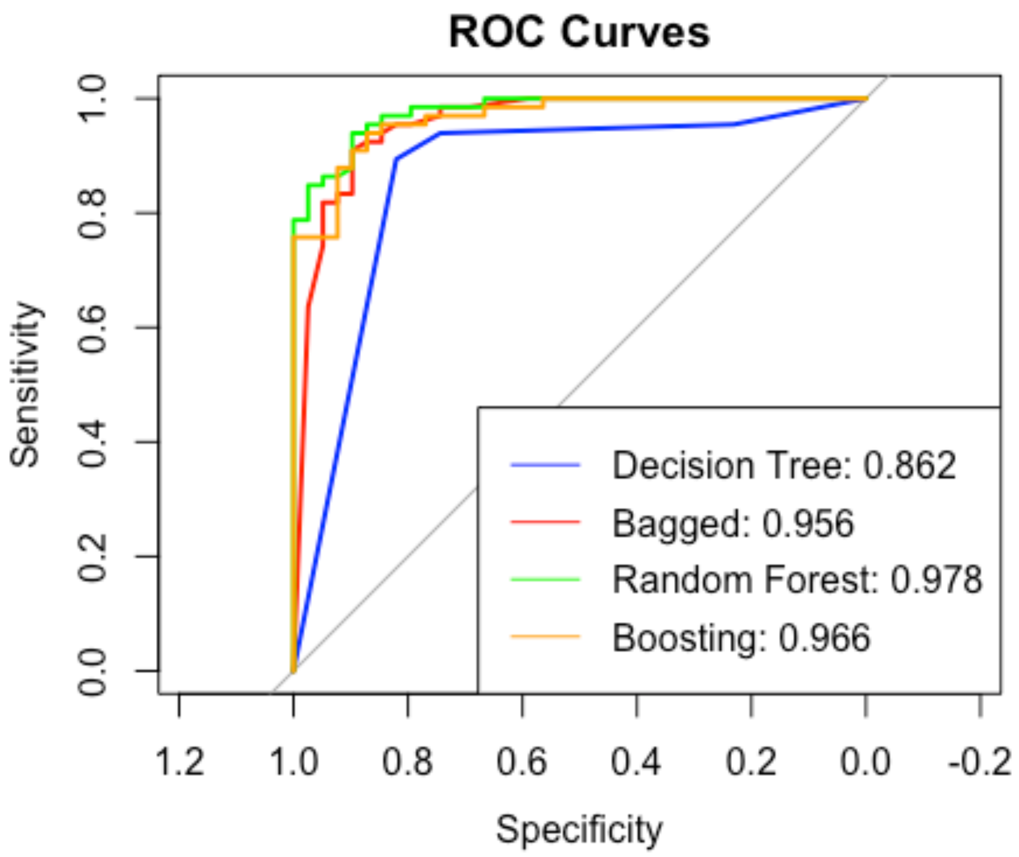
Bagged Model
Area under the curve: 0.9555

Random Forest Model
Area under the curve: 0.9777

Boosting Model
Area under the curve: 0.9662

The Random Forest Model is the better model as the area under the curve is closer to 1 which would be a perfect model.

```
#Generate predicted probabilities for bad, good using the model
levels(Ionosphere_Data$Class)

## [1] "bad"  "good"

tree_preds_prob <- predict(model, Ionosphere_test,
                           type="prob")[, 2] #Probability of predicting
'good'
roc1 <- roc(Ionosphere_test$Class, tree_preds_prob,
           levels=c("bad","good"))

## Setting direction: controls < cases

bagged_preds_prob <- predict(bagged_model, Ionosphere_test,
                           type="prob")[, 2] #Probability of predicting
'good'
roc2 <- roc(Ionosphere_test$Class, bagged_preds_prob,
           levels=c("bad","good"))

## Setting direction: controls < cases

rf_preds_prob <- predict(rf_model, Ionosphere_test,
                           type="prob")[, 2] #Probability of predicting
'good'
roc3 <- roc(Ionosphere_test$Class, rf_preds_prob,
           levels=c("bad","good"))

## Setting direction: controls < cases

gbm_preds_prob <- predict(gbm_model, Ionosphere_test_gbm,
                           type="response", n.trees=5000) #Probability of
predicting 'good'
roc4 <- roc(Ionosphere_test_gbm$Class, gbm_preds_prob,
           levels=c(0,1))

## Setting direction: controls < cases

# Plot the first ROC curve
plot(roc1, col = "blue", main = "ROC Curves")

# Add the second ROC curve to the same plot
lines(roc2, col = "red")
lines(roc3, col = "green")
lines(roc4, col = "orange")

# Add a legend to the plot
legend("bottomright", legend = c(paste("Decision Tree:",round(roc1$auc,3)),
paste("Bagged:", round(roc2$auc,3)), paste("Random Forest:",
round(roc3$auc,3)), paste("Boosting:", round(roc4$auc,3))), col = c("blue",
"red", "green", "orange"), lty = 1)
```

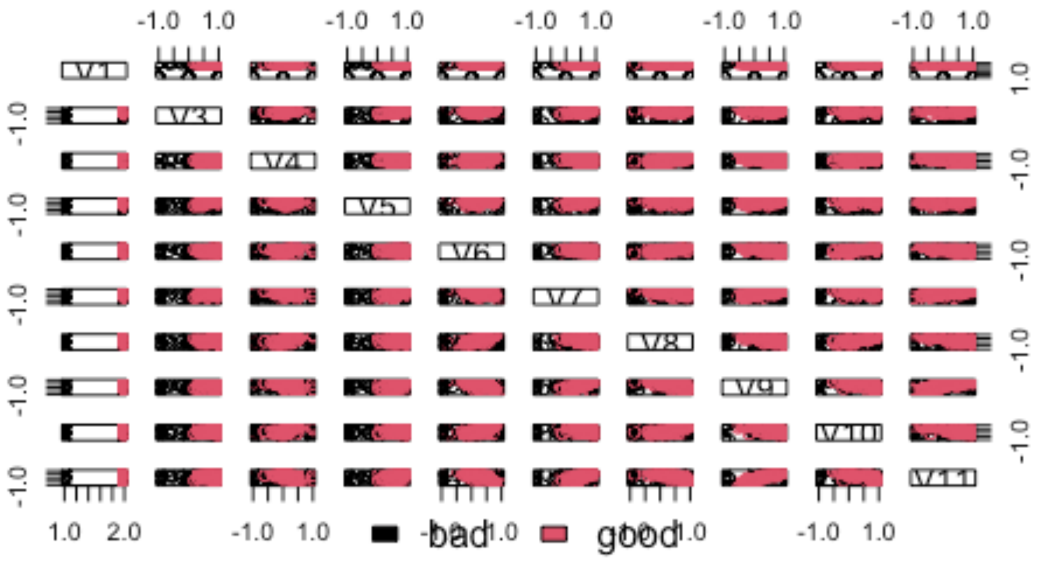


```
roc1$auc

## Area under the curve: 0.8619

roc2$auc

## Area under the curve: 0.9555

roc3$auc

## Area under the curve: 0.9777

roc4$auc

## Area under the curve: 0.9662
```

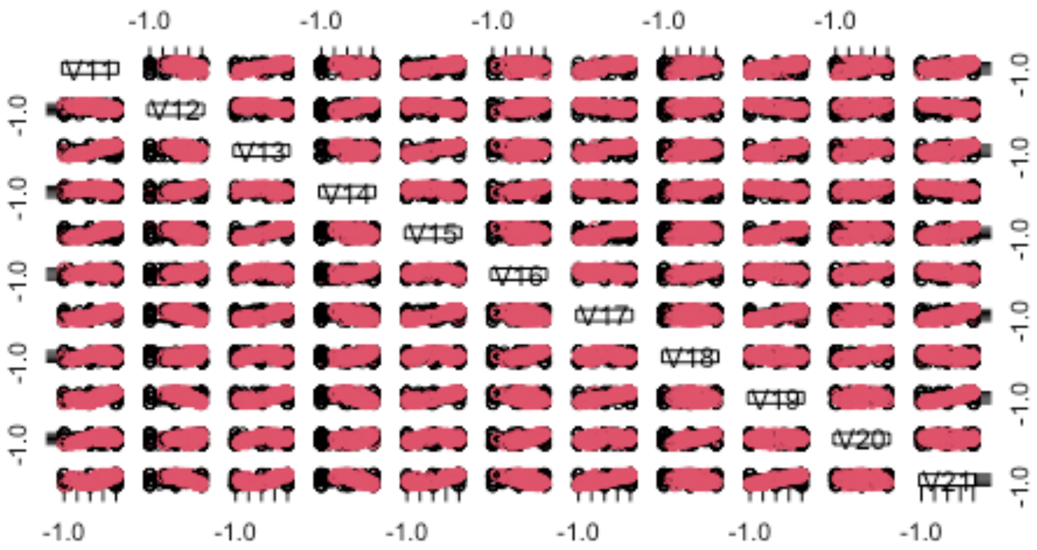
Support Vector Machines:
(p)
Have a look at the data briefly, do you think a linear or radial kernel is more appropriate given the visualizations.

Some of the ‘good’ radar seems linear when we look at the plots but the ‘good’ and ‘bad’ radar signals are not linear separable so radial kernel seems to be more appropriate.

```r
#PairPlots
pairs(Ionosphere_Data[,1:10],col=Ionosphere_Data$Class,oma=c(12,3,3,3))
par(xpd = TRUE)
legend("bottom", fill = 1:2,legend = c( levels(Ionosphere_Data$Class)),
       bty='n',ncol=2)
```



```r
pairs(Ionosphere_Data[,10:20],col=Ionosphere_Data$Class,oma=c(12,3,3,3))
par(xpd = TRUE)
legend("bottom", fill = 1:2,legend = c( levels(Ionosphere_Data$Class)),
       bty='n',ncol=2)
```



(q)

Use tune.svm to select the best hyperparameter values for the svm model with the kernel selected in part q.
Run this model on training dataset

```
tune_out <- tune.svm(x=Ionosphere_train[,1:33], y=Ionosphere_train$Class,
gamma=10^(-3:3), cost=c(0.01,0.1,1,10,100,1000), kernel="radial")

Ionosphere_train <- na.omit(Ionosphere_train)
#print best values of cost and gamma
tune_out$best.parameters$cost

## [1] 100

#10
tune_out$best.parameters$gamma

## [1] 0.1

#0.1

svm_model <- svm(Class~ ., data=Ionosphere_train, method="C-classification",
kernel="radial", cost=tune_out$best.parameters$cost,
gamma=tune_out$best.parameters$gamma)
```

```
#compute training accuracy
pred_train <- predict(svm_model, Ionosphere_train)
mean(pred_train == Ionosphere_train$Class)

## [1] 1

#1
```

(r) Predict the response for the test set and create a confusion matrix. What is the accuracy of this model?

Accuracy : 0.9048
Sensitivity : 0.9545
This indicates that it can correctly identify 95.45% of 'good' radar.
Correctly predicted 63 true positives.
Specificity : 0.8205
This indicates that it can correctly identify 82.05% of 'bad' radar.
Correctly predicted 32 true negatives.

95% of the time our true accuracy should lie between 0.8435, 0.9601.

```
pred_test <- predict(svm_model, Ionosphere_test)
mean(pred_test == Ionosphere_test$Class)

## [1] 0.9047619

#0.9047619

confusionMatrix(data = pred_test,  reference = Ionosphere_test$Class,
positive = "good")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   32    3
##       good   7   63
##
##                Accuracy : 0.9048
##                  95% CI : (0.8318, 0.9534)
##     No Information Rate : 0.6286
##     P-Value [Acc > NIR] : 0.0000000001215
##
##                   Kappa : 0.7917
##
##  Mcnemar's Test P-Value : 0.3428
##
##             Sensitivity : 0.9545
##             Specificity : 0.8205
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.9143
##                Prevalence : 0.6286
##            Detection Rate : 0.6000
##      Detection Prevalence : 0.6667
##         Balanced Accuracy : 0.8875
##
##          'Positive' Class : good
##

#Accuracy : 0.9048
#Sensitivity : 0.9545
#Specificity : 0.8205
```

Overall:
(s)
Based on all the models performed here and the different measures of performance, which of these techniques would you recommend, giving reasons.

Based on all the models performed here as it has the best accuracy and highest percentages for sensitivity and sensitivity and greater area under the curve in the ROC curve. I would recommend the random forest model.

Random Forest Model
Accuracy : 0.9143
Sensitivity : 0.9545
Specificity : 0.8462