Linear classification and trees

Jieqi Tu (jt3098)

5/13/2019

#Load and tidy data

```r
#read data
rawdata <- read.csv("kc_house_data.csv", header = TRUE)

#inspect the structure of data
str(rawdata)
```

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : Factor w/ 372 levels "20140502T000000",..: 165 221 291 221 284 11 57 252 340 306 .
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

```r
#clean the rawdata; create a tidied dataset for analysis and modelling.
#subset data: only those with view >0 and basement >0.
housing =
  rawdata %>%
  select(-id, -date, -zipcode, -lat, -long) %>%
  filter(view > 0, bedrooms <30) %>%
  mutate(basement = ifelse(sqft_basement == 0, 0, 1),
         renovated = ifelse(yr_renovated == 0, 0, 1)) %>%
  filter(basement > 0) %>%
  select(-sqft_basement, -yr_renovated, -view, - sqft_living, -sqft_lot, -basement)

# dichotimize response variable
median(housing$price) #805000
```

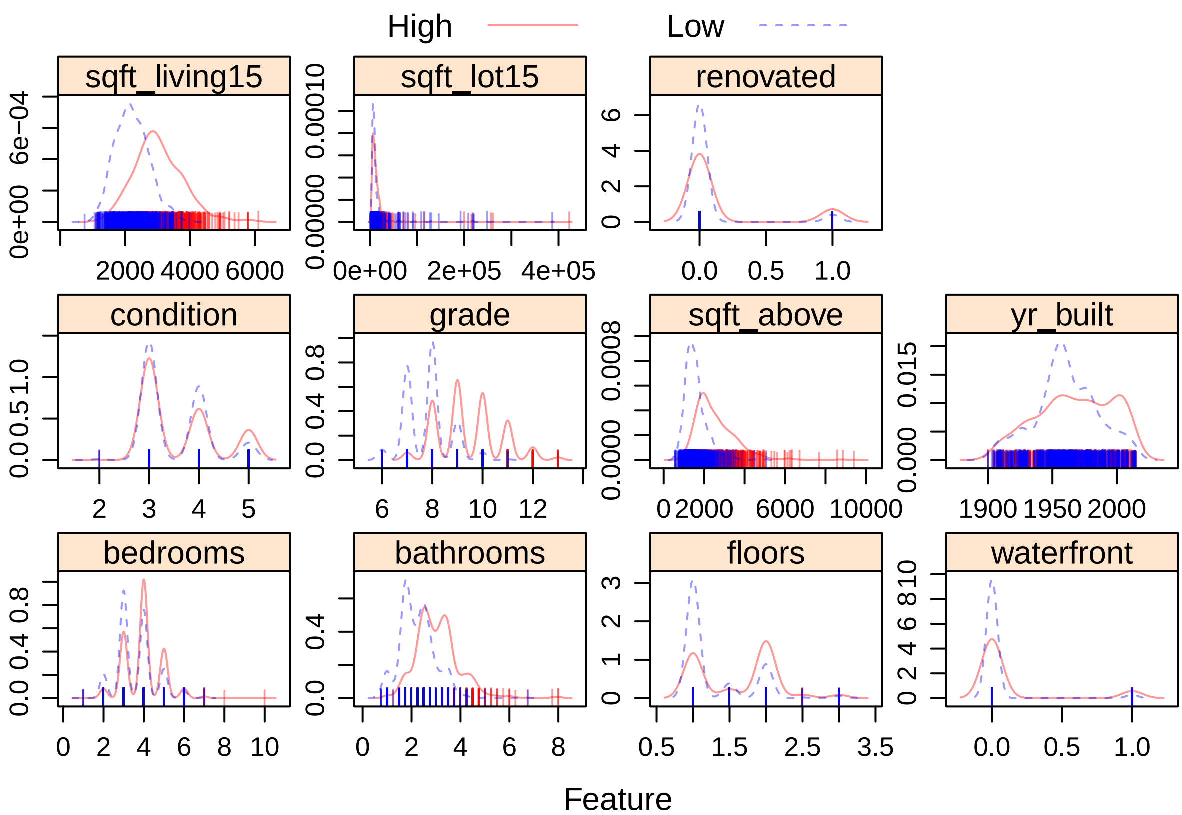
```
## [1] 805000
```

```r
housing <- housing %>% mutate(price.new = ifelse(price>805000, "High", "Low"))
housing$price.new <- factor(housing$price.new, c("High", "Low"))
```

```r
# create training data and testing data.
rowTrain <- createDataPartition(y = housing$price,
                                p=0.8, list = FALSE)
```

Data visuailization

```r
transparentTheme(trans = 0.4)
featurePlot(x = housing[,2:12],
            y = housing$price.new,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



From the feature plot we could see that, the two classes of price are distributed differently in features.

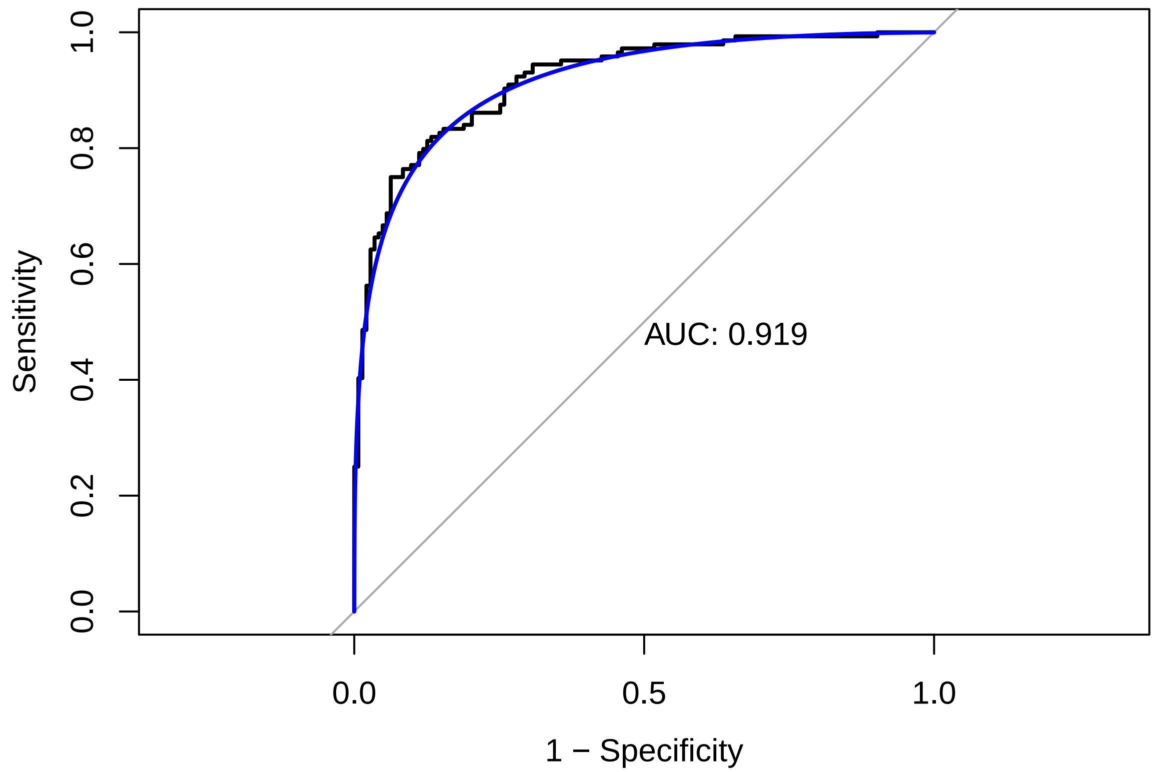
Logistic Regression

```r
# Use caret package
ctrl = trainControl(method = "cv",
                    summaryFunction = twoClassSummary,
                    classProbs = T)
set.seed(1)
model.glm = train(x = housing[rowTrain, 2:12],
```

```r
                   y = housing$price.new[rowTrain],
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
# test performance
pred.glm = predict(model.glm, newdata = housing[-rowTrain,2:13], response = "prob")
pred.glm.prob = predict(model.glm, newdata = housing[-rowTrain,2:13], type = "prob")
confusionMatrix(data = as.factor(pred.glm), reference = housing$price.new[-rowTrain])
```

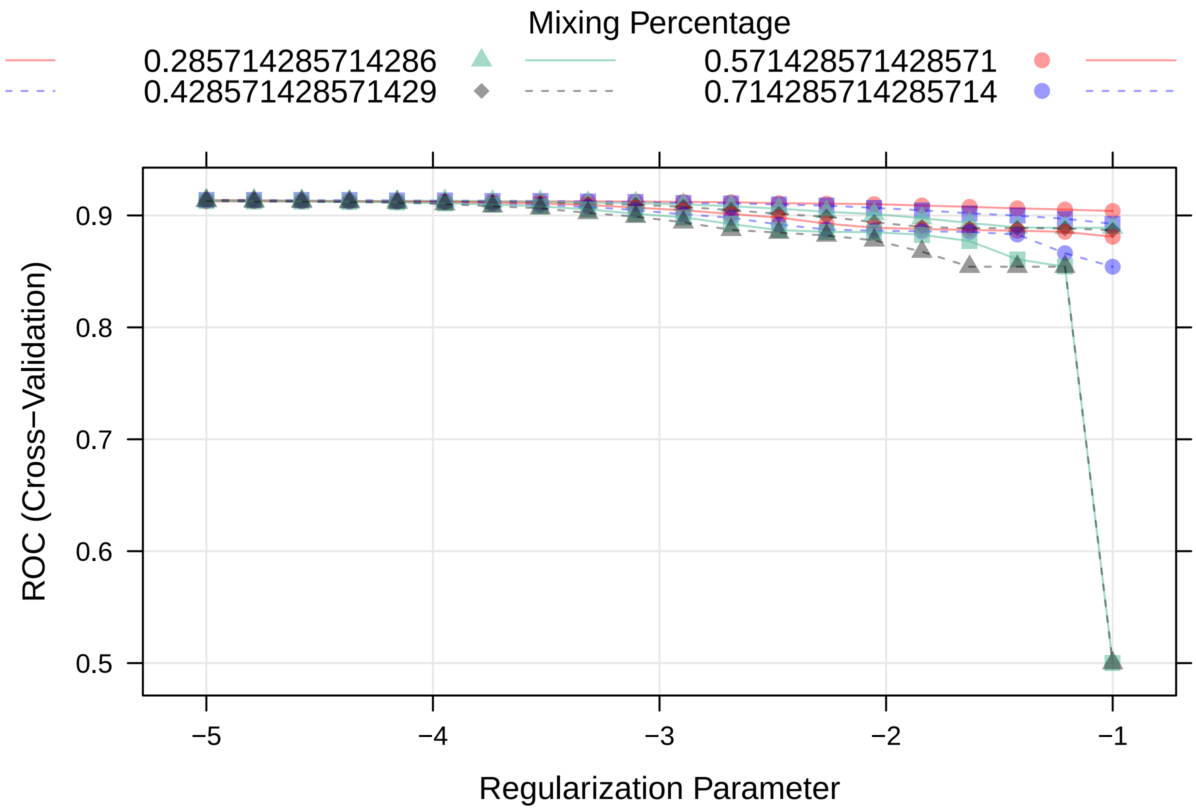
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  116  24
##       Low    27 120
##
##                Accuracy : 0.8223
##                  95% CI : (0.7731, 0.8647)
##     No Information Rate : 0.5017
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6446
##  Mcnemar's Test P-Value : 0.7794
##
##             Sensitivity : 0.8112
##             Specificity : 0.8333
##          Pos Pred Value : 0.8286
##          Neg Pred Value : 0.8163
##              Prevalence : 0.4983
##          Detection Rate : 0.4042
##    Detection Prevalence : 0.4878
##       Balanced Accuracy : 0.8223
##
##        'Positive' Class : High
##
```

```r
# plot the ROC curve
roc.glm = roc(housing$price.new[-rowTrain], pred.glm.prob$High)
plot(roc.glm, legacy.axes = T, print.auc = T)
plot(smooth(roc.glm), col = 4, add = T)
```



The AUC is 0.943. The accuracy is 0.8676. Sensitivity is 0.8042, Specificity is 0.9306.

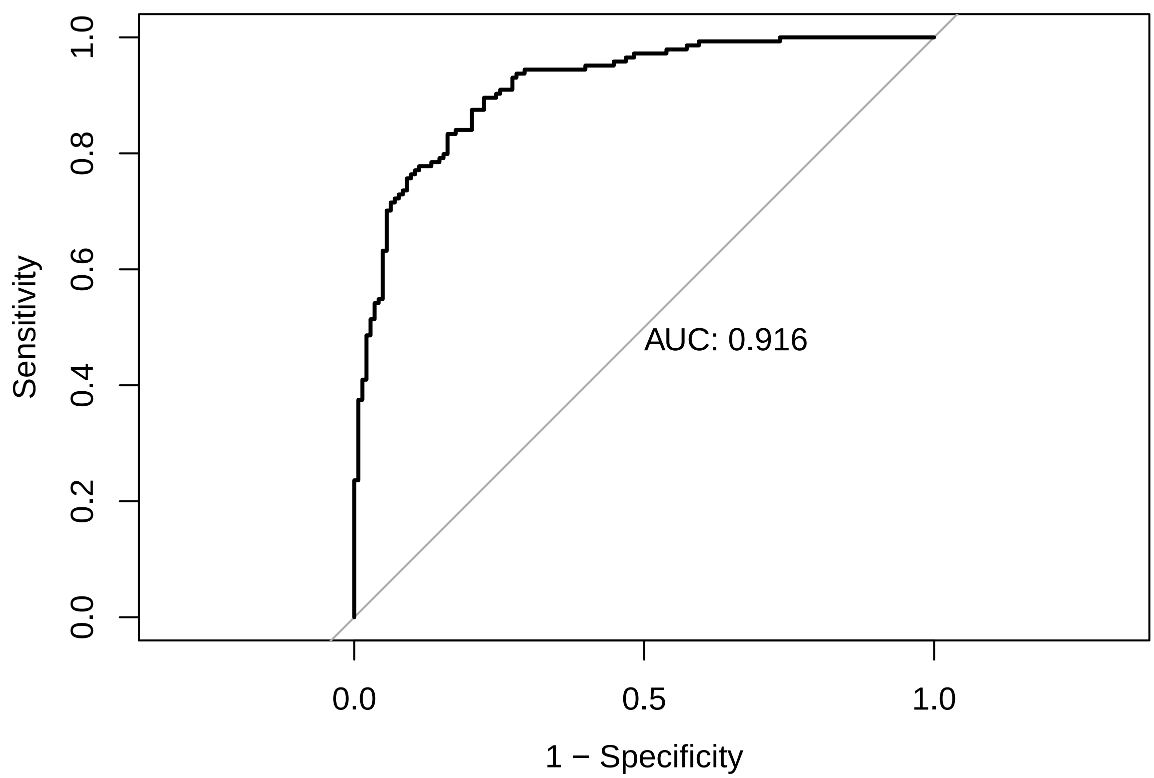
```
glmnGrid = expand.grid(.alpha = seq(0, 1, length = 8),
                       .lambda = exp(seq(-5, -1, length = 20)))
set.seed(1)
model.glmn = train(x = housing[rowTrain, 2:12],
                   y = housing$price.new[rowTrain],
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   metric = "ROC",
                   trControl = ctrl)

plot(model.glmn, xTrans = function(x) log(x))
```




```
pred.glmn = predict(model.glmn, newdata = housing[-rowTrain, 2:13], type = "prob")
```

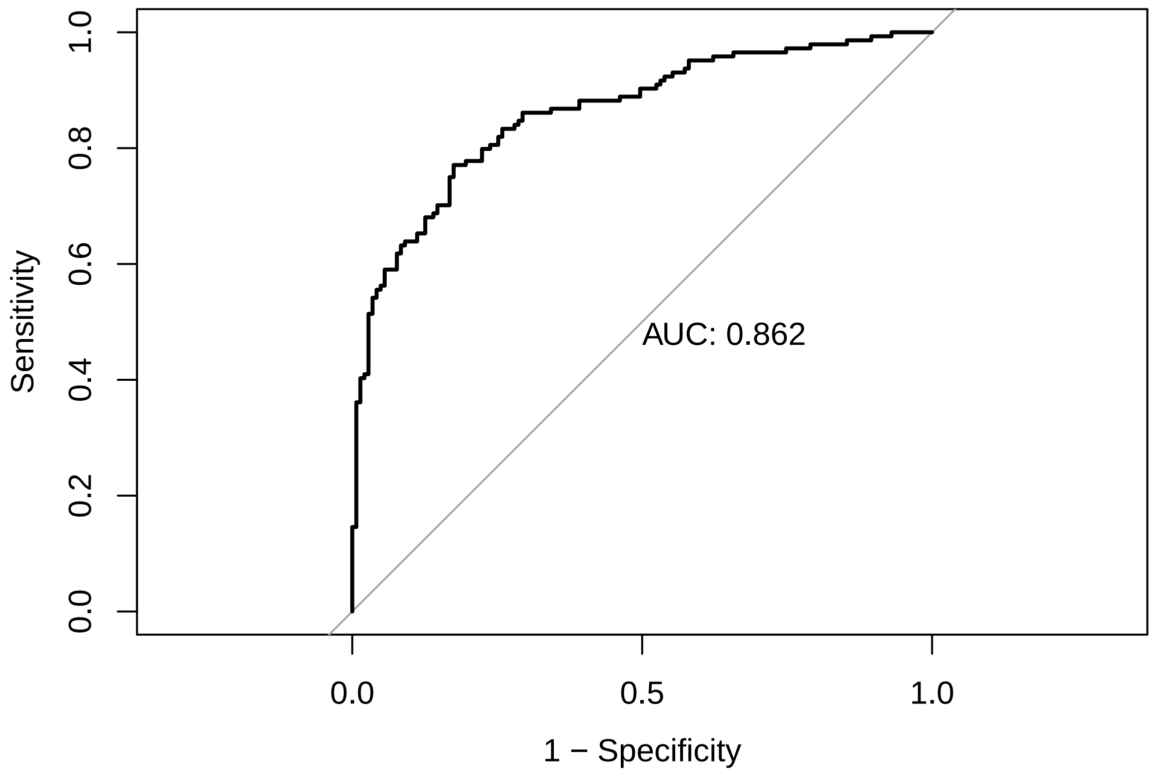
Discriminant Analysis

```
set.seed(1)
model.lda = train(x = housing[rowTrain, 2:12],
                  y = housing$price.new[rowTrain],
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)
pred.lda = predict(model.lda, newdata = housing[-rowTrain,2:13], type = "prob")
roc.lda = roc(housing$price.new[-rowTrain], pred.lda$High, levels = c("High", "Low"))
plot(roc.lda, legacy.axes = T, print.auc = T)
```



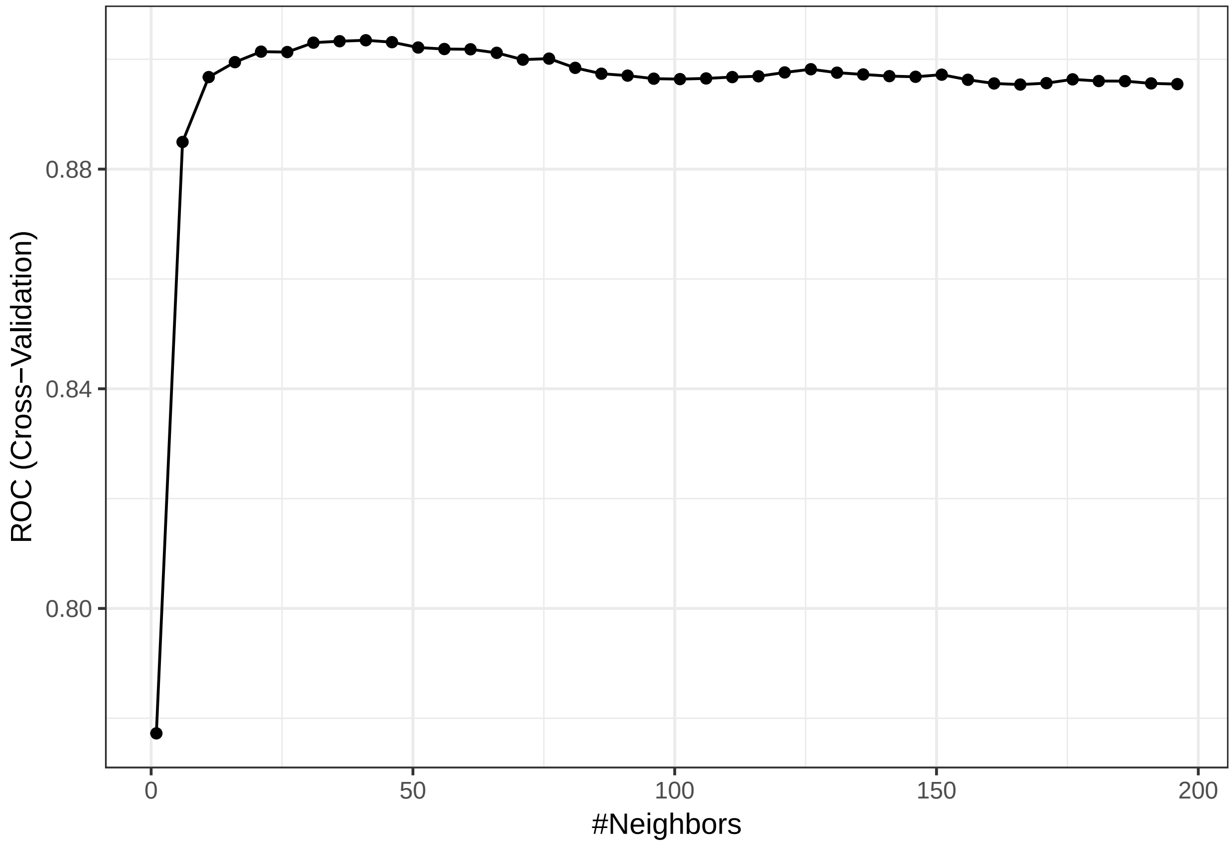
The AUC for LDA is 0.933.

```
set.seed(1)
model.qda = train(x = housing[rowTrain, 2:12],
                  y = housing$price.new[rowTrain],
                  method = "qda",
                  metric = "ROC",
                  trControl = ctrl)

pred.qda = predict(model.qda, newdata = housing[-rowTrain, 2:13], type = "prob")
roc.qda = roc(housing$price.new[-rowTrain], pred.qda$High, levels = c("High", "Low"))
plot(roc.qda, legacy.axes = T, print.auc = T)
```



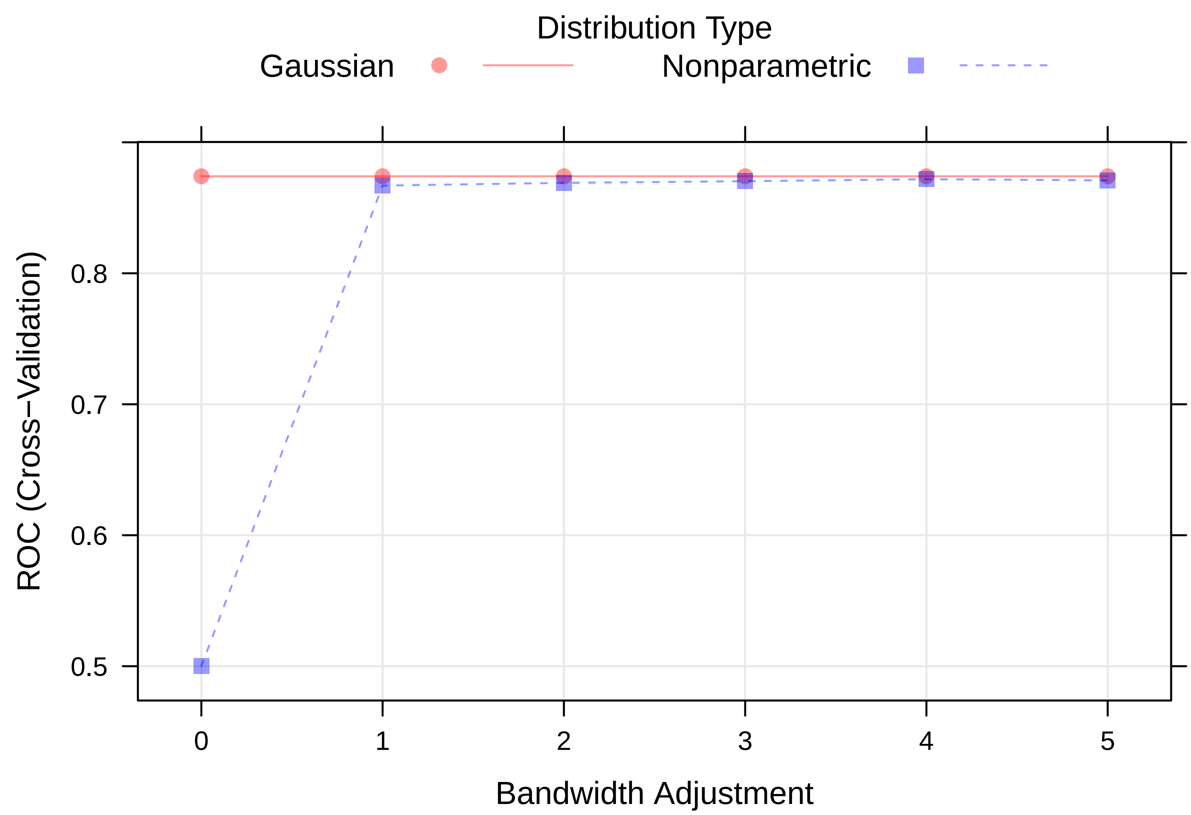
The AUC for QDA is 0.889.

```
set.seed(1)
model.knn = train(x = housing[rowTrain, 2:12],
                  y = housing$price.new[rowTrain],
                  method = "knn",
                  preProcess = c("center", "scale"),
                  metric = "ROC",
                  tuneGrid = data.frame(k = seq(1, 200, by = 5)),
                  trControl = ctrl)
ggplot(model.knn) + theme_bw()
```




```r
pred.knn = predict(model.knn, newdata = housing[-rowTrain, 2:13], type = "prob")
```

```r
set.seed(1)
nbGrid = expand.grid(usekernel = c(FALSE, TRUE),
                     fL = 0,
                     adjust = seq(0, 5, by = 1))
housing = na.omit(housing)

model.nb = train(x = housing[rowTrain, 2:12],
                 y = housing$price.new[rowTrain],
                 method = "nb",
                 metric = "ROC",
                 tuneGrid = nbGrid,
                 trControl = ctrl)

pred.nb = predict(model.nb, newdata = housing[-rowTrain, 2:13], type = "prob")
plot(model.nb)
```



Model Comparism

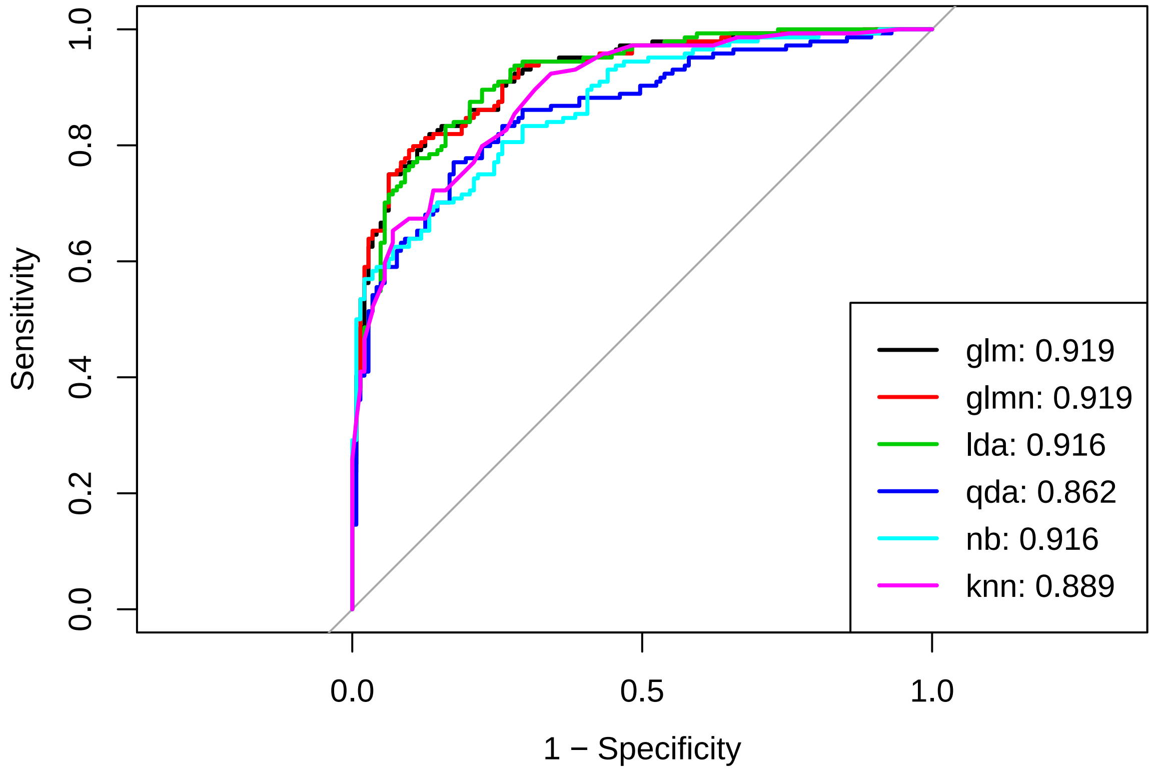
```
res = resamples(list(GLM = model.glm, LDA = model.lda, QDA = model.qda, NB = model.nb, KNN = model.knn)

summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, LDA, QDA, NB, KNN
## Number of resamples: 10
##
## ROC
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM 0.8573127 0.9089521 0.9198215 0.9142585 0.9309589 0.9600726    0
## LDA 0.8550555 0.8987283 0.9103433 0.9089397 0.9334439 0.9522081    0
## QDA 0.7728894 0.8615840 0.8815172 0.8634833 0.8907290 0.9183303    0
## NB  0.8096017 0.8601367 0.8821448 0.8740595 0.8910120 0.9479734    0
## KNN 0.8496493 0.8914454 0.9045779 0.9034661 0.9159496 0.9581065    0
##
## Sens
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM 0.7241379 0.8103448 0.8348457 0.8230792 0.8421053 0.8965517    0
## LDA 0.6896552 0.7456897 0.7652753 0.7692982 0.8070175 0.8448276    0
## QDA 0.6034483 0.6753630 0.7041742 0.7032365 0.7468240 0.7586207    0
```

```
## NB  0.6896552 0.7112069 0.7566546 0.7520266 0.7860708 0.8245614    0
## KNN 0.7413793 0.8103448 0.8245614 0.8248639 0.8384755 0.9298246    0
##
## Spec
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM 0.7627119 0.7974138 0.8534483 0.8471946 0.8885155 0.9482759    0
## LDA 0.7966102 0.8232759 0.8793103 0.8729690 0.9056838 0.9482759    0
## QDA 0.7966102 0.8448276 0.8534483 0.8574226 0.8879310 0.9137931    0
## NB  0.7586207 0.7887931 0.8362069 0.8367329 0.8620690 0.9322034    0
## KNN 0.6949153 0.8275862 0.8362069 0.8420807 0.8879310 0.9482759    0
```

GLM and LDA tend to have higher AUC values.

```r
roc.glmn = roc(housing$price.new[-rowTrain], pred.glmn[,2])
roc.nb = roc(housing$price.new[-rowTrain], pred.nb[,2])
roc.knn = roc(housing$price.new[-rowTrain], pred.knn[,2])

auc = c(roc.glm$auc[1], roc.glmn$auc[1], roc.lda$auc[1],
        roc.qda$auc[1], roc.lda$auc[1], roc.knn$auc[1])

plot(roc.glm, legacy.axes = T)
plot(roc.glmn, col = 2, add = T)
plot(roc.lda, col = 3, add = T)
plot(roc.qda, col = 4, add = T)
plot(roc.nb, col = 5, add = T)
plot(roc.knn, col = 6, add = T)
modelNames = c("glm", "glmn", "lda", "qda", "nb", "knn")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc, 3)), col = 1:6, lwd = 2)
```



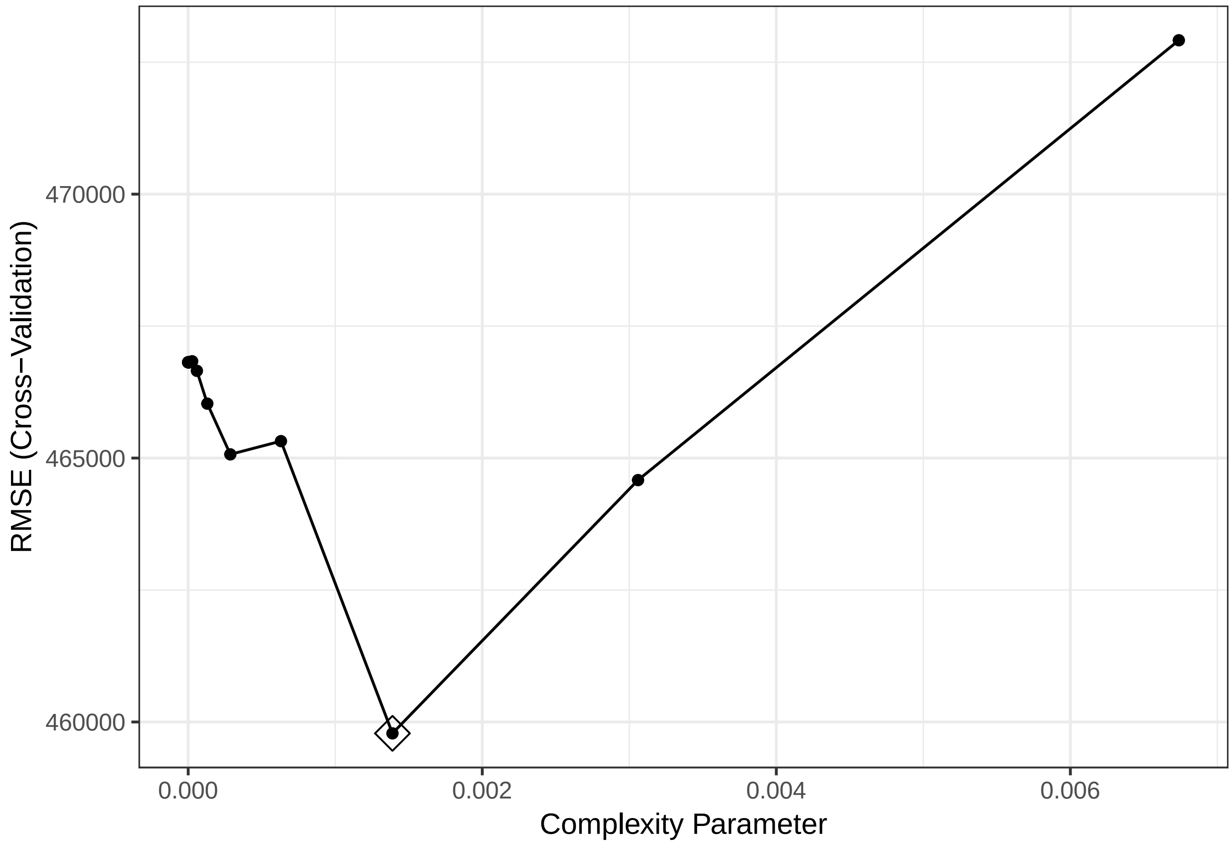
Regression Trees

```
set.seed(1)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.3
```

```
## Loading required package: rpart
```

```
housing =
  rawdata %>%
  dplyr::select(-id, -date, -zipcode, -lat, -long) %>%
  filter(view > 0, bedrooms <30) %>%
  mutate(basement = ifelse(sqft_basement == 0, 0, 1),
         renovated = ifelse(yr_renovated == 0, 0, 1)) %>%
  filter(basement > 0) %>%
  dplyr::select(-sqft_basement, -yr_renovated, -view, - sqft_living, -sqft_lot, -basement)
housing = na.omit(housing)

ctrl1 = trainControl(method = "cv", number = 10)
rpart.fit = train(price~., housing,
                  method = "rpart",
                  tuneGrid = data.frame(cp = exp(seq(-20, -5, length = 20))),
                  trControl = ctrl1)

ggplot(rpart.fit, highlight = T) + theme_bw()
```

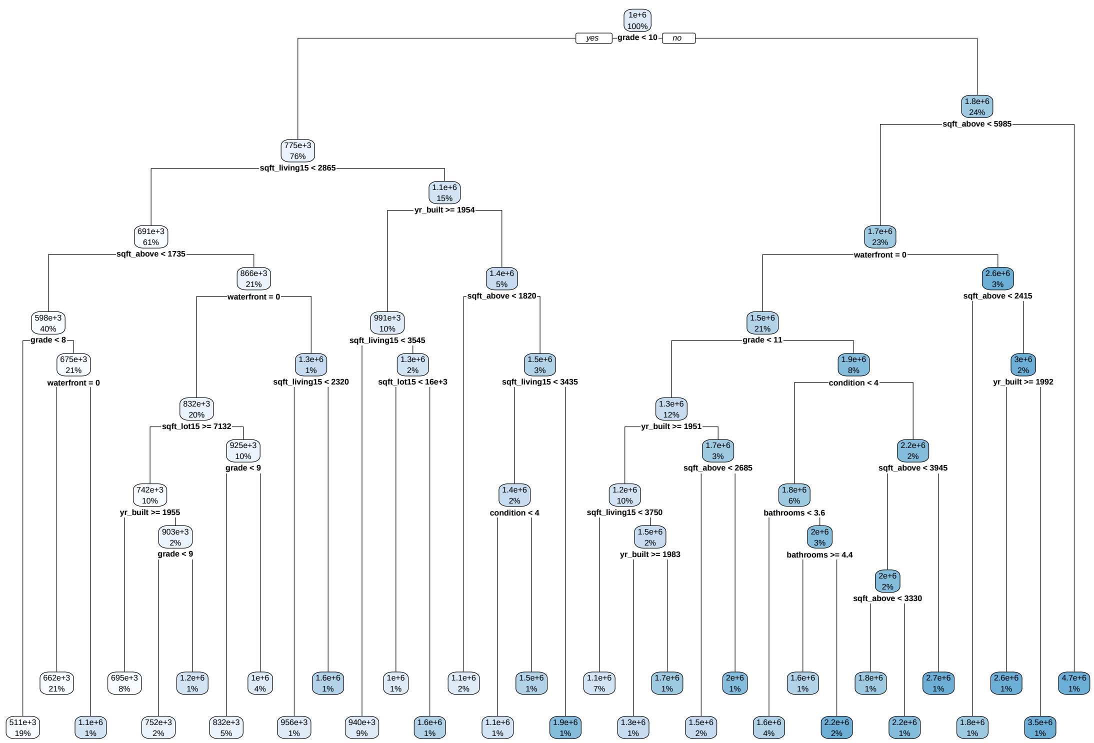



```
rpart.fit$finalModel$cptable
```

```
##              CP nsplit rel error
## 1  0.347972599      0 1.0000000
## 2  0.120364953      1 0.6520274
## 3  0.054320762      2 0.5316624
## 4  0.043453850      3 0.4773417
## 5  0.027721368      4 0.4338878
## 6  0.019599772      5 0.4061665
## 7  0.015964791      6 0.3865667
## 8  0.010705965      7 0.3706019
## 9  0.009684767      8 0.3598959
## 10 0.006843633      9 0.3502112
## 11 0.006279519     10 0.3433675
## 12 0.006069700     11 0.3370880
## 13 0.005251318     12 0.3310183
## 14 0.005199933     13 0.3257670
## 15 0.005047791     14 0.3205671
## 16 0.004979332     15 0.3155193
## 17 0.004732901     16 0.3105399
## 18 0.004686030     17 0.3058070
## 19 0.004466411     18 0.3011210
## 20 0.003792708     19 0.2966546
## 21 0.003259688     20 0.2928619
## 22 0.002957927     21 0.2896022
## 23 0.002818640     22 0.2866443
```

```
## 24 0.002045235     23 0.2838256
## 25 0.001956454     24 0.2817804
## 26 0.001935589     25 0.2798240
## 27 0.001911743     27 0.2759528
## 28 0.001887865     28 0.2740410
## 29 0.001575524     29 0.2721532
## 30 0.001501664     30 0.2705776
## 31 0.001389311     31 0.2690760
```

```r
rpart.plot(rpart.fit$finalModel)
```

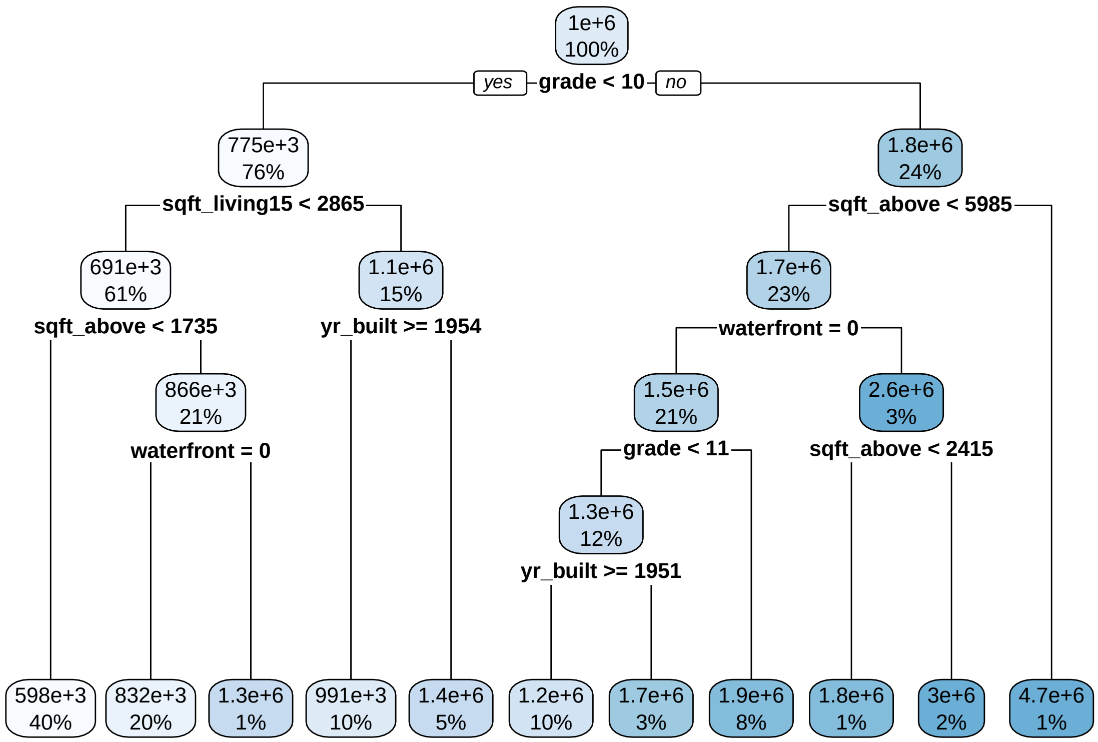



```r
# 1se rule
rpart.fit.1se = train(price~., housing,
                      method = "rpart",
                      tuneGrid = data.frame(cp = exp(seq(-20, -5, length = 20))),
                      trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"))

rpart.fit.1se$finalModel$cptable
```

```
##              CP nsplit rel error
## 1  0.347972599      0 1.0000000
## 2  0.120364953      1 0.6520274
## 3  0.054320762      2 0.5316624
## 4  0.043453850      3 0.4773417
## 5  0.027721368      4 0.4338878
## 6  0.019599772      5 0.4061665
## 7  0.015964791      6 0.3865667
```

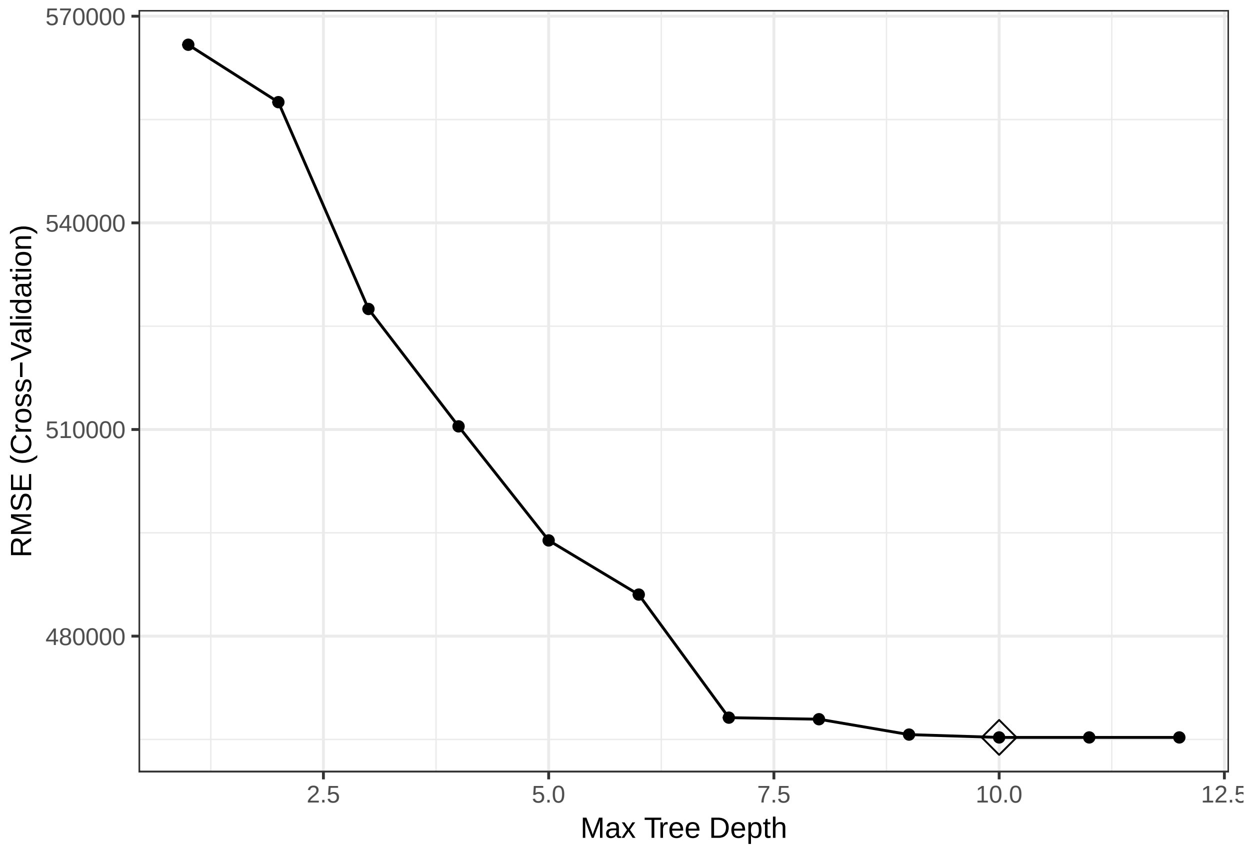
```
## 8   0.010705965      7 0.3706019
## 9   0.009684767      8 0.3598959
## 10  0.006843633      9 0.3502112
## 11  0.006737947     10 0.3433675
```

```r
rpart.plot(rpart.fit.1se$finalModel)
```

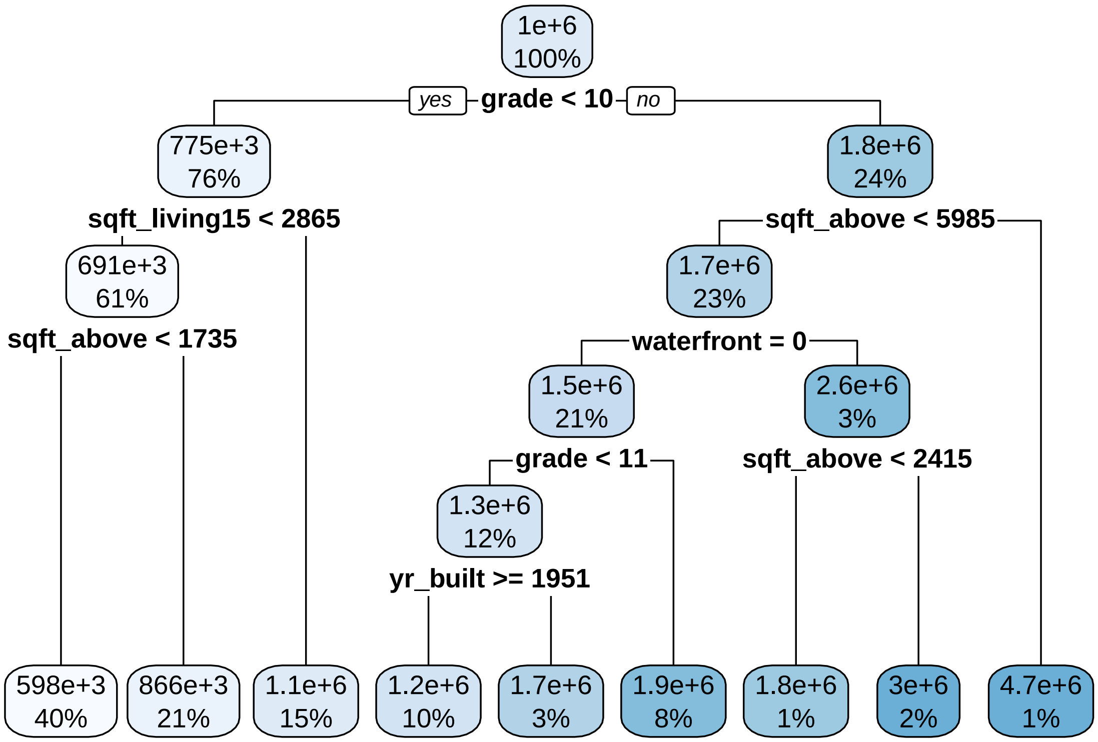


The 1se rule provided a much more simple model.

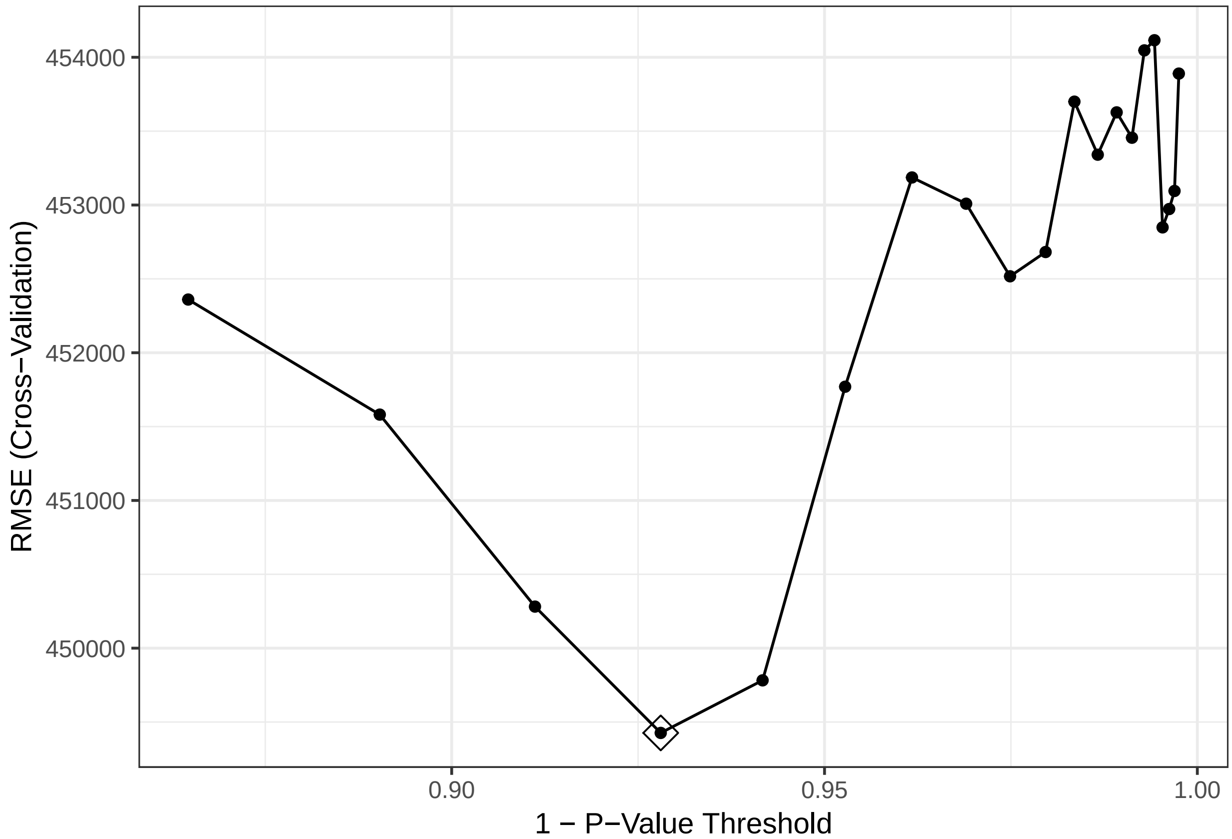
```r
# tune over maximum depth, method = "rpart2"
set.seed(1)
rpart2.fit = train(price~., housing,
                   method = "rpart2",
                   tuneGrid = data.frame(maxdepth = 1:12),
                   trControl = ctrl1)
ggplot(rpart2.fit, highlight = T) + theme_bw()
```



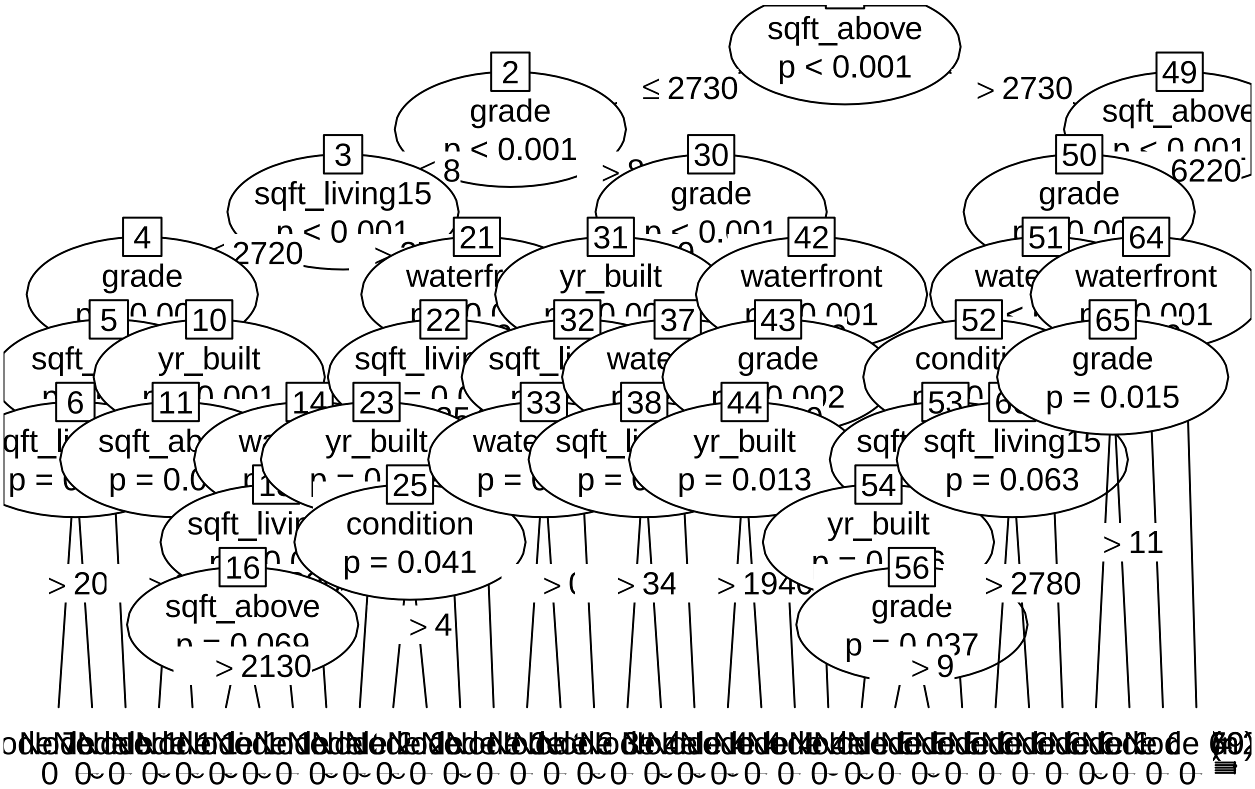

```r
rpart.plot(rpart2.fit$finalModel)
```




```r
# fit a conditional inference tree
set.seed(1)
ctree.fit = train(price~., housing,
                  method = "ctree",
                  tuneGrid = data.frame(mincriterion = 1 - exp(seq(-6, -2, length = 20))),
                  trControl = ctrl1)
ggplot(ctree.fit, highlight = T) + theme_bw()
```



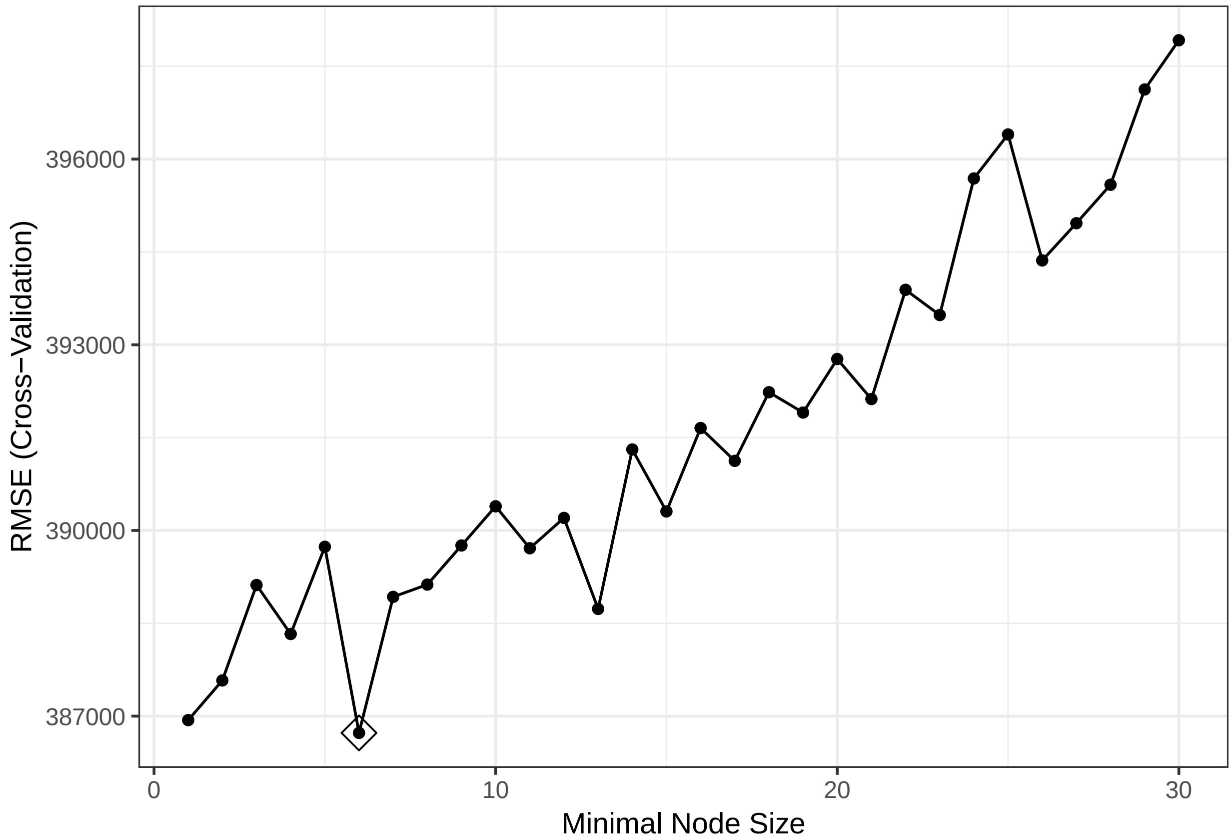

```r
plot(ctree.fit$finalModel)
```



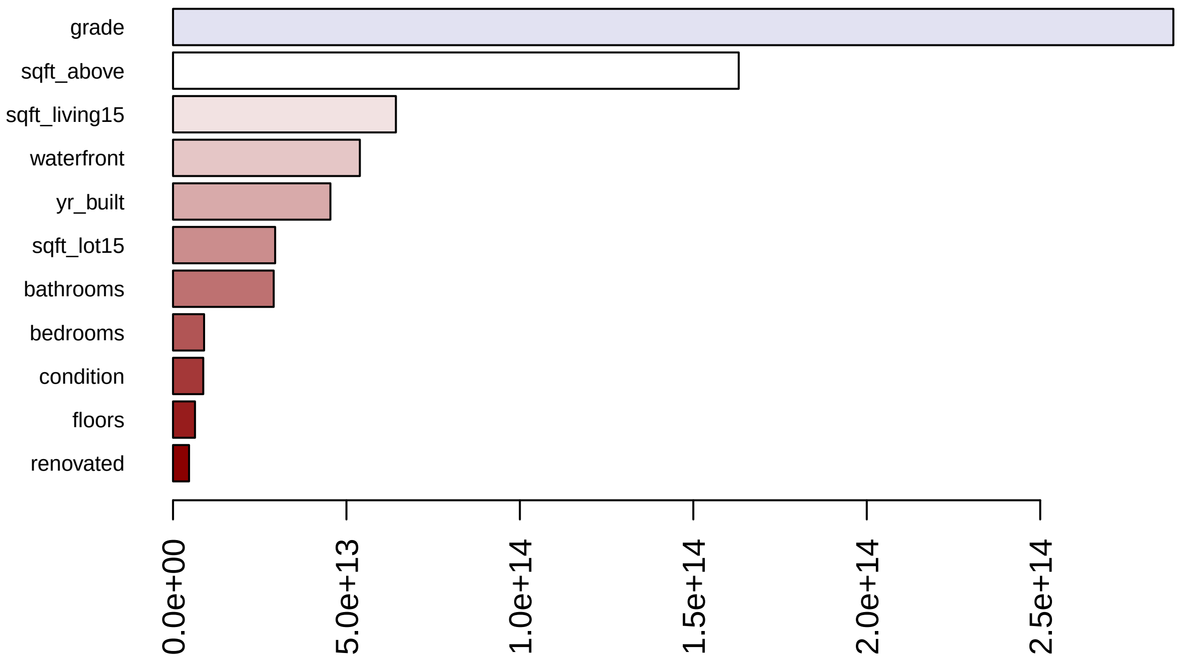
Ensemble Methods

Bagging and Random forests

```
# bagging
bagging.grid = expand.grid(mtry = 11,
                           splitrule = "variance",
                           min.node.size = 1:30)
set.seed(1)
bagging.fit = train(price~., housing,
                    method = "ranger",
                    tuneGrid = bagging.grid,
                    trControl = ctrl1,
                    importance = "impurity")

ggplot(bagging.fit, highlight = T) + theme_bw()
```

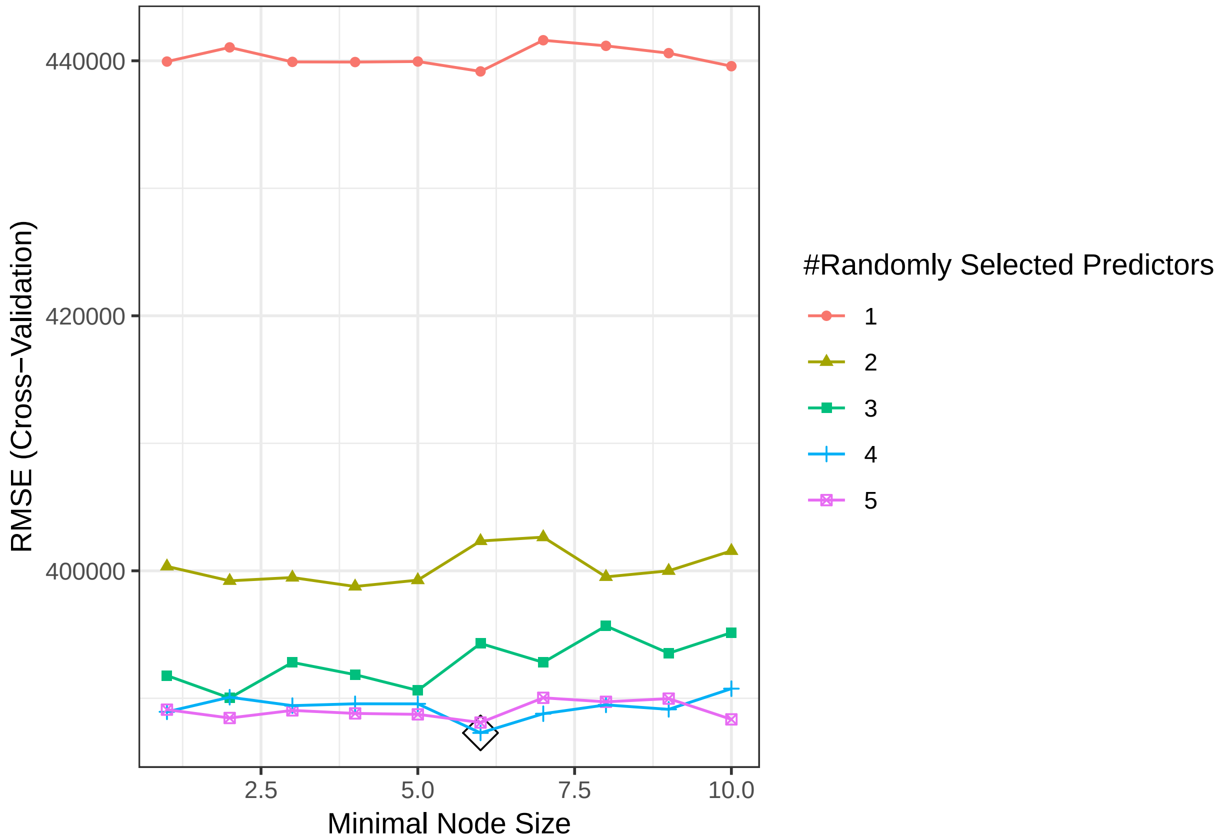



```r
# check variable importance
barplot(sort(ranger::importance(bagging.fit$finalModel), decreasing = FALSE),
        las = 2, horiz = T, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred", "white", "darkblue"))(19))
```

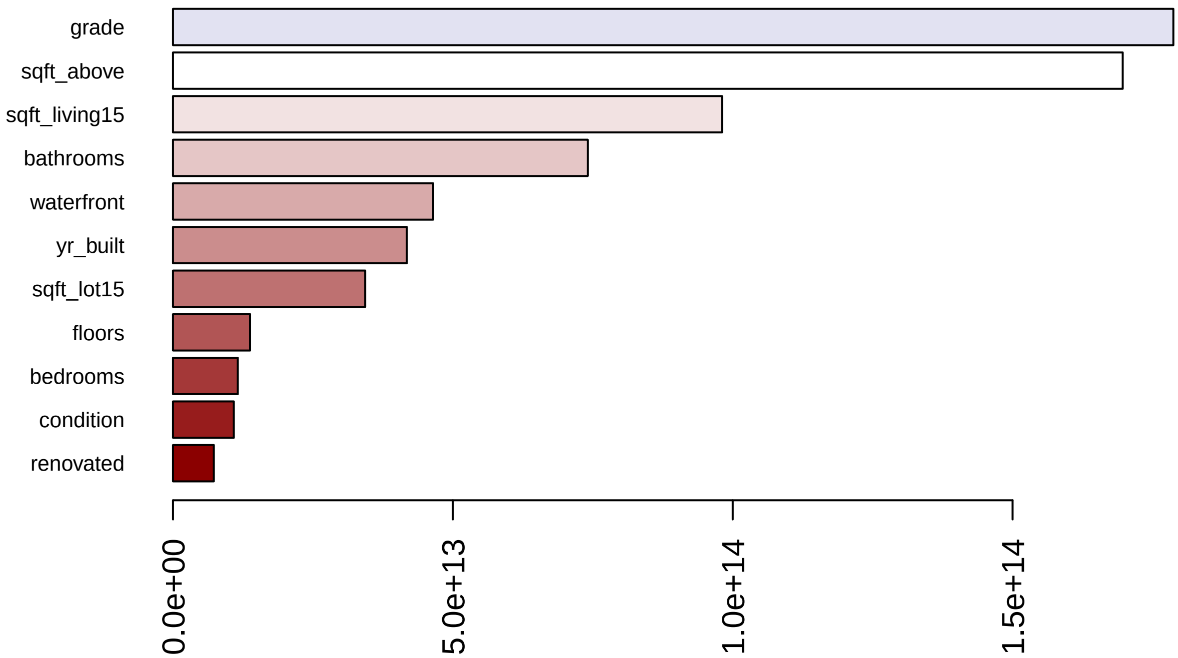


The most important variables are grade, the area above and the area of living room.

```
# ranger
rf.grid = expand.grid(mtry = 1:5, splitrule = "variance",
                      min.node.size = 1:10)
set.seed(1)
rf.fit = train(price~., housing,
               method = "ranger",
               tuneGrid = rf.grid,
               trControl = ctrl1,
               importance = "impurity")

ggplot(rf.fit, highlight = T) + theme_bw()
```

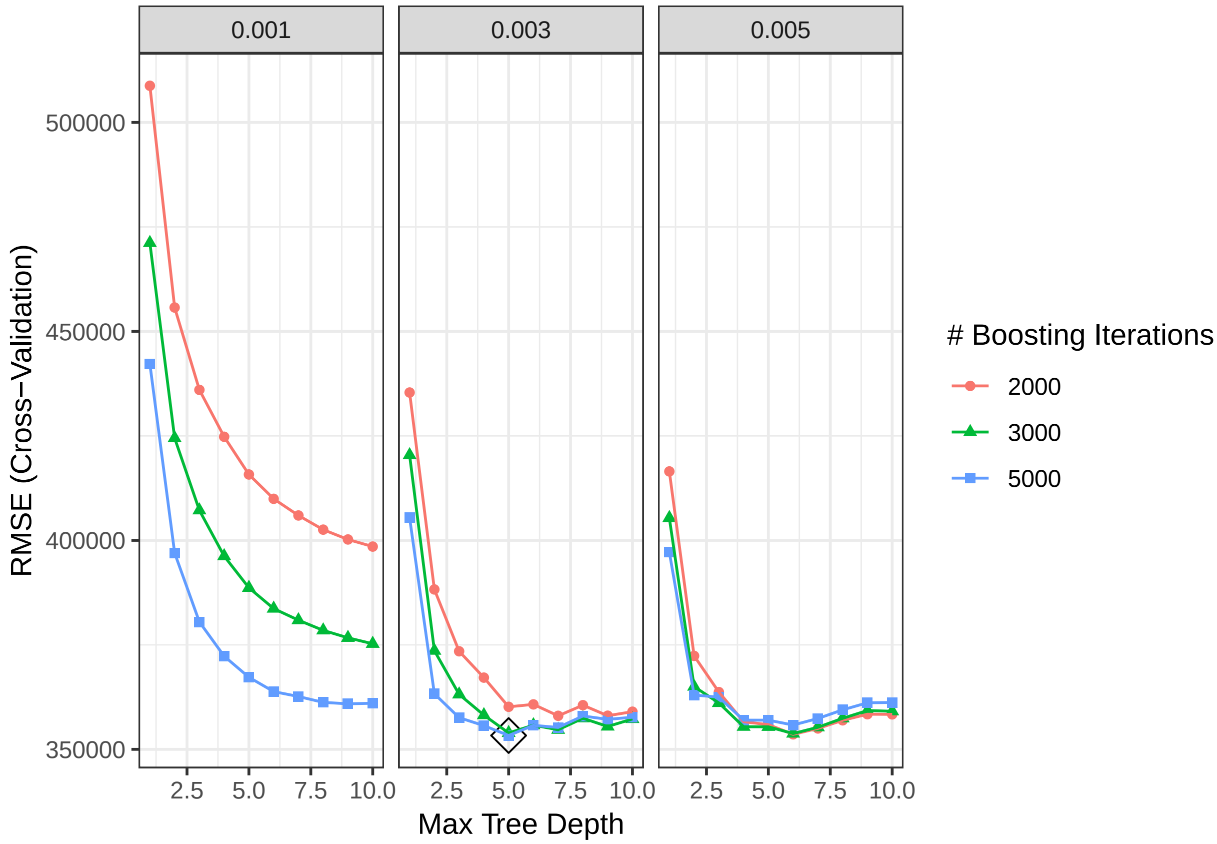



```
# check variable importance
barplot(sort(ranger::importance(rf.fit$finalModel), decreasing = FALSE),
        las = 2, horiz = T, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred", "white", "darkblue"))(19))
```

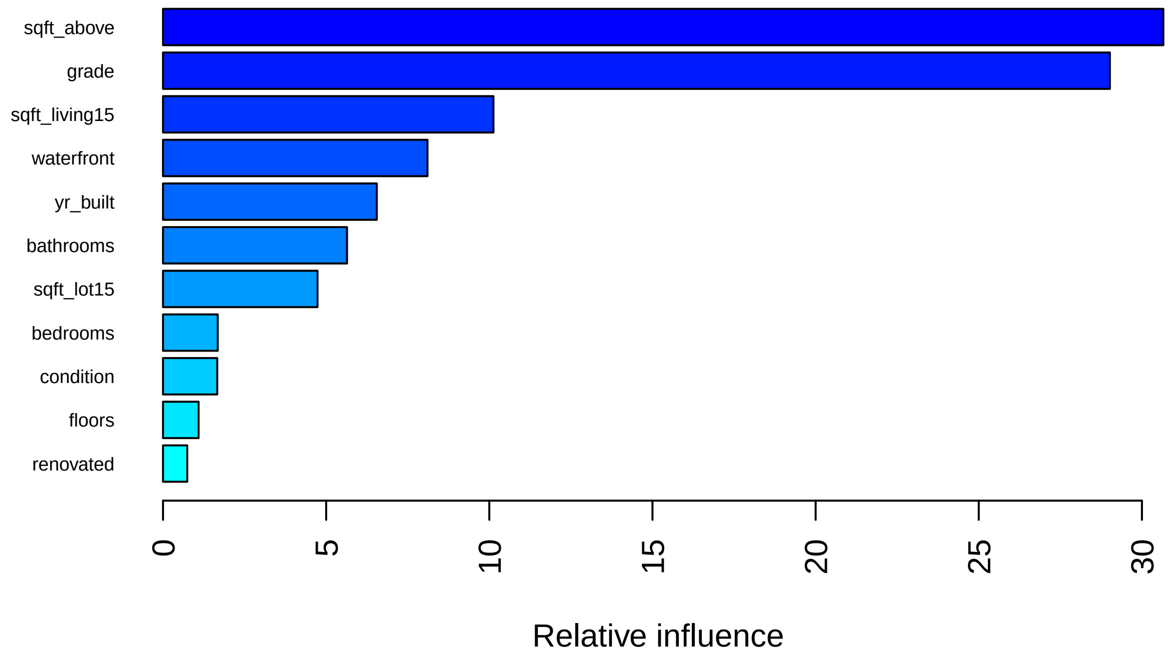


When p = 4, minimal node size = 6, we get the lowest RMSE value. The importance is the same as bagging.

```
# boosting
set.seed(1)
gbm.grid = expand.grid(n.trees = c(2000, 3000, 5000),
                       interaction.depth = 1:10,
                       shrinkage = c(0.001, 0.003, 0.005),
                       n.minobsinnode = 1)
gbm.fit = train(price~., housing,
                method = "gbm",
                tuneGrid = gbm.grid,
                trControl = ctrl1,
                verbose = FALSE)

ggplot(gbm.fit, highlight = T) + theme_bw()
```




```r
# check variable importance
summary(gbm.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```




```
##                          var    rel.inf
## sqft_above       sqft_above 30.6566942
## grade                 grade 29.0176242
## sqft_living15 sqft_living15 10.1261636
## waterfront       waterfront  8.1046630
## yr_built           yr_built  6.5510861
## bathrooms         bathrooms  5.6391518
## sqft_lot15       sqft_lot15  4.7346052
## bedrooms           bedrooms  1.6765995
## condition         condition  1.6611523
## floors               floors  1.0898455
## renovated         renovated  0.7424147
```

Model comparism

```r
# compare cross-validation performance of models
resamp = resamples(list(rf = rf.fit, bagging = bagging.fit, gbm = gbm.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: rf, bagging, gbm
## Number of resamples: 10
##
```

```
## MAE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf      222586.2 237395.5 246262.8 242687.6 248095.3 257313.1    0
## bagging 219118.1 237240.2 242723.8 243554.4 255106.5 262286.6    0
## gbm     220109.7 225375.2 226562.6 230960.4 233725.3 258841.4    0
##
## RMSE
##             Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## rf      343088.4 359983.9 379197.1 387286.1 419449.8 428591.3    0
## bagging 354378.5 361057.3 372789.1 386728.9 406661.4 456152.2    0
## gbm     313757.5 332149.8 349160.4 353304.8 364934.6 435422.9    0
##
## Rsquared
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf      0.5633065 0.6299104 0.7297866 0.7109605 0.7768532 0.8403974    0
## bagging 0.5410218 0.6218812 0.7148888 0.7051874 0.7799590 0.8456256    0
## gbm     0.6011863 0.6648199 0.7426753 0.7407741 0.8126325 0.8936769    0
```