Tree-Based Methods and SVMs Student

Anna Yeaton

Fall 2018

Lab Section

In this lab, we will go over tree-based regression, tree-based classification, bagging, random forest, boosting, and svm.

Perfomance Metrics

Classification Error rate for Tree-based Regression and Classification

The fraction of training observations in that region that do not belong to the most common class

$$E = 1 - max_k(\hat{p}_{mk})$$

Where $\hat{p}_{mk}$ is the proportion of the training observation in the mth region that are from the kth class.

Hyperparameter search in caret:

Grid search: The user can specify the values in the grid, and the system goes through each combination of parameters. Or the system will automatically do a grid search and the user just specifies the number of attributes in the grid.

```
#Creating grid
grid <- expand.grid(n.trees=c(10,20,50,100,500,1000),shrinkage=c(0.01,0.05,0.1,0.5),n.minobsinnode = c(3
```

Random Search: Specify random search in trainControl(), and specify the tune length in the train function.

Tree-Based Regression

Tree-Based regression uses recursive binary splitting, which is a top-down, greedy approach. The algorithm splits the predictor space into regions, with the objective of minimizing RSS within each region.

1. Create a tree
2. Use cross validation to find the optimal number of nodes for the tree
3. Prune tree accordingly

We will use the tree package for the tree based methods, and the quakes dataset for the regression methods.

```
library(tree)
data(quakes)

train_size <- floor(0.75 * nrow(quakes))
set.seed(543)
train_pos <- sample(seq_len(nrow(quakes)), size = train_size)

train_regression <- quakes[train_pos, ]
test_regression <- quakes[-train_pos, ]

dim(train_regression)
```

```
## [1] 750   5
```

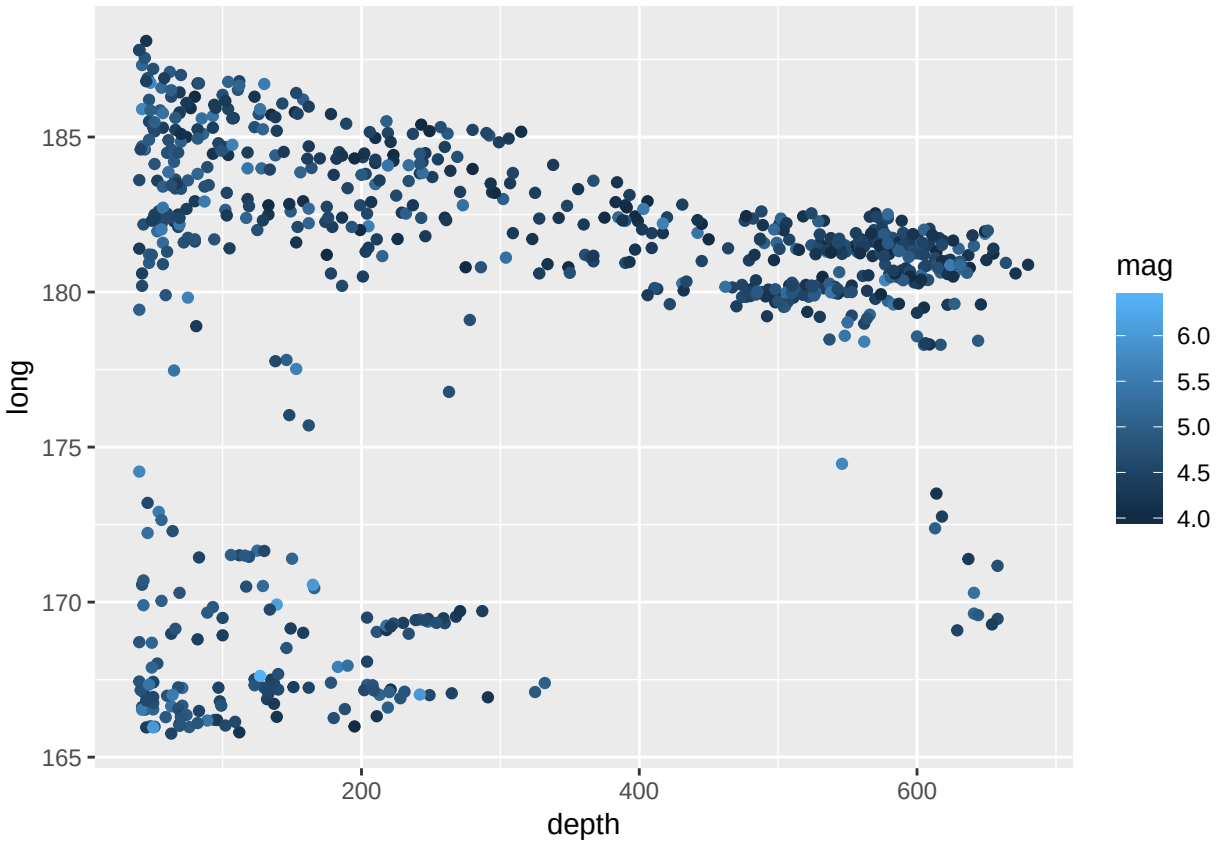
```
dim(test_regression)
```

```
## [1] 250   5
```

```
help("tree")
```

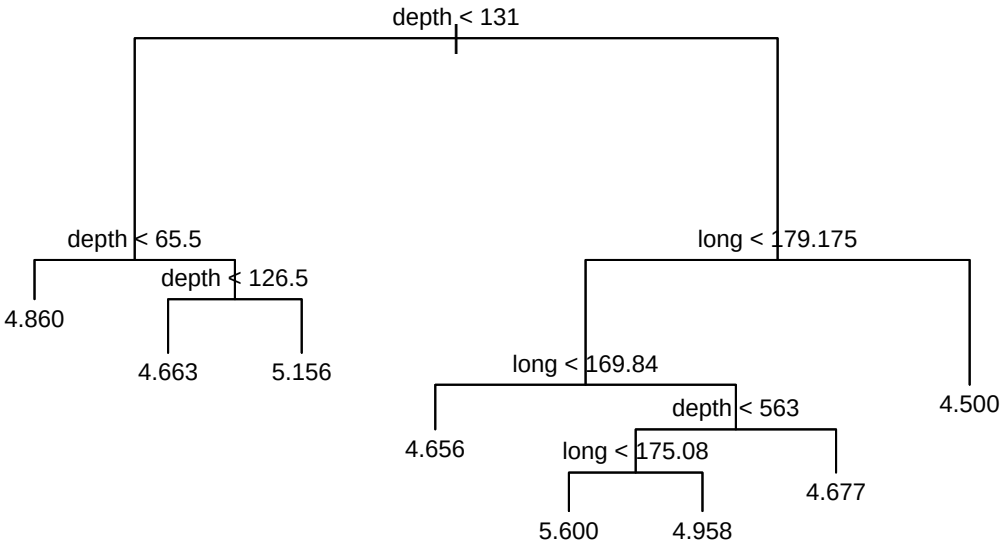
Visualize data

```
ggplot(data = train_regression) +
  geom_point(aes(x= depth, y = long, col = mag))
```



Use depth and longitude to predict magnitude

```
#create tree
set.seed(543)
regression_tree <- tree(mag ~ depth + long, data = train_regression)

plot(regression_tree)
text(regression_tree,cex=0.75)
```



```r
summary(regression_tree)
```

```
##
## Regression tree:
## tree(formula = mag ~ depth + long, data = train_regression)
## Number of terminal nodes:  8
## Residual mean deviance:  0.1369 = 101.6 / 742
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.75560 -0.27350 -0.06038  0.00000  0.19980  1.40000
```

Use cross validation to find the optimal number of nodes

```r
tree_complexity <- cv.tree(regression_tree, K = 15)

#the tree with 8 nodes has the lowest deviance
tree_complexity
```

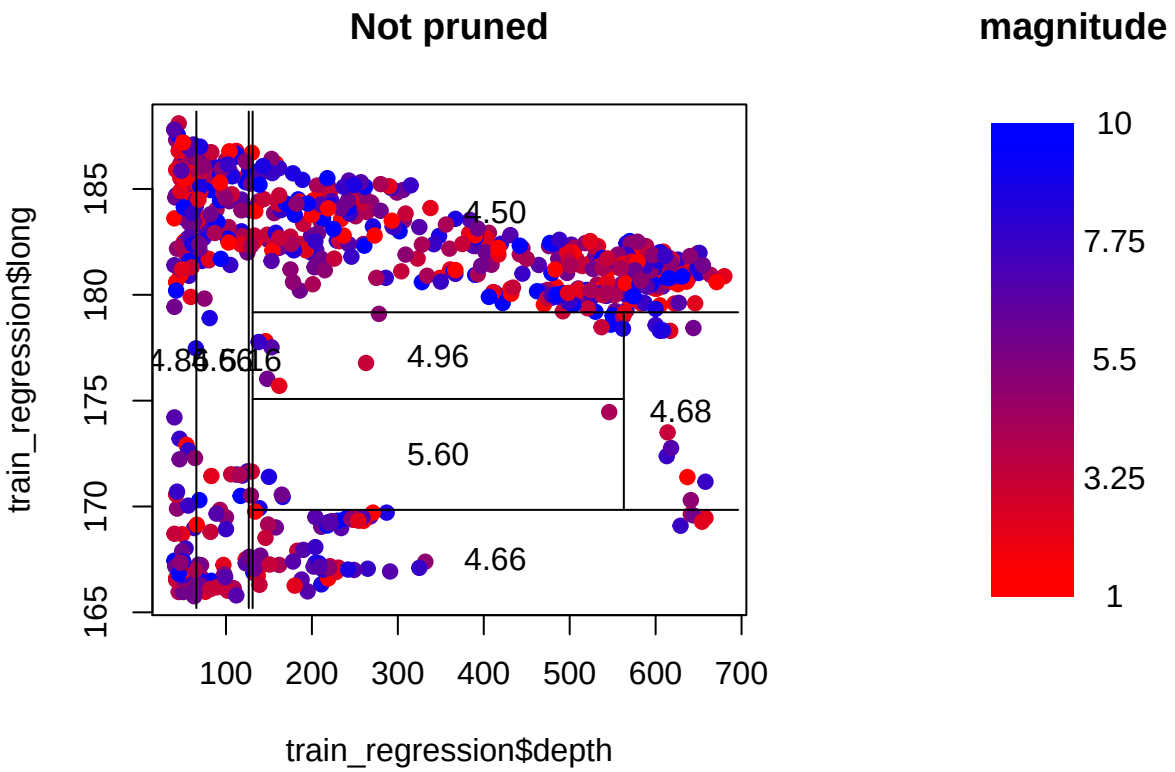
```
## $size
## [1] 8 7 6 5 3 2 1
##
## $dev
## [1] 115.7257 117.3762 116.9069 117.3161 117.1816 119.3149 124.8261
##
## $k
## [1]     -Inf 1.453186 1.628237 1.683036 1.750862 4.704675 8.361882
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Prune tree

```r
#prune.tree evaluates error of tree at various prunings and returns the best tree with X number of leav
tree_complexity_prune <- prune.tree(regression_tree, best = 8)
```

Visdualize partitions of the whole tree on the dataset

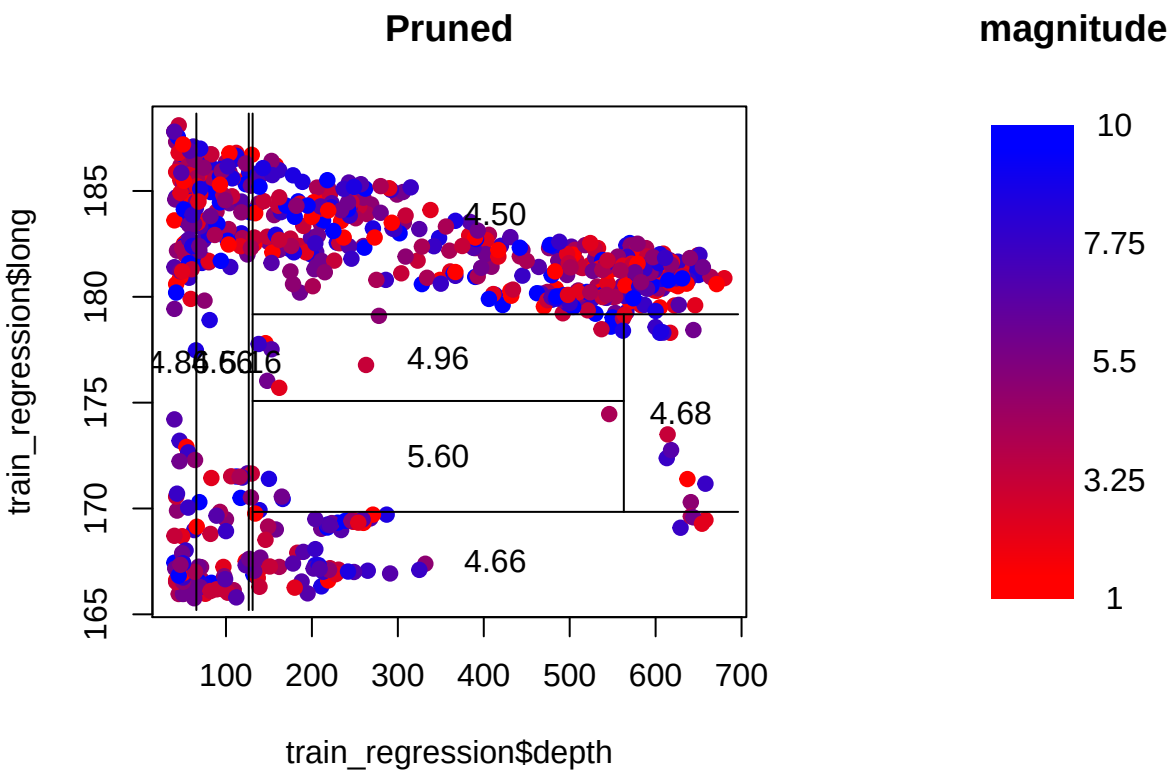
```r
layout(matrix(1:2,ncol=2), width = c(2,1),height = c(1,1))
cols <- colorRampPalette(c("blue", "red"), 100)
plot(x = train_regression$depth, y = train_regression$long, pch = 19, col =  cols(10), main = "Not prun
partition.tree(regression_tree,ordvars=c("depth","long"),add=TRUE)

legend_image <- as.raster(matrix(cols(10), ncol=1))
plot(c(0,2),c(0,1),type = 'n', axes = F,xlab = '', ylab = '', main = 'magnitude')
text(x=1.5, y = seq(0,1,l=5), labels = seq(1,10,l=5))
rasterImage(legend_image, 0, 0, 1,1)
```




```
#https://stackoverflow.com/questions/13355176/gradient-legend-in-base
```

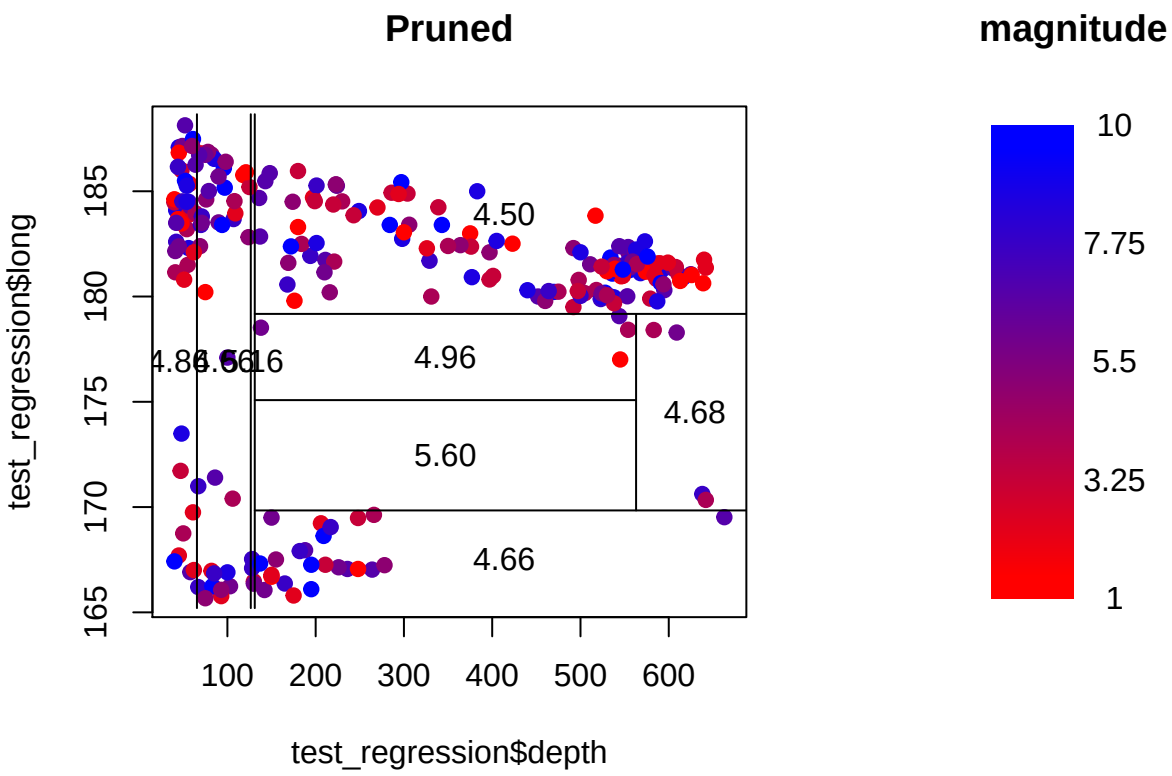
Partitions of the pruned tree on the dataset

```
layout(matrix(1:2,ncol=2), width = c(2,1),height = c(1,1))
cols <- colorRampPalette(c("blue", "red"), 100)
plot(x = train_regression$depth, y = train_regression$long, pch = 19, col =  cols(10), main = "Pruned")
partition.tree(tree_complexity_prune,ordvars=c("depth","long"),add=TRUE)

legend_image <- as.raster(matrix(cols(10), ncol=1))
plot(c(0,2),c(0,1),type = 'n', axes = F,xlab = '', ylab = '', main = 'magnitude')
text(x=1.5, y = seq(0,1,l=5), labels = seq(1,10,l=5))
rasterImage(legend_image, 0, 0, 1,1)
```



Visualize tree on test set

```
layout(matrix(1:2,ncol=2), width = c(2,1),height = c(1,1))
cols <- colorRampPalette(c("blue", "red"), 100)
plot(x = test_regression$depth, y = test_regression$long, pch = 19, col =  cols(10), main = "Pruned")
partition.tree(tree_complexity_prune,ordvars=c("depth","long"),add=TRUE)

legend_image <- as.raster(matrix(cols(10), ncol=1))
plot(c(0,2),c(0,1),type = 'n', axes = F,xlab = '', ylab = '', main = 'magnitude')
text(x=1.5, y = seq(0,1,l=5), labels = seq(1,10,l=5))
rasterImage(legend_image, 0, 0, 1,1)
```


Pruned
magnitude
10
7.75
5.5
3.25
1
test_regression$long
165
170
175
180
185
100
200
300
400
500
600
test_regression$depth
4.50
4.96
4.68
5.60
4.66

Tree-based Classification

Tree-based classification also utilizes recursive binary splitting, but instead of minimizing RSS, the goal is to maximize node purity. Measures of node purity include Gini, Cross Entropy, and classification error.

1. Create a tree
2. Use cross validation to find the optimal number of nodes for the tree
3. Prune tree accordingly

```
library(mlbench)
data(BreastCancer)

train_size <- floor(0.75 * nrow(BreastCancer))
set.seed(543)
train_pos <- sample(seq_len(nrow(BreastCancer)), size = train_size)

BreastCancer1 <- transform(BreastCancer, Id = as.numeric(Id), Cl.thickness = as.numeric(Cl.thickness),
                           Cell.size = as.numeric(Cell.size),
                           Cell.shape = as.numeric(Cell.shape), Marg.adhesion = as.numeric(Marg.adhesio
                           Epith.c.size = as.numeric(Epith.c.size),
                           Bare.nuclei = as.numeric(Bare.nuclei), Bl.cromatin = as.numeric(Bl.cromatin)
                           Normal.nucleoli = as.numeric(Normal.nucleoli),
                           Mitoses = as.numeric(Mitoses))

train_classification <- BreastCancer1[train_pos, ]
test_classification <- BreastCancer1[-train_pos, ]

dim(train_classification)
```

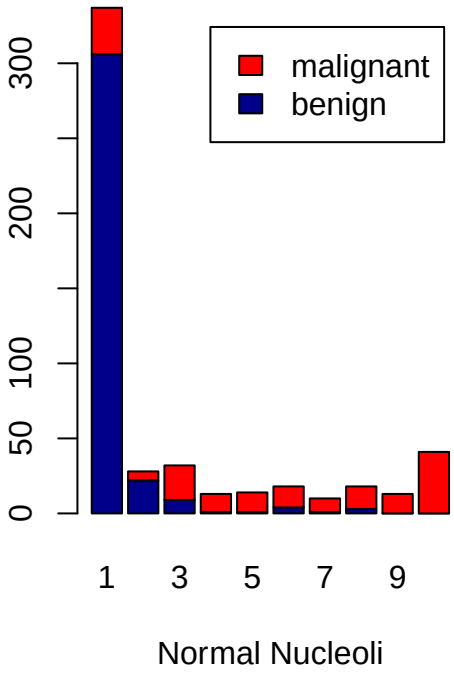
```
## [1] 524  11
```

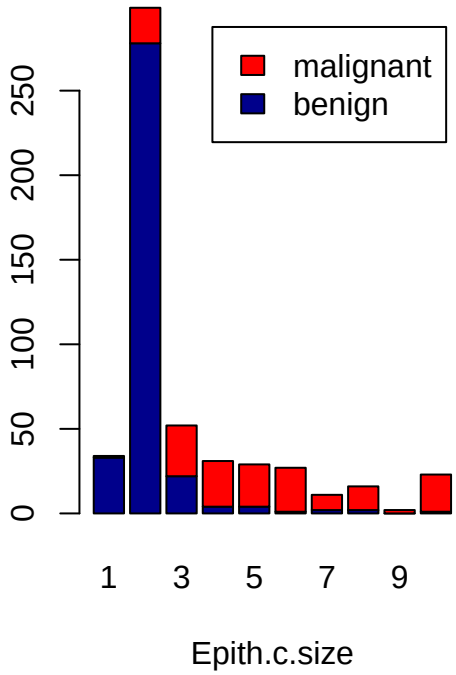
```
dim(test_classification)
```

```
## [1] 175  11
```

Visualize data

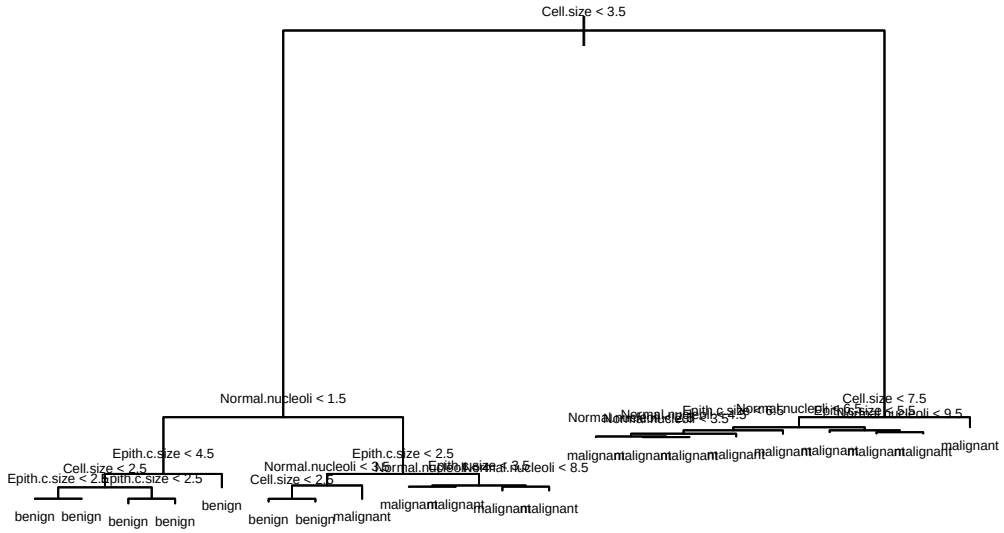
```
par(mfrow= c(1,2))

barplot(table(train_classification$Class,factor(train_classification$Normal.nucleoli)),
  xlab="Normal Nucleoli", col=c("darkblue","red"),
    legend = rownames(table(train_classification$Class,train_classification$Normal.nucleoli)))

barplot(table(train_classification$Class,factor(train_classification$Epith.c.size)),
  xlab="Epith.c.size", col=c("darkblue","red"),
    legend = rownames(table(train_classification$Class,train_classification$Epith.c.size)))
```





Create a classification tree

```
set.seed(30495)
classification_tree <- tree(Class ~ Normal.nucleoli + Epith.c.size + Cell.size, data = train_classifica
plot(classification_tree)
text(classification_tree,cex=0.45)
```




```
summary(classification_tree)
```

```
##
## Classification tree:
## tree(formula = Class ~ Normal.nucleoli + Epith.c.size + Cell.size,
##     data = train_classification, split = "gini")
## Number of terminal nodes:  21
## Residual mean deviance:  0.247 = 124.2 / 503
## Misclassification error rate: 0.05344 = 28 / 524
```

Test this tree on the test set

```
classification_test <- predict(classification_tree, newdata = test_classification, type = "class")

confusionMatrix(classification_test, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       104         2
##   malignant      7        62
##
##                Accuracy : 0.9486
##                  95% CI : (0.9046, 0.9762)
##     No Information Rate : 0.6343
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.891
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.9369
##             Specificity : 0.9688
##          Pos Pred Value : 0.9811
##          Neg Pred Value : 0.8986
##              Prevalence : 0.6343
##          Detection Rate : 0.5943
##    Detection Prevalence : 0.6057
##       Balanced Accuracy : 0.9528
##
##        'Positive' Class : benign
##
```

Fit the tree using cross validation. Use FUN = prune.misclass to indicate we want to classification error to guide cross val and pruning.

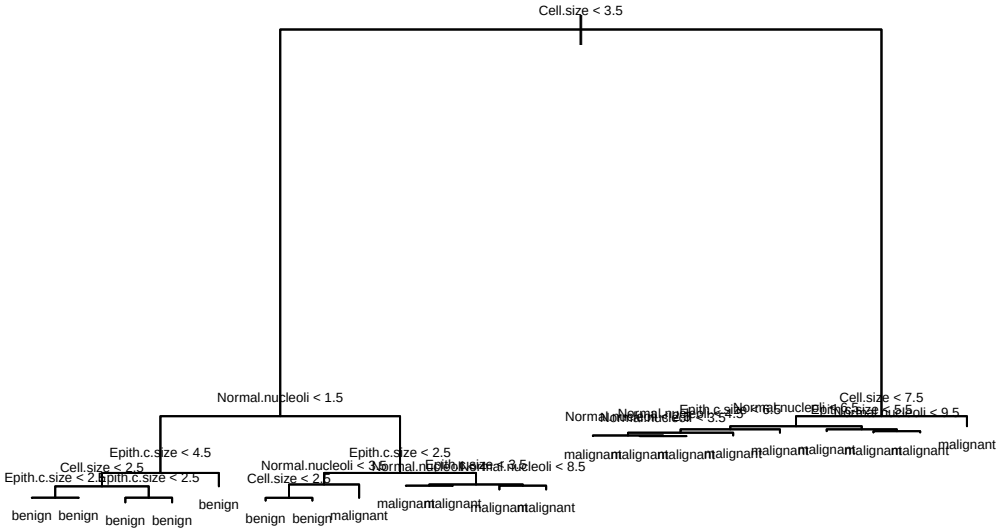
```
fit_classification_tree <- cv.tree(classification_tree,FUN=prune.misclass, K = 15)
```

```
fit_classification_tree
```

```
## $size
## [1] 21  5  4  2  1
##
## $dev
## [1]  35  35  36  36 177
##
## $k
## [1]  -Inf   0.0   3.0   3.5 139.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Now prune the tree

```
prune_classification_tree=prune.misclass(classification_tree, best=13)

plot(prune_classification_tree)
text(prune_classification_tree,cex=0.45)
```



Test the pruned tree on the test set

```
classification_test_pruned <- predict(prune_classification_tree, newdata = test_classification, type =

confusionMatrix(classification_test_pruned, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       103         2
##   malignant      8        62
##
##                Accuracy : 0.9429
##                  95% CI : (0.8974, 0.9723)
##     No Information Rate : 0.6343
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8792
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9279
##             Specificity : 0.9688
##          Pos Pred Value : 0.9810
##          Neg Pred Value : 0.8857
##              Prevalence : 0.6343
##          Detection Rate : 0.5886
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.9483
##
##        'Positive' Class : benign
##
```

Bagged classification Tree

Bagging trees, or bootstrap aggregating, combines predictions from multiple algorithms together to create a more accurate model in a majority vote fashion.

We will use the random forest package to do bagging. We can do this because we explicitly state that we want to use all of the available variables.

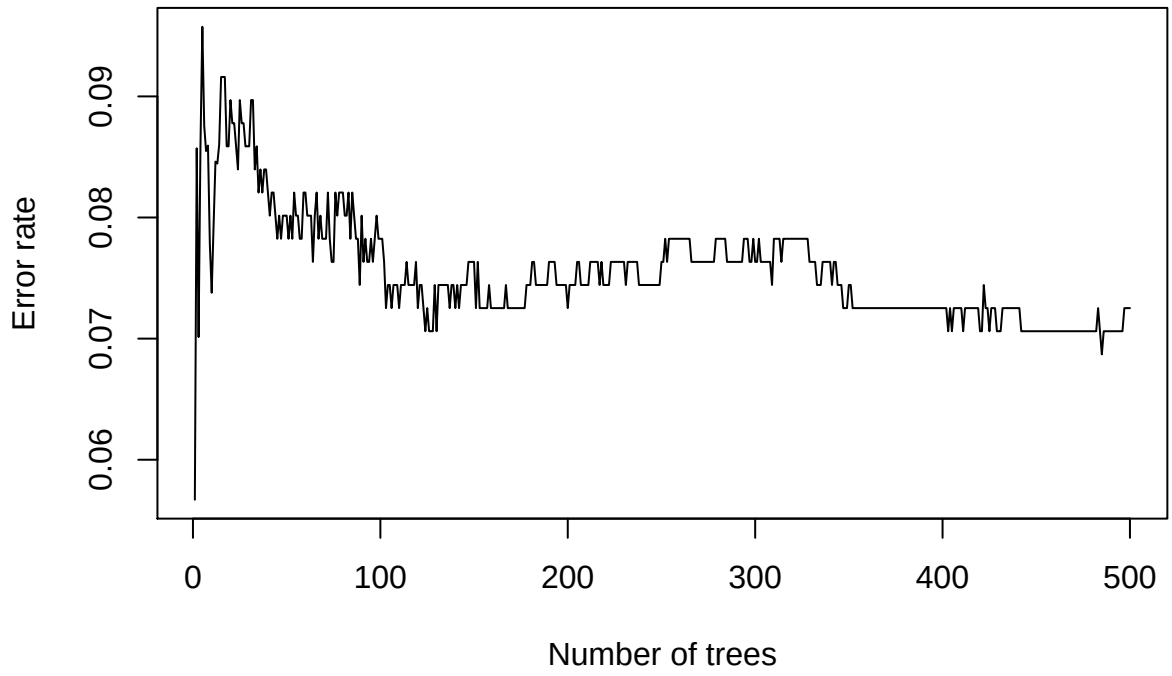
```
set.seed(30495)
bag_classification <- randomForest(Class ~ Normal.nucleoli+ Epith.c.size + Cell.size , data=train_classi
                                   mtry = 3, importance = TRUE, oob.times = 15, confusion = T)
```

```
bag_classification
```

```
##
## Call:
##  randomForest(formula = Class ~ Normal.nucleoli + Epith.c.size +      Cell.size, data = train_classi
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 7.25%
## Confusion matrix:
##           benign malignant class.error
## benign       327        20  0.05763689
## malignant     18       159  0.10169492
```

Vizualize OOB

```
plot(bag_classification$err.rate[,1], type = "l", xlab = "Number of trees", ylab = "Error rate")
```



Look at importance of features

```
importance(bag_classification)
```

```
##                    benign malignant MeanDecreaseAccuracy MeanDecreaseGini
## Normal.nucleoli  31.83166 15.293157             34.60502         27.55927
## Epith.c.size     18.42582 -2.952274             17.47610         14.91003
## Cell.size       127.67794 42.404886            108.59585        179.54813
```

Test bagged tree on test data

```
test_bag_classification <- predict (bag_classification , newdata =test_classification)

confusionMatrix(test_bag_classification, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       104         1
##   malignant      7        63
##
##                Accuracy : 0.9543
##                  95% CI : (0.9119, 0.9801)
##     No Information Rate : 0.6343
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9034
##  Mcnemar's Test P-Value : 0.0771
##
##             Sensitivity : 0.9369
##             Specificity : 0.9844
##          Pos Pred Value : 0.9905
##          Neg Pred Value : 0.9000
##              Prevalence : 0.6343
##          Detection Rate : 0.5943
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.9607
##
##        'Positive' Class : benign
##
```

Random Forest

Random forest is similar to bagging, but the number of features available is less than the total number of features, often 1/3n or sqrt(n). This allows weaker learners more voice. This also helps de correlate the trees.

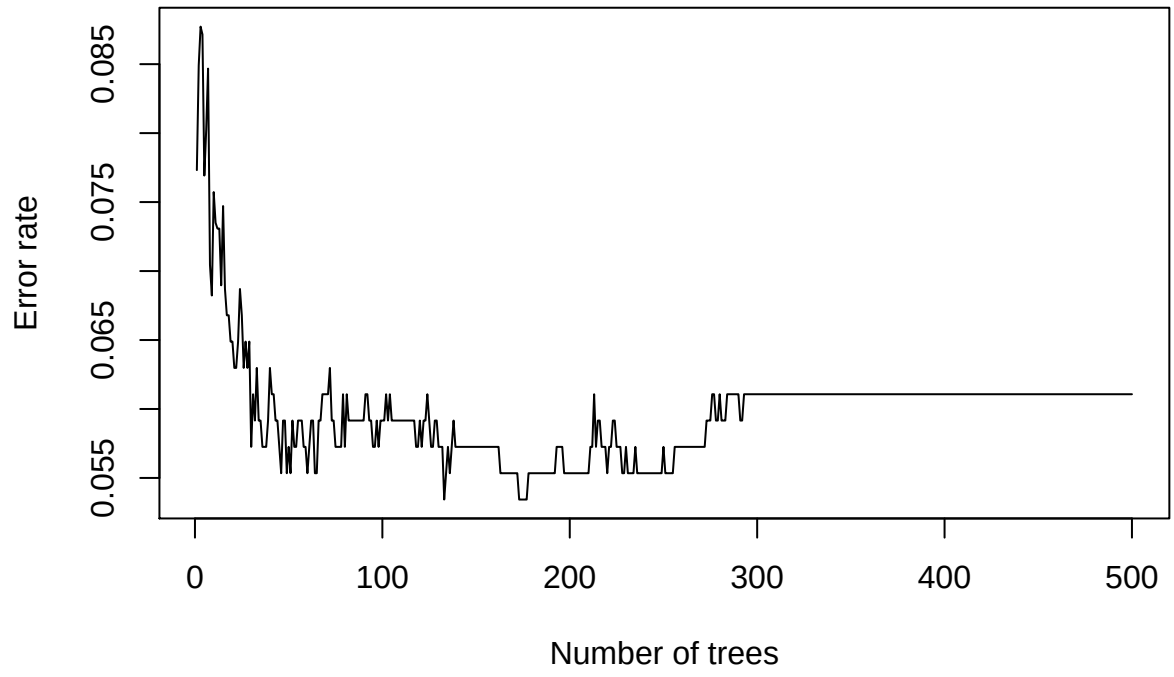
```
set.seed(30495)
#do not specify mtry. The default for classification is sqrt(p) where p is the number of variables
RF_classification <- randomForest(Class ~ Normal.nucleoli + Epith.c.size + Cell.size , data=train_classi
```

```
RF_classification
```

```
##
## Call:
##  randomForest(formula = Class ~ Normal.nucleoli + Epith.c.size +      Cell.size, data = train_classi
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 6.11%
## Confusion matrix:
##           benign malignant class.error
## benign       328        19  0.05475504
## malignant     13       164  0.07344633
```

Visualize OOB error rate

```
plot(RF_classification$err.rate[,1], type = "l", ylab = "Error rate", xlab = "Number of trees")
```



Visualize importance of features

```
importance(RF_classification)
```

```
##                   benign malignant MeanDecreaseAccuracy MeanDecreaseGini
## Normal.nucleoli 24.49552 13.354093             27.02063         62.29779
## Epith.c.size    15.55308  8.478846             17.64613         56.64914
```

```
## Cell.size       39.07064 35.548773             48.57648         83.66063
```

Predict using test set

```
test_RF_classification <- predict (RF_classification , newdata =test_classification)

confusionMatrix(test_RF_classification, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       104         1
##   malignant      7        63
##
##                Accuracy : 0.9543
##                  95% CI : (0.9119, 0.9801)
##     No Information Rate : 0.6343
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9034
##  Mcnemar's Test P-Value : 0.0771
##
##             Sensitivity : 0.9369
##             Specificity : 0.9844
##          Pos Pred Value : 0.9905
##          Neg Pred Value : 0.9000
##              Prevalence : 0.6343
##          Detection Rate : 0.5943
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.9607
##
##        'Positive' Class : benign
##
```

Boosting

Boosting is a method that slowly adds trees to the existing model.

- Boosting can overfit. Contraints to impose on the model to reduce overfitting include:
- Constraints on the trees. Such as the number of trees, and tree depth
- Constraints on the data used. Stochastic Gradient Boosting randomly samples a subset of the data to create trees. This helps reduce correlations in the trees.
- Regularize. Weights such as leaf weights can be regularized
- Shrinkage, or learning rate. Reduce the rate of learning by adding predictions of trees sequentially.

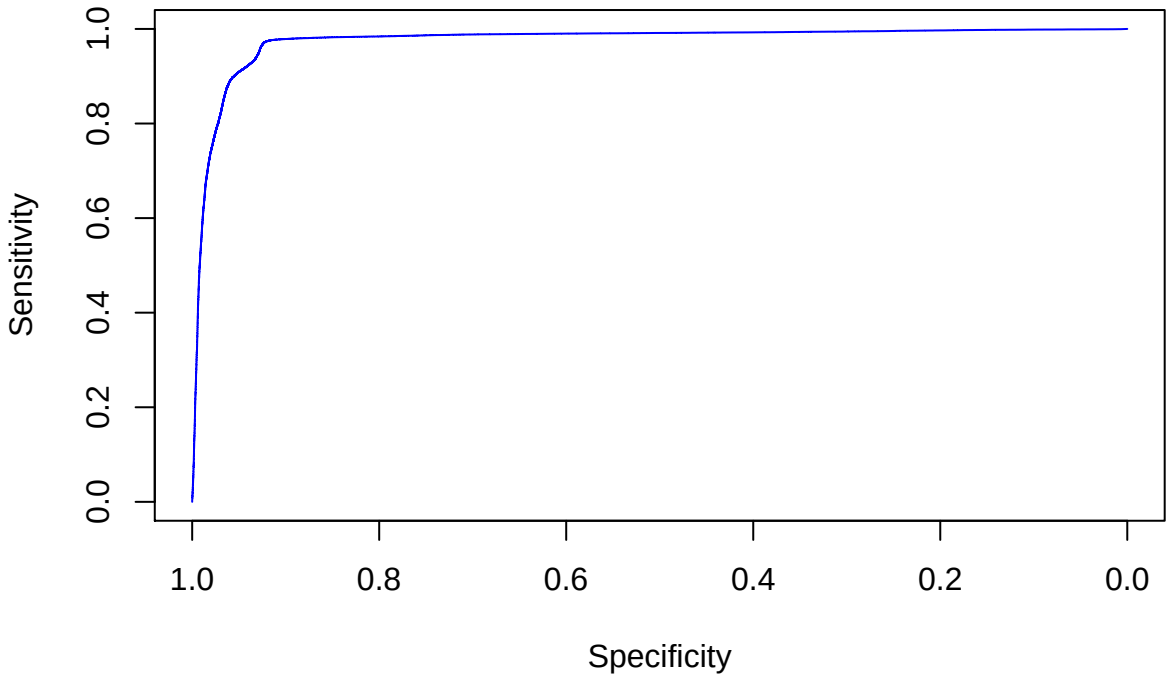
```
set.seed(30495)
ctrl <- trainControl(method = "repeatedcv", repeats = 2, classProbs = T, savePredictions = T)
# training the model
model_gbm<-train(Class ~ Normal.nucleoli + Epith.c.size + Cell.size, data = train_classification , metho
```

```
roc(predictor = model_gbm$pred$malignant, response = model_gbm$pred$obs)$auc
```

```
## Area under the curve: 0.9755
```

Visualize ROC curve

```
plot(x = roc(predictor = model_gbm$pred$malignant, response = model_gbm$pred$obs)$specificities, y = roc
```



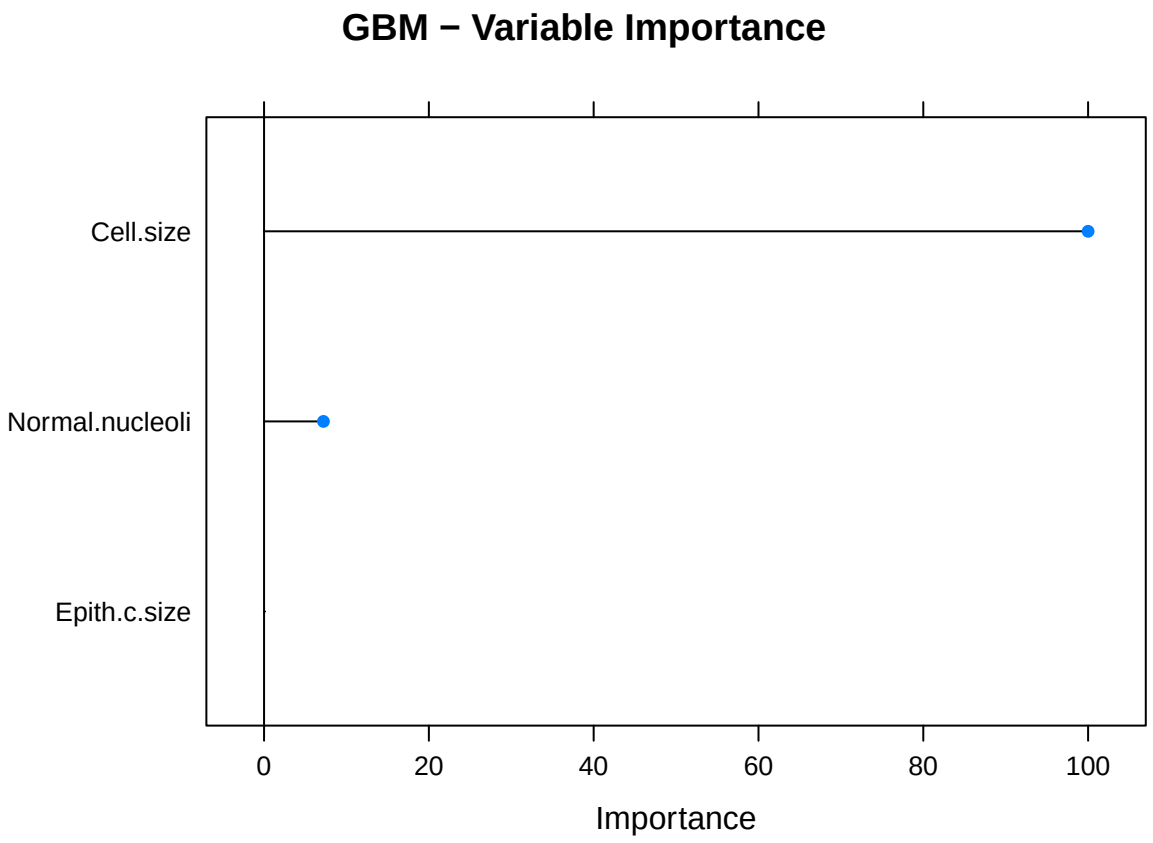
Visualize importance of features

```
varImp(object = model_gbm)
```

```
## gbm variable importance
##
##                 Overall
## Cell.size       100.000
## Normal.nucleoli   7.204
```

```
## Epith.c.size      0.000
```

```
plot(varImp(object=model_gbm),main="GBM - Variable Importance")
```



Support Vector Machine

SVMs are models that transform the data so that they can be linearly separated, and linearly separates them in a way that maximizes the decision boundary.

Supplemental information :https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-16-learning-support-vector-machines/

Train linear SVM

```
set.seed(30495)
ctrl <- trainControl(method = "repeatedcv", repeats = 5,classProbs = T, savePredictions = T)
svm <- train(Class ~ Normal.nucleoli + Epith.c.size + Cell.size , data = train_classification, method =
```

```
svm
```

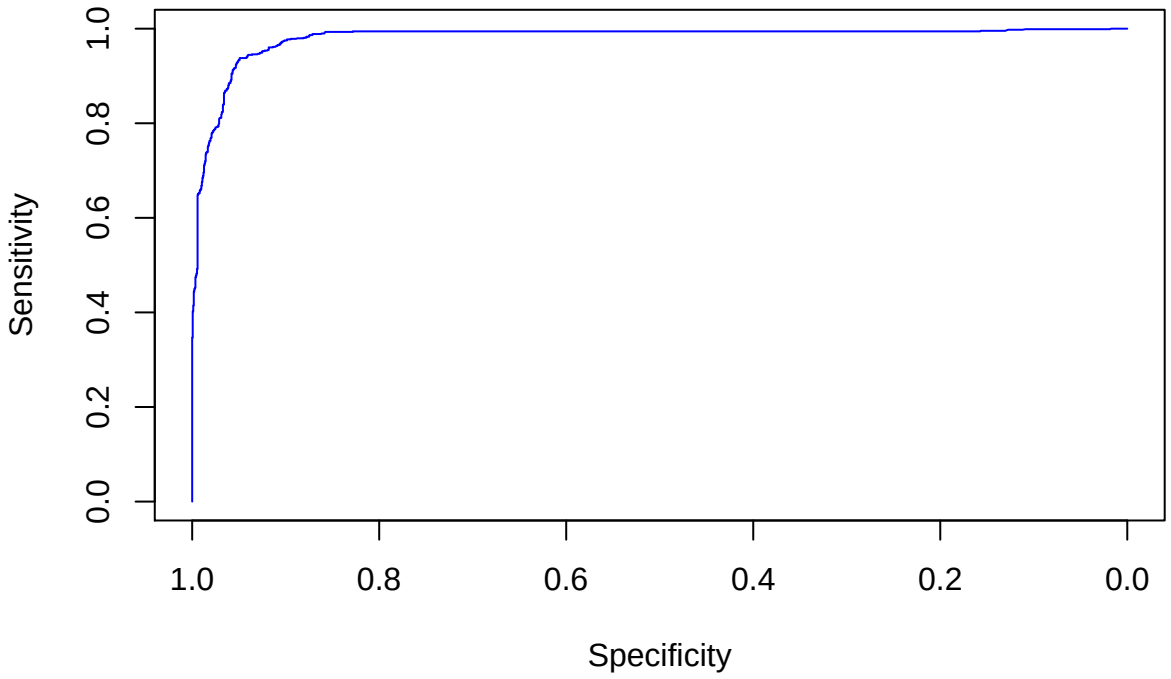
```
## Support Vector Machines with Linear Kernel
##
## 524 samples
##   3 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 471, 471, 473, 471, 473, 472, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9323662  0.8453577
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
roc(predictor = svm$pred$malignant, response = svm$pred$obs)$auc
```

```
## Area under the curve: 0.9808
```

Visualize ROC curve

```
plot(x = roc(predictor = svm$pred$malignant, response = svm$pred$obs)$specificities, y = roc(predictor =
```



Predict using test set

```
svm_test <- predict(svm, newdata = test_classification)
confusionMatrix(svm_test, reference = test_classification$Class)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       104         4
##   malignant      7        60
##
##                Accuracy : 0.9371
##                  95% CI : (0.8903, 0.9682)
##     No Information Rate : 0.6343
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8658
##  Mcnemar's Test P-Value : 0.5465
##
##             Sensitivity : 0.9369
##             Specificity : 0.9375
##          Pos Pred Value : 0.9630
##          Neg Pred Value : 0.8955
##              Prevalence : 0.6343
##          Detection Rate : 0.5943
##    Detection Prevalence : 0.6171
##       Balanced Accuracy : 0.9372
##
##        'Positive' Class : benign
##
```

Train radial SVM

```
#set.seed(30495)
#ctrl <- trainControl(method = "repeatedcv", repeats = 5,classProbs = T, savePredictions = T)
#svm_rad <- train()
```

```
#roc(predictor = svm$pred$malignant, response = svm$pred$obs)$auc
```

Visualize ROC curve

```
#plot(x = roc(predictor = svm$pred$malignant, response = svm$pred$obs)$specificities, y = roc(predictor
```

Predict using test set

```
#svm_test <- predict(svm, newdata = test_classification)
#confusionMatrix(svm_test, reference = test_classification$Class)
```

Homework

Find a dataset of your choosing. Use two different tree based methods, and SVM with two different kernels. Compare and contrast.