SVM and Clustering

Jiayi Shen

5/11/2019

#Load and tidy data

```
#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 ...
```

```
#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
```

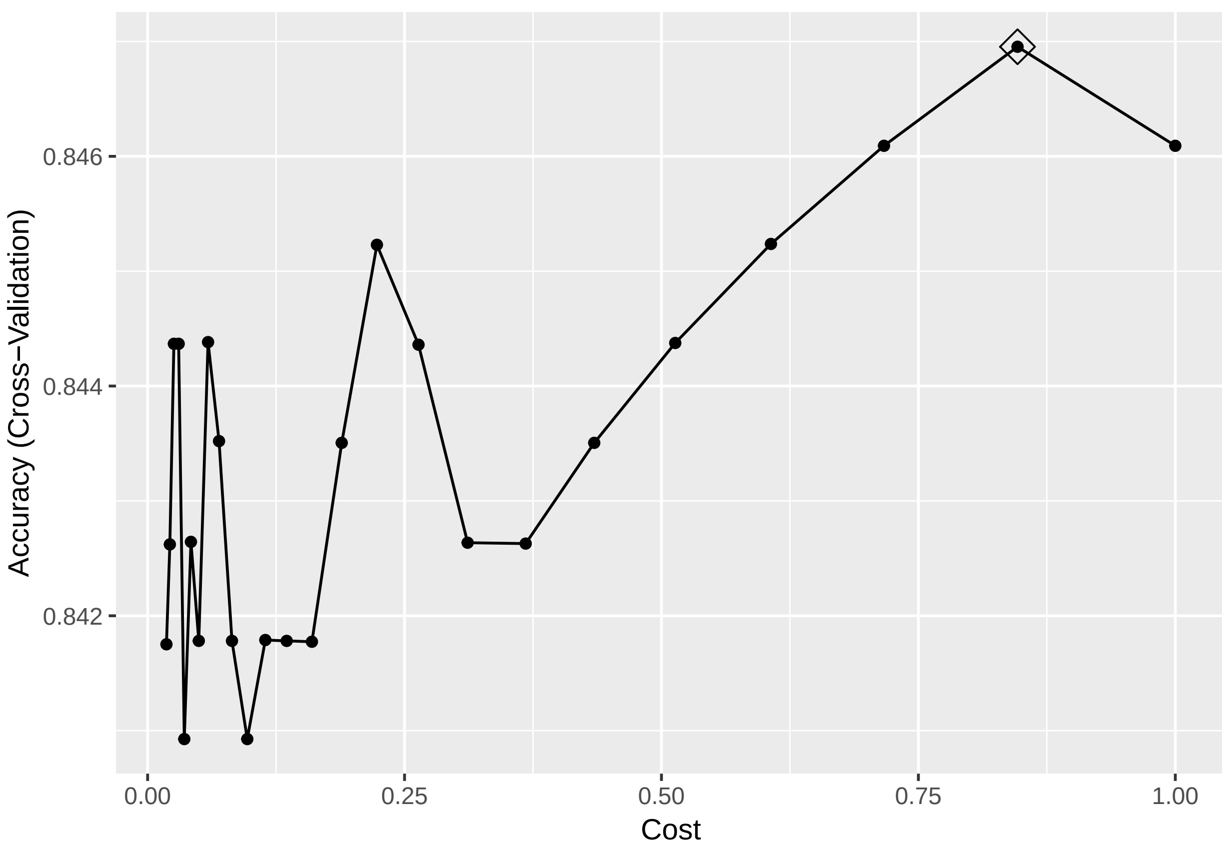
```
housing <- housing %>%
  mutate(price.new = ifelse(price>805000, "High", "Low")) %>%
```

```r
  select(-price)
housing$price.new <- factor(housing$price.new, c("High", "Low"))
# create training data and testing data.
rowTrain <- createDataPartition(y = housing$price.new,
                                p=0.8, list = FALSE)
```

SVM

Using caret
Linear Kernel

```r
ctrl <- trainControl(method = "cv")

set.seed(1)
svml.fit <- train(price.new~.,
                  data = housing[rowTrain,],
                  method = "svmLinear2",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(cost = exp(seq(-4,0,len=25))),
                  trControl = ctrl)

ggplot(svml.fit, highlight = TRUE)
```

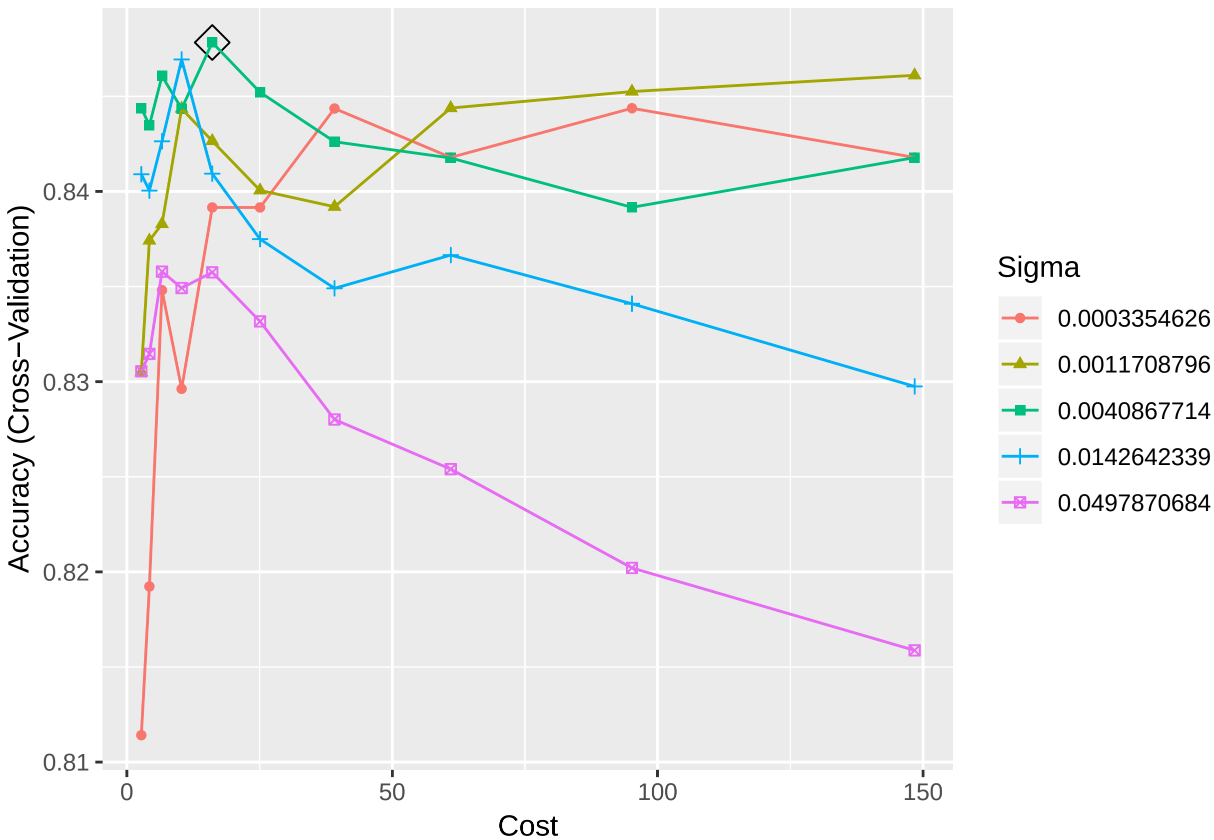



```r
svml.fit$bestTune$cost #0.311
```

```
## [1] 0.8464817
```

Radial Kernel

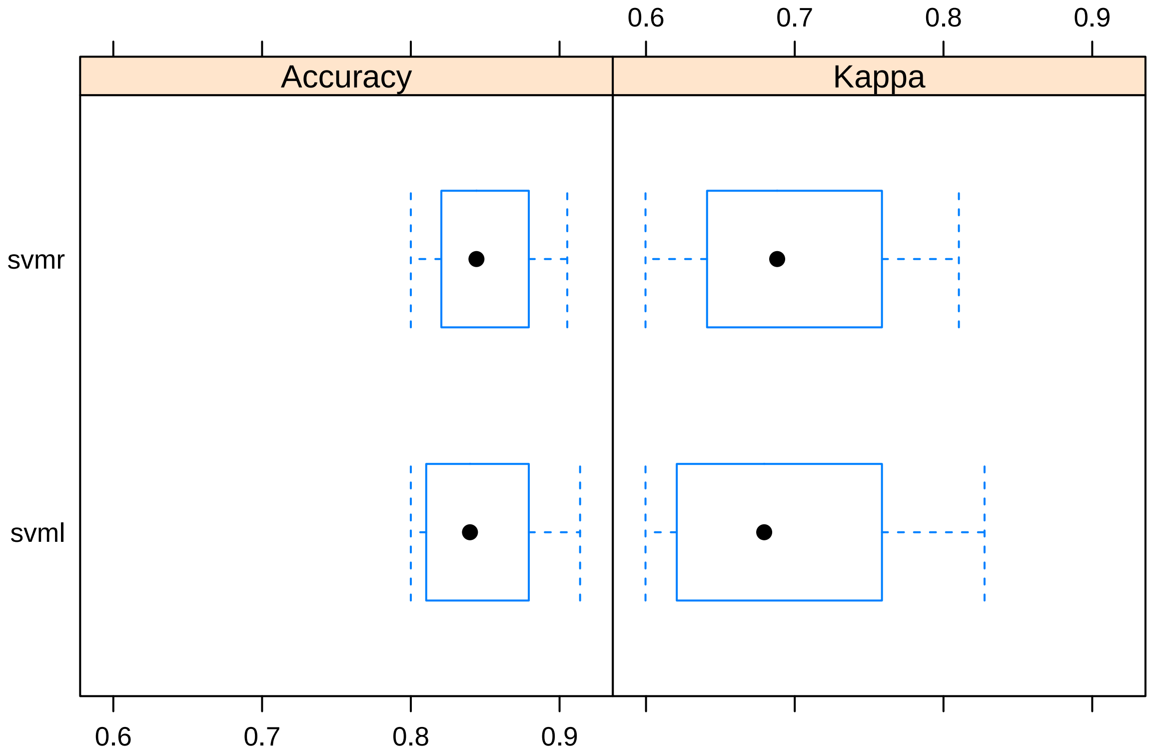
```
svmr.grid <- expand.grid(C = exp(seq(1,5,len=10)),
                         sigma = exp(seq(-8,-3,len=5)))
set.seed(1)
svmr.fit <- train(price.new~.,
                  data = housing,
                  subset = rowTrain,
                  method = "svmRadial",
                  preProcess = c("center", "scale"),
                  tuneGrid = svmr.grid,
                  trControl = ctrl)

ggplot(svmr.fit, highlight = TRUE)
```




```
svmr.fit$bestTune #sigma=0.004;  C=95
```

```
##          sigma        C
## 23 0.004086771 16.08324
```

```
resamp <- resamples(list(svmr = svmr.fit, svml = svml.fit))
bwplot(resamp)
```



Test data performance for SVM

```
pred.svml <- predict(svml.fit, newdata = housing[-rowTrain,])
pred.svmr <- predict(svmr.fit, newdata = housing[-rowTrain,])

confusionMatrix(data = pred.svml,
                reference = housing$price.new[-rowTrain]) #0.8229
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  120  22
##       Low    23 123
##
##                Accuracy : 0.8438
##                  95% CI : (0.7966, 0.8837)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6875
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8392
##             Specificity : 0.8483
##          Pos Pred Value : 0.8451
##          Neg Pred Value : 0.8425
##              Prevalence : 0.4965
```

```
##          Detection Rate : 0.4167
##    Detection Prevalence : 0.4931
##       Balanced Accuracy : 0.8437
##
##        'Positive' Class : High
##
```

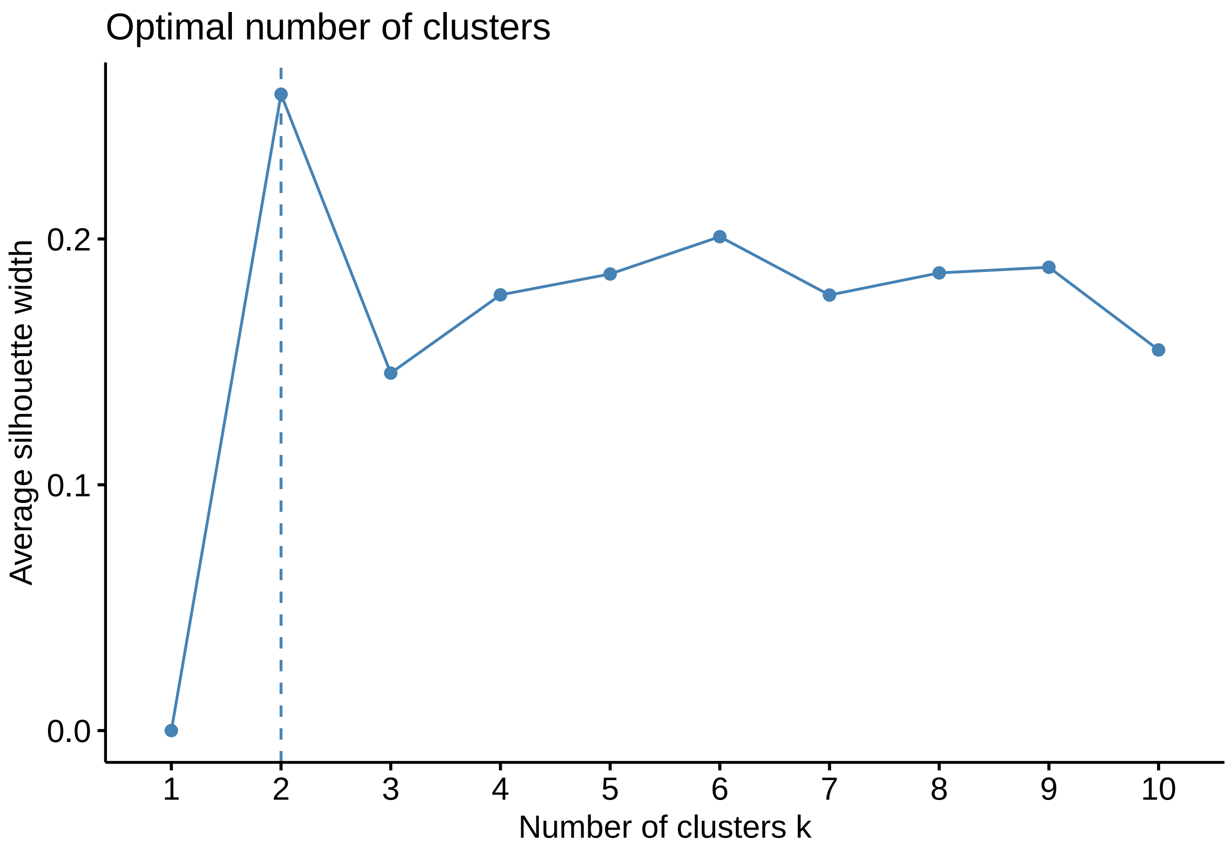
```r
confusionMatrix(data = pred.svmr,
                reference = housing$price.new[-rowTrain]) #0.8368
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  121  22
##       Low    22 123
##
##                Accuracy : 0.8472
##                  95% CI : (0.8004, 0.8867)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6944
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8462
##             Specificity : 0.8483
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8483
##              Prevalence : 0.4965
##          Detection Rate : 0.4201
##    Detection Prevalence : 0.4965
##       Balanced Accuracy : 0.8472
##
##        'Positive' Class : High
##
```

Clustering

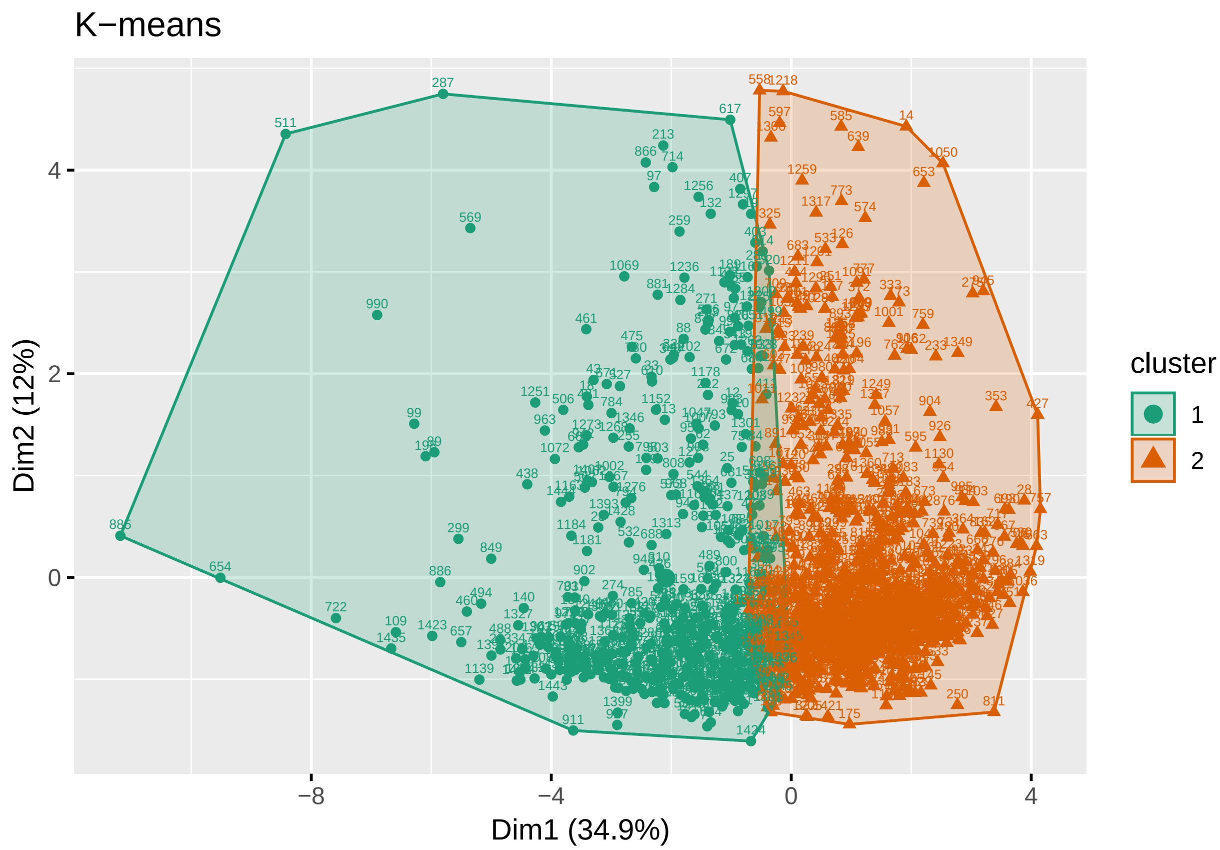
K-means

```r
housing1 <- housing[,1:11]
housing1 <- scale(housing1)

fviz_nbclust(housing1,
             FUNcluster = kmeans,
             method = "silhouette")
```



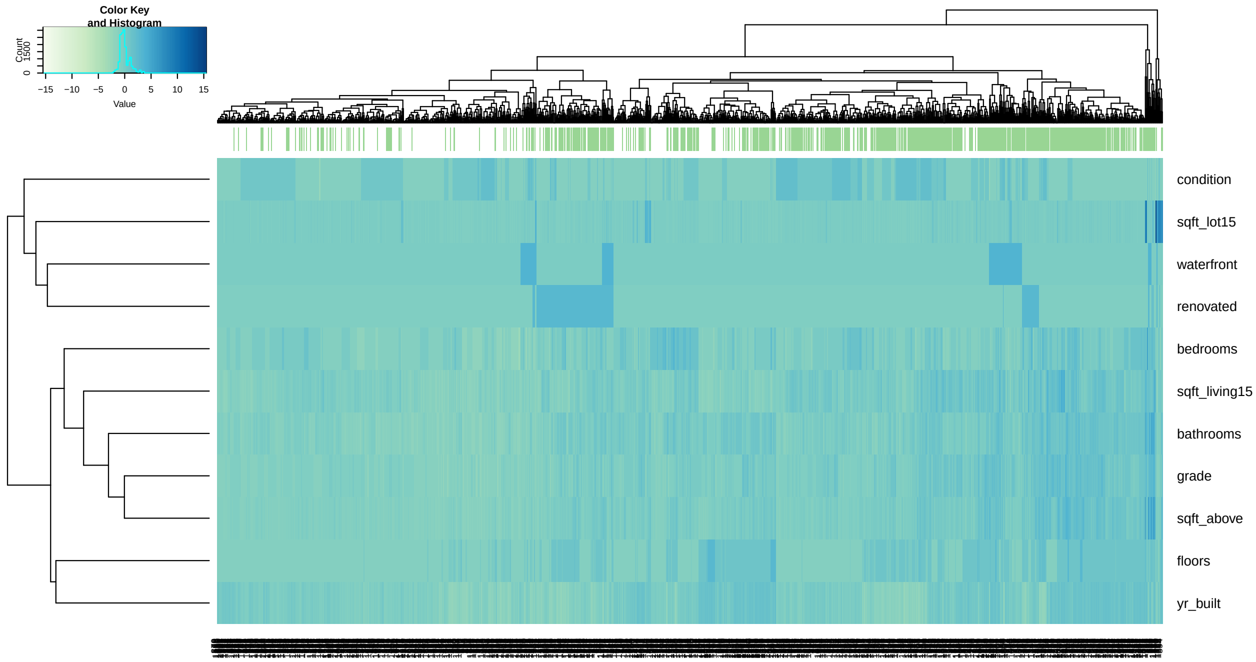

```
#optimal number of clusters = 2
```

```
set.seed(1)
km <- kmeans(housing1, centers = 2, nstart = 20)
km_vis <- fviz_cluster(list(data = housing1, cluster = km$cluster),
                       ellipse.type = "convex",
                       geom = c("point","text"),
                       labelsize = 5,
                       palette = "Dark2") + labs(title = "K-means")

km_vis
```



Hierarchical clustering

```
#display.brewer.all(n=NULL, type="all", select=NULL, exact.n=TRUE)
col1 <- colorRampPalette(brewer.pal(9, "GnBu"))(100)
col2 <- colorRampPalette(brewer.pal(3, "Spectral"))(2)

# now try to reduce the number of clusters
heatmap.2(t(housing1),
          col = col1, keysize=.8, key.par = list(cex=.5),
          trace = "none", key = TRUE, cexCol = 0.75,
          ColSideColors = col2[as.numeric(housing[,"price.new"])+1],
          margins = c(10, 10))
```



PCA

```
pca <- prcomp(housing1)
pca$rotation
```

```
##                      PC1          PC2         PC3          PC4
## bedrooms      -0.27130829 -0.038457629 -0.36324180  0.189225815
## bathrooms     -0.43966664  0.055997681 -0.09217362  0.060750743
## floors        -0.31065079  0.163256563  0.10065788  0.288658775
## waterfront    -0.05212398  0.412624273  0.28272994 -0.450307178
## condition      0.11554810  0.165291671 -0.71456953 -0.363133129
## grade         -0.44187019  0.005398891 -0.04141166 -0.058355492
## sqft_above    -0.46113396  0.110458358 -0.06313243 -0.005025367
## yr_built      -0.27507154 -0.524484902  0.30856097 -0.035491023
## sqft_living15 -0.35657475 -0.001045787 -0.17170504 -0.279229641
## sqft_lot15    -0.09305778  0.004613684  0.29291721 -0.622317069
## renovated     -0.03390324  0.695568436  0.20661065  0.269441773
##                      PC5         PC6         PC7          PC8         PC9
## bedrooms      -0.35399857  0.38097304 -0.58982345  0.148405327 -0.28415445
## bathrooms     -0.02400276 -0.03136323 -0.19745597 -0.290187880  0.55039788
## floors         0.26119663 -0.63287771 -0.15290085  0.218964669 -0.44991255
## waterfront     0.52635327  0.34923995 -0.36719468  0.087195797 -0.04898310
## condition      0.17148620 -0.28248307 -0.04329887 -0.418921865 -0.14596884
## grade          0.13733403  0.02976869  0.32030941  0.009535377  0.16428984
## sqft_above     0.02347765 -0.11999447  0.02195195  0.155862201  0.27028093
## yr_built       0.15415832  0.08015550 -0.08594278 -0.617846360 -0.32086070
## sqft_living15 -0.08523537  0.29681488  0.53230811  0.201937120 -0.39417354
## sqft_lot15    -0.59049724 -0.35879225 -0.16548611  0.041867099 -0.01992272
## renovated     -0.31670370  0.11417429  0.17841050 -0.462680412 -0.17727566
##                     PC10        PC11
```

```
## bedrooms      -0.184355976 -0.04538328
## bathrooms      0.495136527 -0.34426040
## floors         0.112030341 -0.16832503
## waterfront     0.007777977 -0.02000870
## condition     -0.071391029  0.06224075
## grade         -0.705611876 -0.39023862
## sqft_above    -0.078396974  0.80760189
## yr_built      -0.030726793  0.17197688
## sqft_living15  0.433455131 -0.05148543
## sqft_lot15    -0.075144429 -0.07509299
## renovated     -0.068090699  0.06666947
```

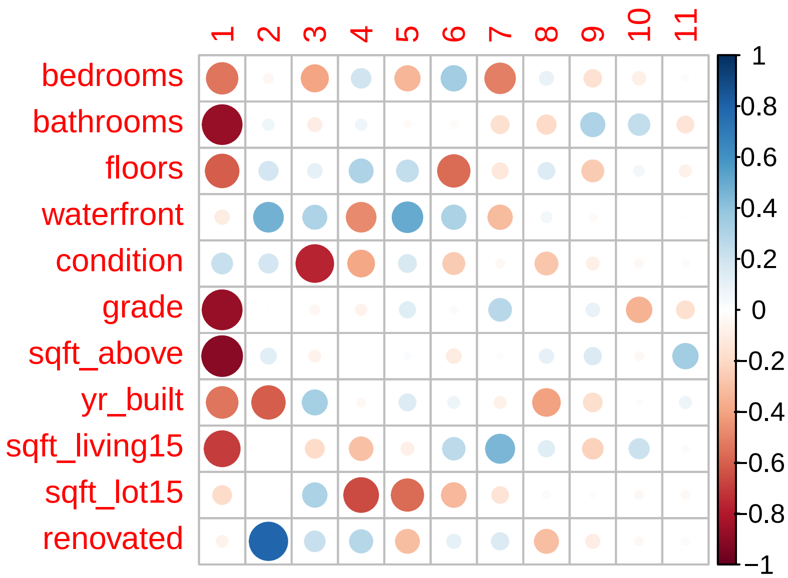
```r
pca$sdev
```

```
##  [1] 1.9592205 1.1478695 1.0762769 1.0476570 0.9526513 0.8940484 0.8482443
##  [8] 0.6586719 0.5603973 0.4859539 0.4212635
```

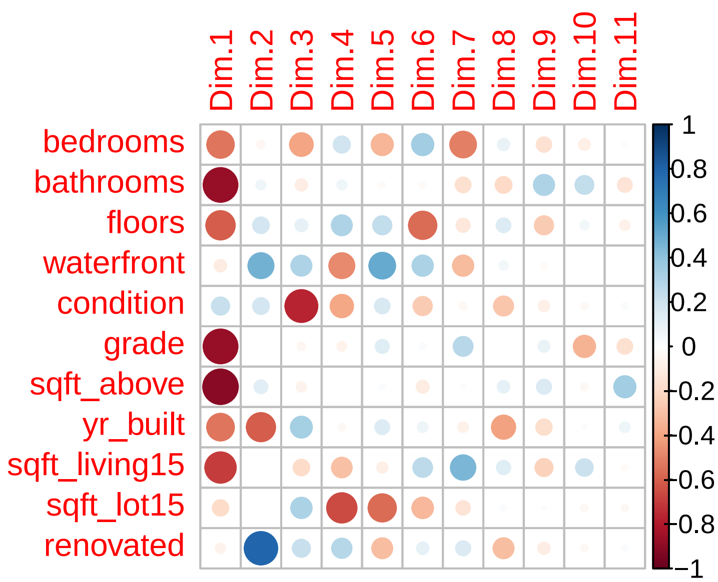
```r
pca$rotation %*% diag(pca$sdev)
```

```
##                      [,1]         [,2]        [,3]        [,4]        [,5]
## bedrooms      -0.53155276 -0.044144339 -0.39094877  0.19824374 -0.33723721
## bathrooms     -0.86140389  0.064278029 -0.09920435  0.06364594 -0.02286626
## floors        -0.60863338  0.187397228  0.10833576  0.30241538  0.24882931
## waterfront    -0.10212236  0.473638813  0.30429571 -0.47176746  0.50143113
## condition      0.22638420  0.189733266 -0.76907471 -0.38043896  0.16336655
## grade         -0.86572112  0.006197222 -0.04457042 -0.06113654  0.13083145
## sqft_above    -0.90346310  0.126791779 -0.06794798 -0.00526486  0.02236601
## yr_built      -0.53892580 -0.602040216  0.33209706 -0.03718242  0.14685913
## sqft_living15 -0.69860855 -0.001200427 -0.18480217 -0.29253688 -0.08119958
## sqft_lot15    -0.18232071  0.005295908  0.31526004 -0.65197482 -0.56253797
## renovated     -0.06642393  0.798421784  0.22237028  0.28228255 -0.30170820
##                      [,6]        [,7]         [,8]        [,9]
## bedrooms       0.34060835 -0.50031439  0.097750423 -0.15923939
## bathrooms     -0.02804025 -0.16749091 -0.191138611  0.30844149
## floors        -0.56582333 -0.12969728  0.144225881 -0.25212979
## waterfront     0.31223743 -0.31147080  0.057433424 -0.02745000
## condition     -0.25255355 -0.03672802 -0.275932074 -0.08180055
## grade          0.02661465  0.27170064  0.006280685  0.09206758
## sqft_above    -0.10728087  0.01862061  0.102662057  0.15146471
## yr_built       0.07166290 -0.07290047 -0.406958054 -0.17980948
## sqft_living15  0.26536688  0.45152733  0.133010313 -0.22089379
## sqft_lot15    -0.32077766 -0.14037265  0.027576683 -0.01116464
## renovated      0.10207735  0.15133569 -0.304754600 -0.09934480
##                      [,10]        [,11]
## bedrooms      -0.089588499 -0.019118321
## bathrooms      0.240613508 -0.145024339
## floors         0.054441577 -0.070909193
## waterfront     0.003779738 -0.008428934
## condition     -0.034692746  0.026219757
## grade         -0.342894818 -0.164393284
## sqft_above    -0.038097312  0.340213195
## yr_built      -0.014931804  0.072447580
## sqft_living15  0.210639196 -0.021688931
## sqft_lot15    -0.036516725 -0.031633935
## renovated     -0.033088938  0.028085413
```

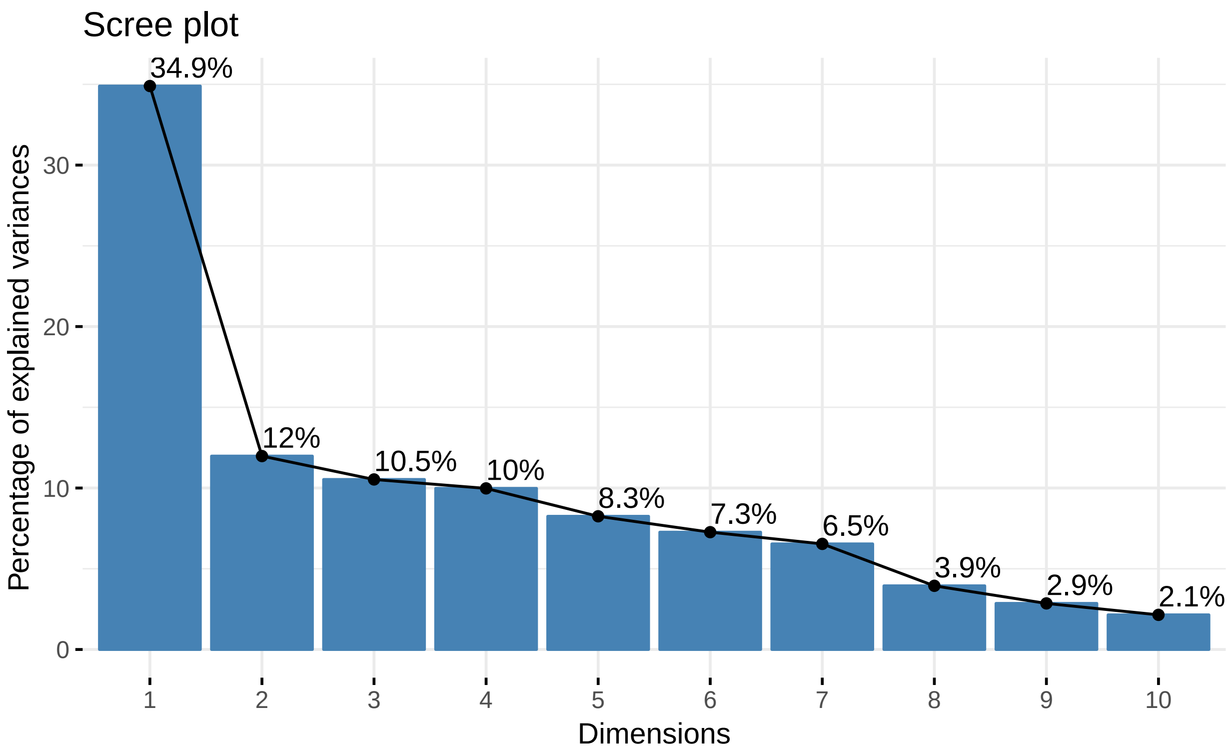
```r
corrplot(pca$rotation %*% diag(pca$sdev))
```



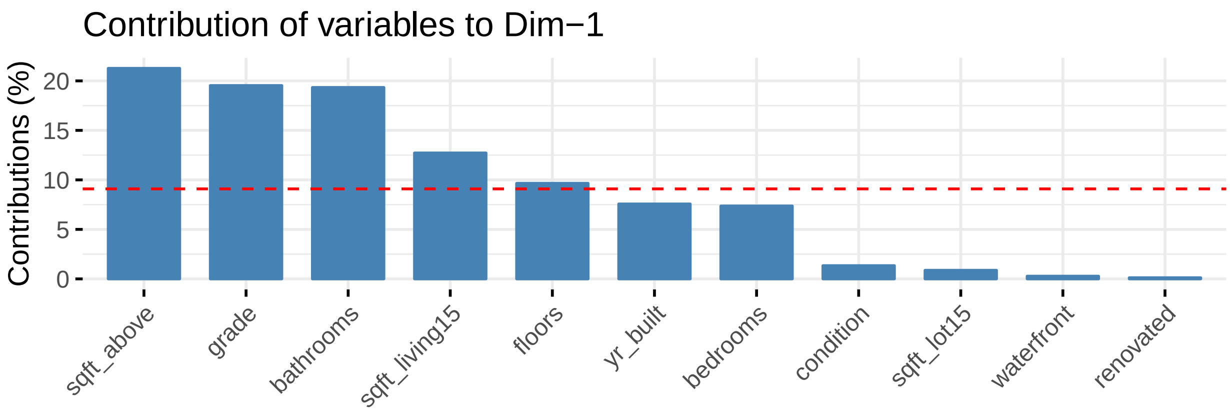

```r
var <- get_pca_var(pca)
corrplot(var$cor)
```

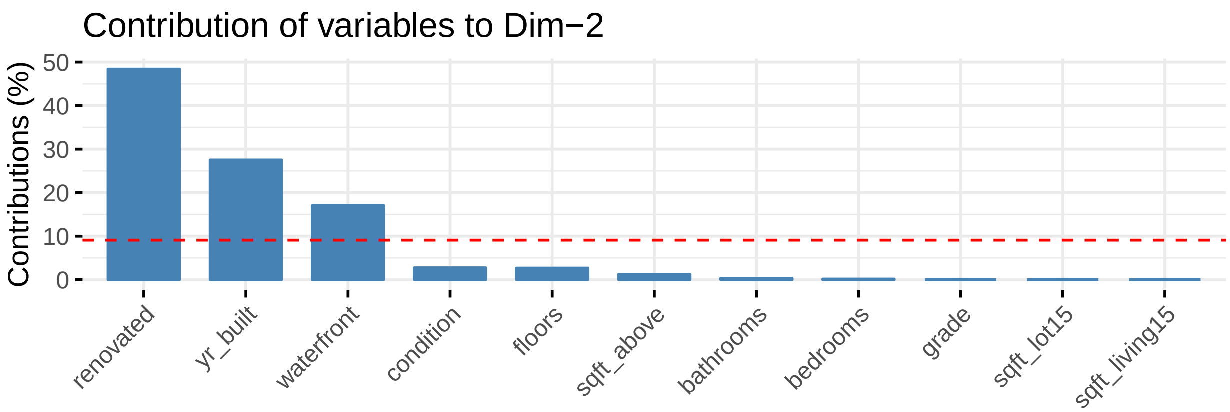



```r
#plots the eigenvalues/variances against the number of dimensions.
fviz_eig(pca, addlabels = TRUE)
```



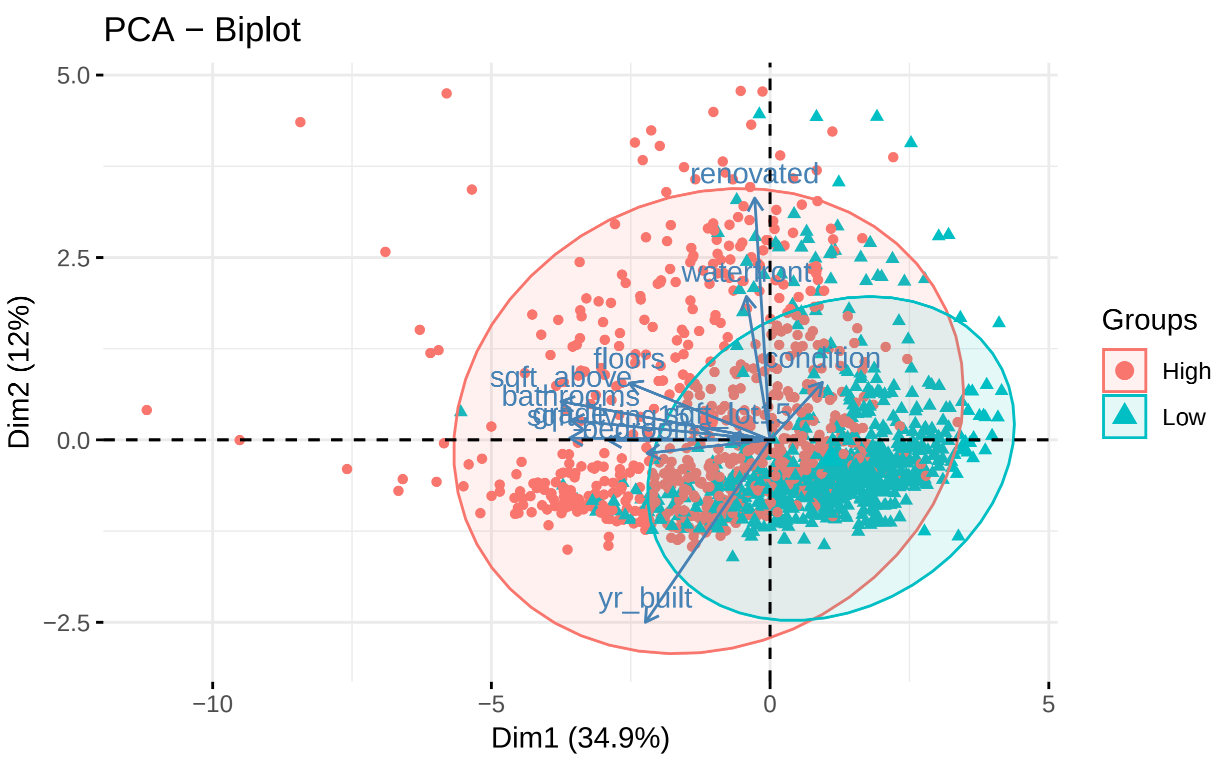

```
a <- fviz_contrib(pca, choice = "var", axes = 1)
b <- fviz_contrib(pca, choice = "var", axes = 2)
#visualize the contribution of variables from the results of PCA.
grid.arrange(a, b, nrow = 2)
```



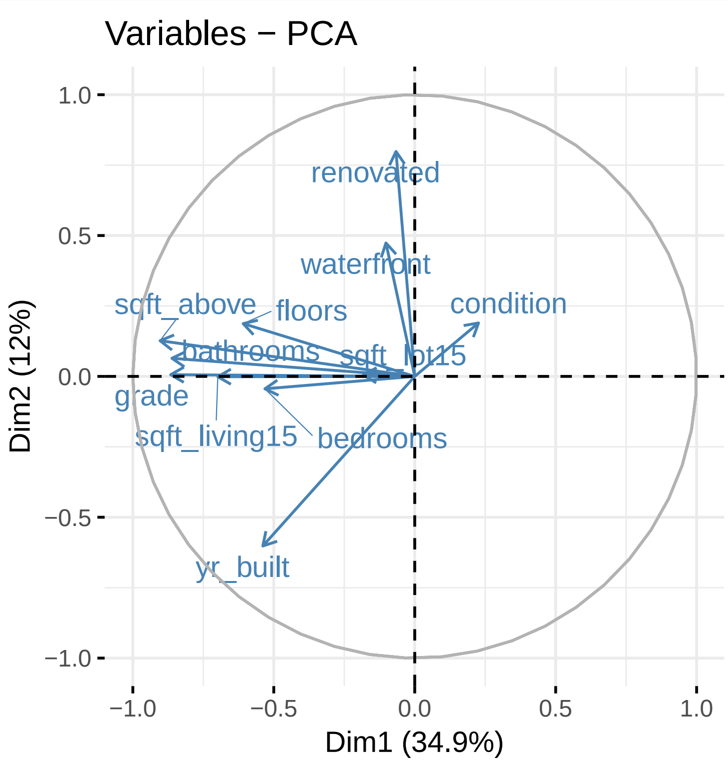


`sqrt_above`, `grade`, `bathrooms`, `sqrt_living15`, and `floors` contribute more to the first principal component, compared to other variables. `renovated`, `yr_built`, and `waterfront` contribute more to the second pricipal ccomponent.

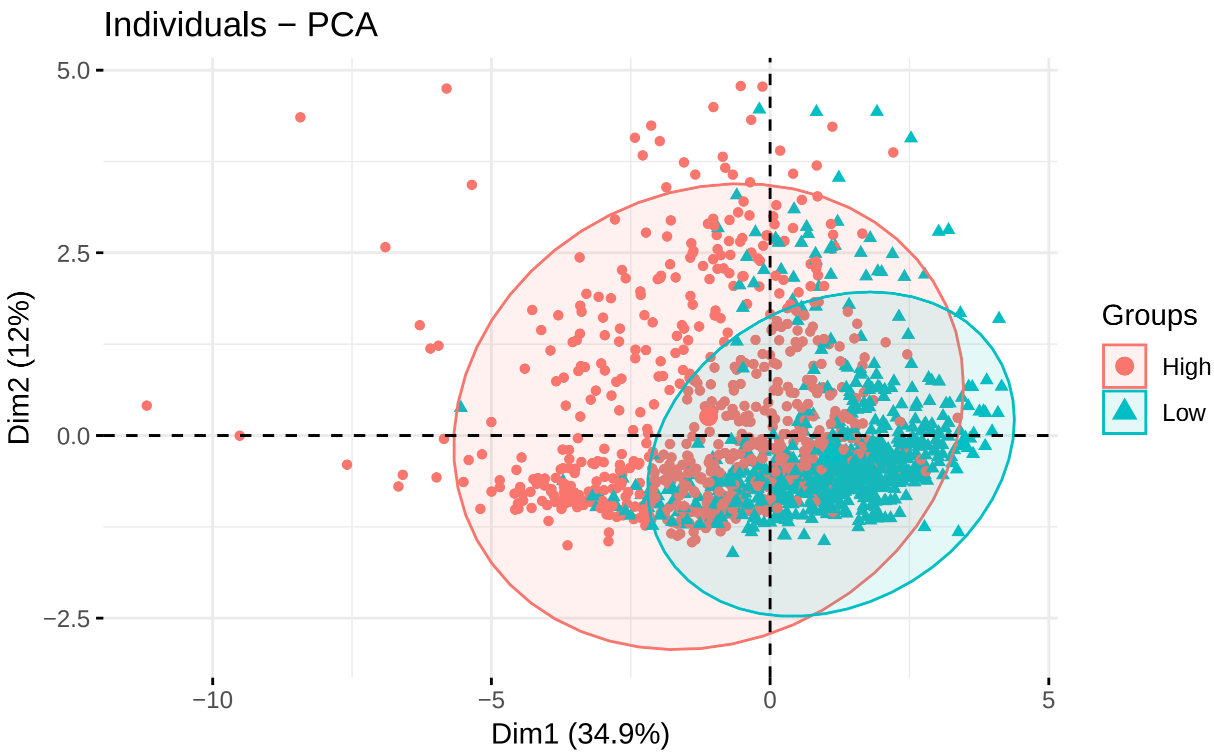
```
# to obtain the biplot of individuals and variables.
fviz_pca_biplot(pca, axes = c(1,2),
                habillage = housing$price.new,
                label = c("var"),
                addEllipses = TRUE)
```




```r
fviz_pca_var(pca, col.var = "steelblue", repel = TRUE)
```




```r
fviz_pca_ind(pca,
             habillage = housing$price.new,
             label = "none",
             addEllipses = TRUE)
```

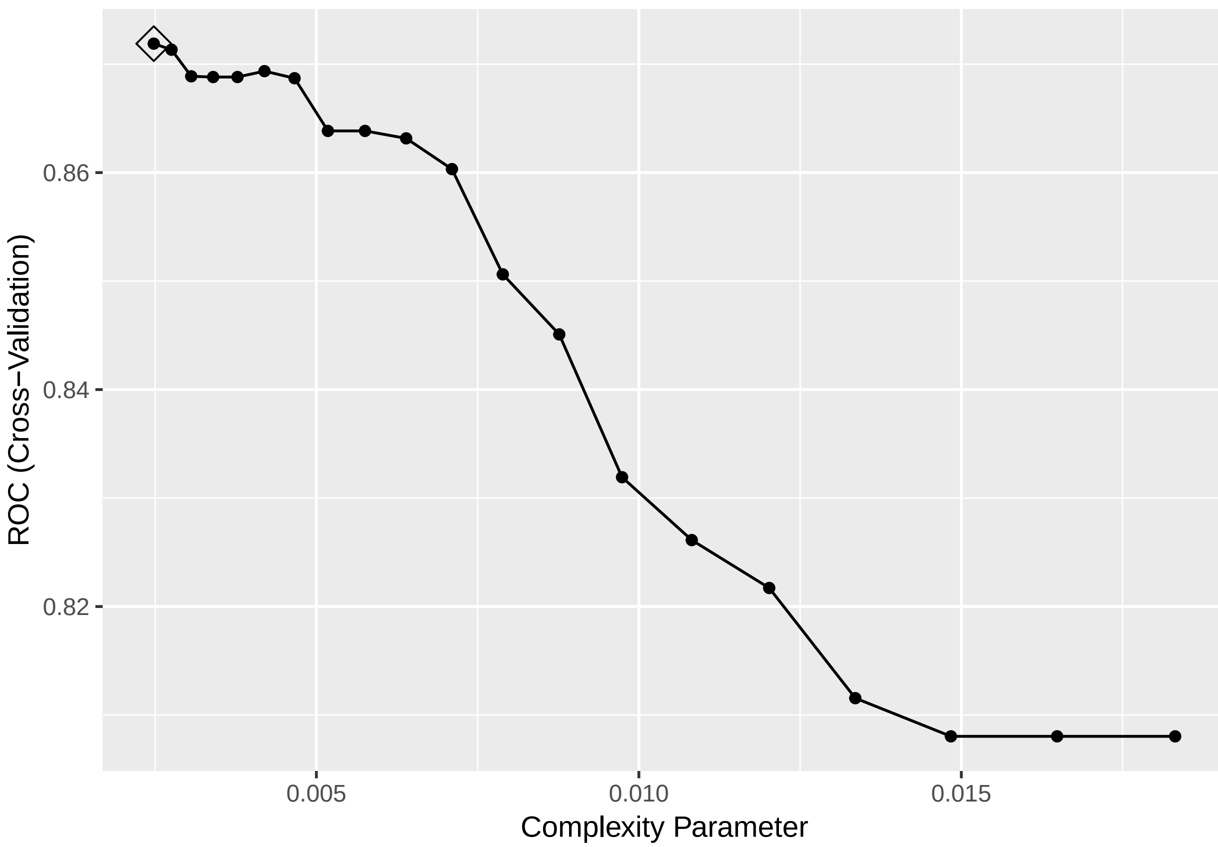


With only two principal components, we can distinguish the two classes reasonably well.

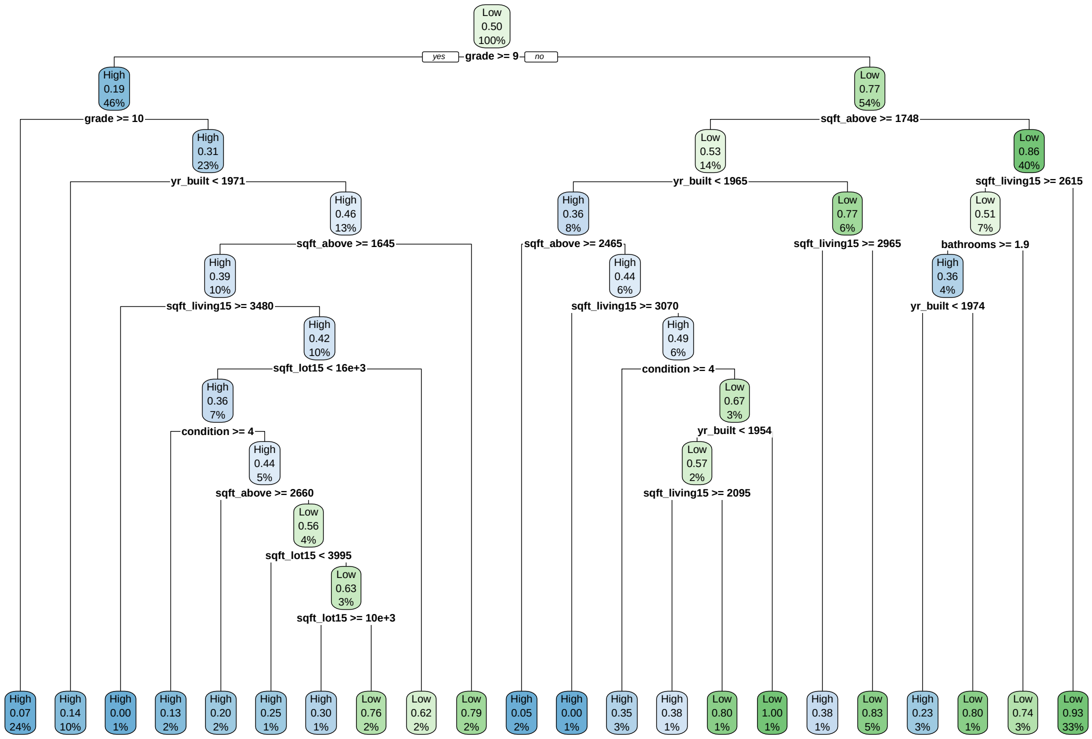
Classification tree

CART

```
ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
set.seed(1)
rpart.fit <- train(price.new~., housing,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-4, len = 20))),
                   trControl = ctrl,
                   metric = "ROC")
ggplot(rpart.fit, highlight = TRUE)
```




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



```r
# Test set performance
rpart.pred.prob <- predict(rpart.fit, newdata = housing[-rowTrain,], type = "prob")[,1]
rpart.pred <- rep("Low", length(rpart.pred.prob))
rpart.pred[rpart.pred.prob>0.5] <- "High"
confusionMatrix(data = as.factor(rpart.pred),
                reference = housing$price.new[-rowTrain],
                positive = "High")
```

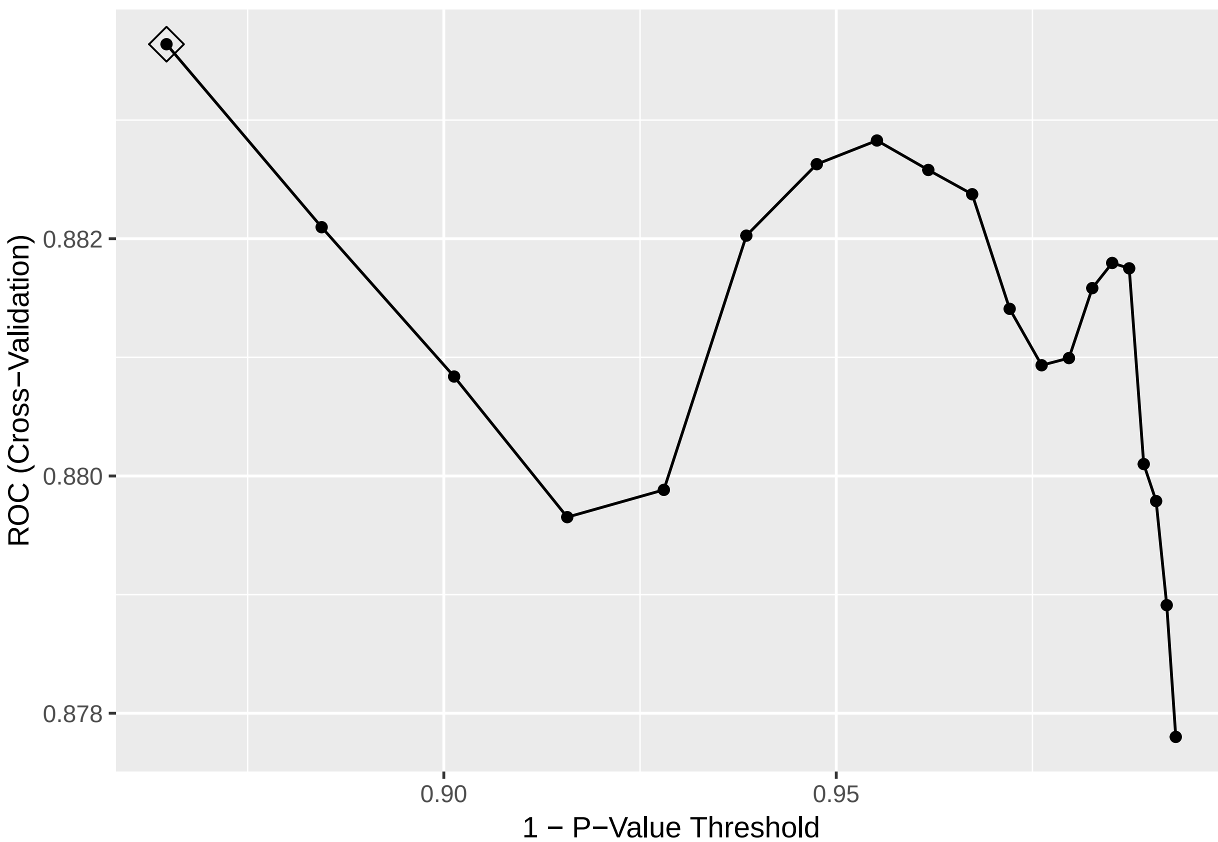
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  122  31
##       Low    21 114
##
##                Accuracy : 0.8194
##                  95% CI : (0.7701, 0.8621)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.639
##
##  Mcnemar's Test P-Value : 0.212
##
##             Sensitivity : 0.8531
##             Specificity : 0.7862
##          Pos Pred Value : 0.7974
##          Neg Pred Value : 0.8444
##              Prevalence : 0.4965
##          Detection Rate : 0.4236
##    Detection Prevalence : 0.5312
```

```
##        Balanced Accuracy : 0.8197
##
##         'Positive' Class : High
##
```

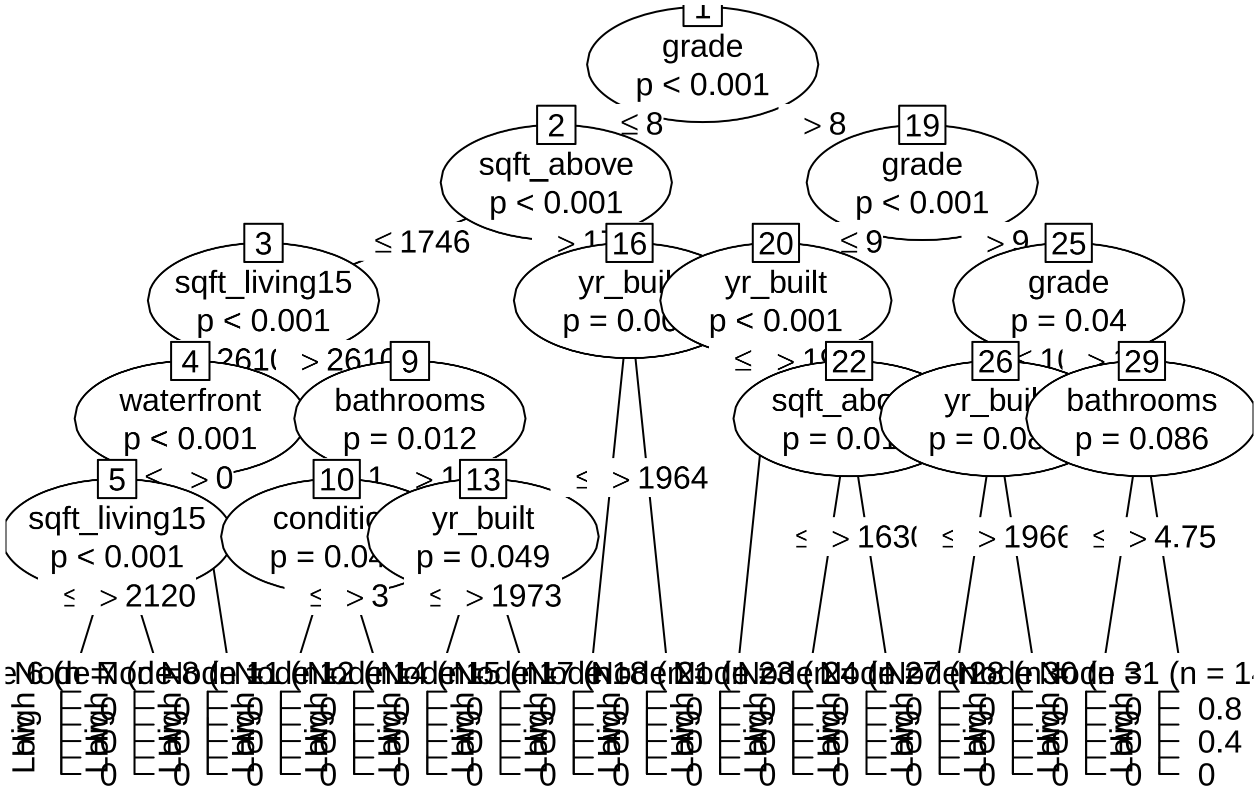
```r
# 0.8194
```

CIT

```r
set.seed(1)
ctree.fit <- train(price.new~., housing,
                   subset = rowTrain,
                   method = "ctree",
                   tuneGrid = data.frame(mincriterion = 1-exp(seq(-5, -2, length = 20))),
                   metric = "ROC",
                   trControl = ctrl)
ggplot(ctree.fit, highlight = TRUE)
```




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




```r
# Test set performance
ctree.pred.prob <- predict(ctree.fit, newdata = housing[-rowTrain,], type = "prob")[,1]
ctree.pred <- rep("Low", length(ctree.pred.prob))
ctree.pred[ctree.pred.prob>0.5] <- "High"
confusionMatrix(data = as.factor(ctree.pred),
                reference = housing$price.new[-rowTrain],
                positive = "High")
```

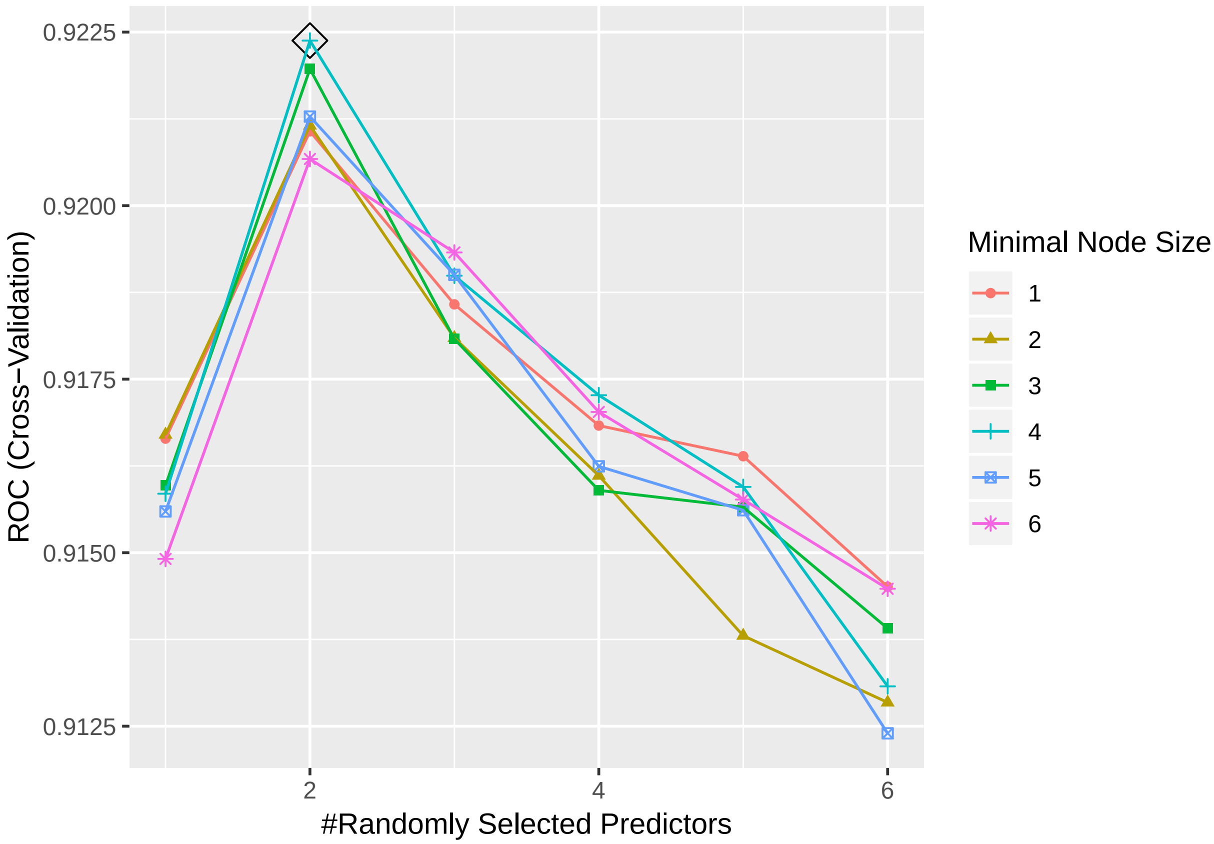
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  126  42
##       Low    17 103
##
##                Accuracy : 0.7951
##                  95% CI : (0.7439, 0.8402)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5908
##
##  Mcnemar's Test P-Value : 0.001781
##
##             Sensitivity : 0.8811
##             Specificity : 0.7103
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.8583
##              Prevalence : 0.4965
```

```
##          Detection Rate : 0.4375
##    Detection Prevalence : 0.5833
##       Balanced Accuracy : 0.7957
##
##        'Positive' Class : High
##
```

```r
# 0.7951
```

Random Forest

```r
rf.grid <- expand.grid(mtry = 1:6,
                       splitrule = "gini",
                       min.node.size = 1:6)
set.seed(1)
rf.fit <- train(price.new~., housing,
                subset = rowTrain,
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",
                trControl = ctrl)

ggplot(rf.fit, highlight = TRUE)
```

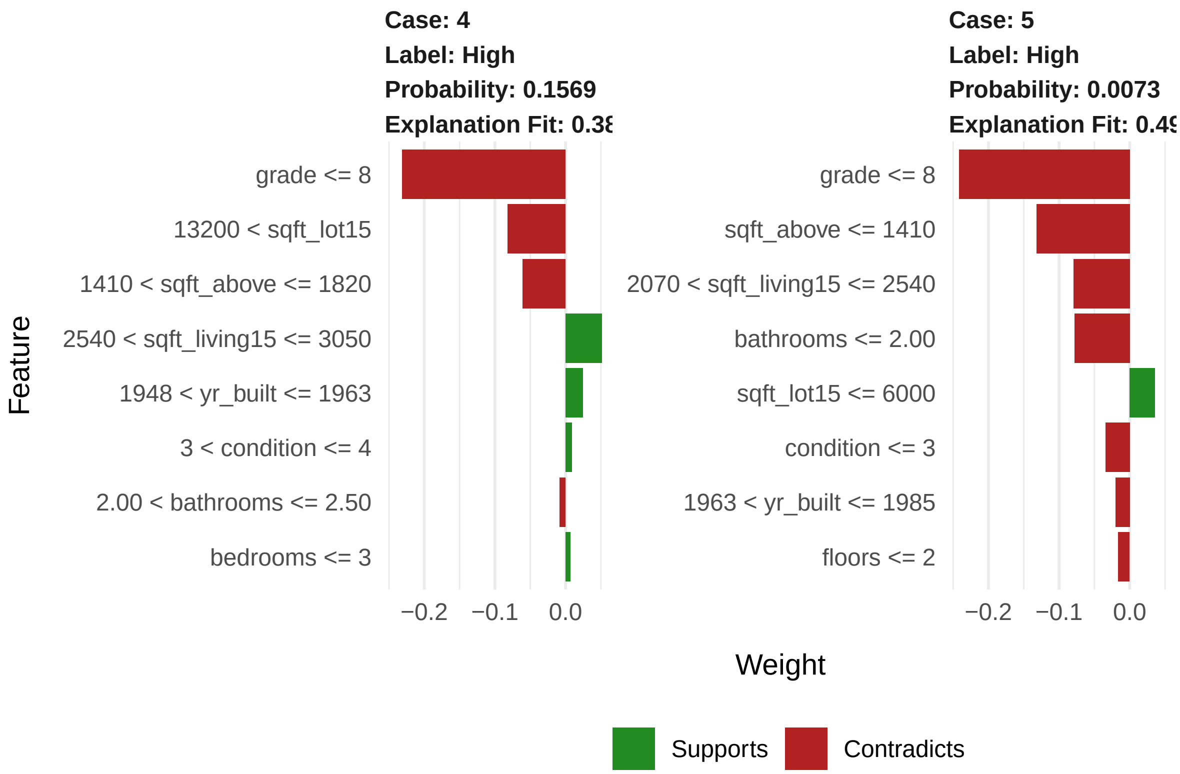



```r
# Test set performance
rf.pred.prob <- predict(rf.fit, newdata = housing[-rowTrain,], type = "prob")[,1]
```

```r
rf.pred <- rep("Low", length(rf.pred.prob))
rf.pred[rf.pred.prob>0.5] <- "High"
confusionMatrix(data = as.factor(rf.pred),
                reference = housing$price.new[-rowTrain],
                positive = "High")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  120  20
##       Low    23 125
##
##                Accuracy : 0.8507
##                  95% CI : (0.8042, 0.8898)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7013
##
##  Mcnemar's Test P-Value : 0.7604
##
##             Sensitivity : 0.8392
##             Specificity : 0.8621
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.8446
##              Prevalence : 0.4965
##          Detection Rate : 0.4167
##    Detection Prevalence : 0.4861
##       Balanced Accuracy : 0.8506
##
##        'Positive' Class : High
##
```

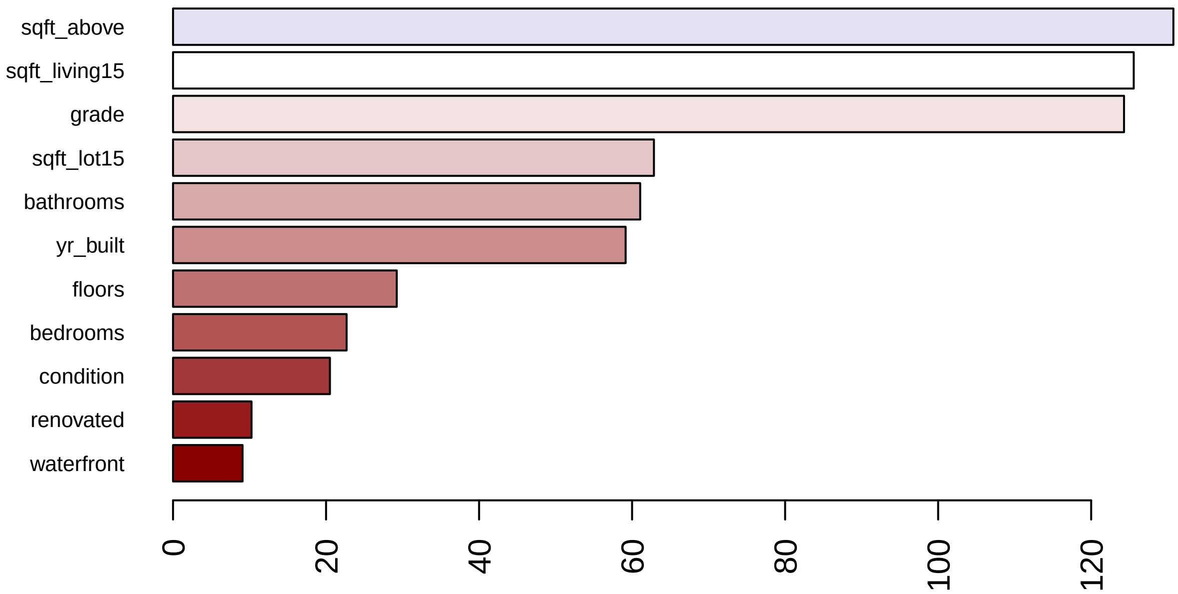
```r
# 0.8056

# Explain your prediction
new_obs <- housing[-rowTrain,-12][1:2,]
explainer.rf <- lime(housing[rowTrain,-12], rf.fit)
explanation.rf <- explain(new_obs, explainer.rf, n_features = 8,
                          labels = "High")
plot_features(explanation.rf)
```



The optimal mtry is 2 and the minimal node size picked up by the optimal model is 3.

Variable importance of random forest model

```
set.seed(1)
rf2.final.imp <- ranger(price.new~., housing,
                        mtry = 2, splitrule = "gini",
                        min.node.size = 3,
                        importance = "impurity")

barplot(sort(ranger::importance(rf2.final.imp), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("darkred","white","darkblue"))(19))
```

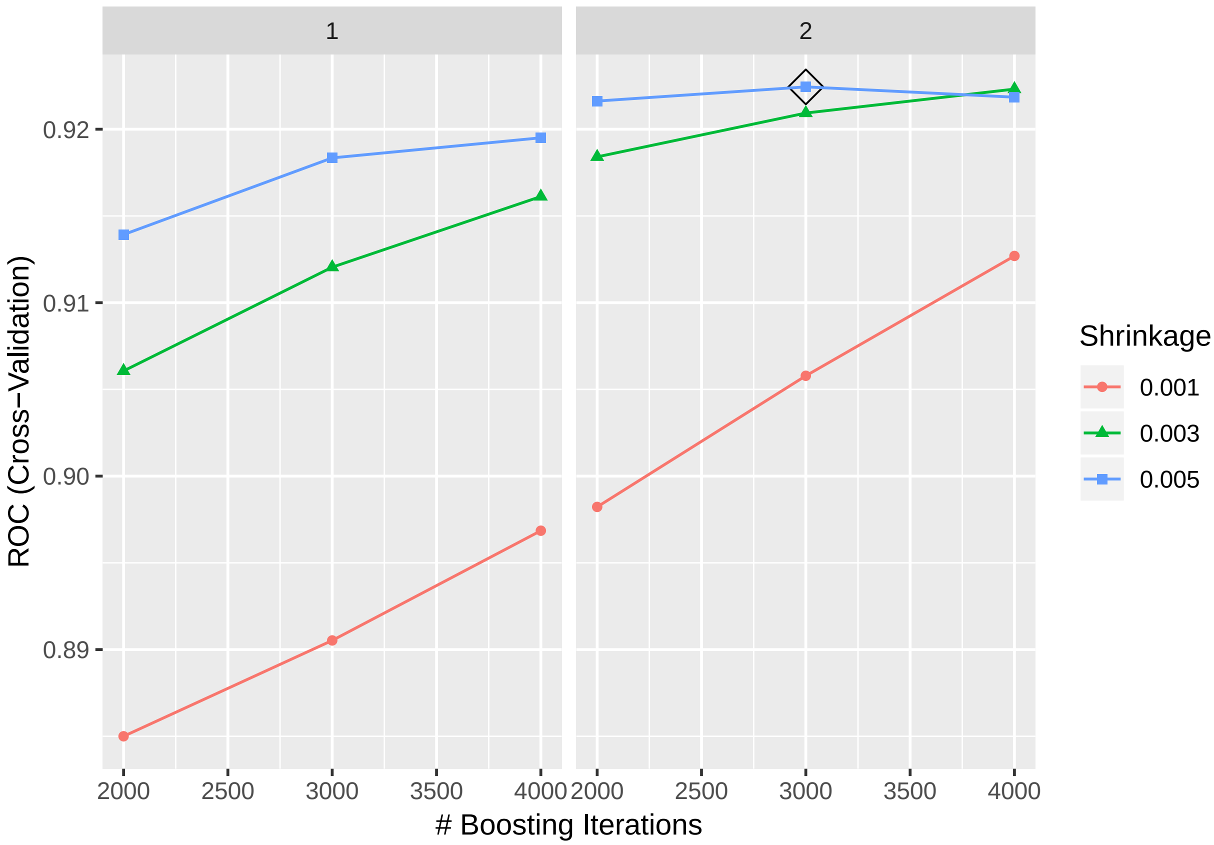


Using node impurity as the measure of varaible importance, the top three important variable are `grade`, `sqft_above`, and `sqft_living15`.

Boosting

Binomial loss

```
gbmB.grid <- expand.grid(n.trees = c(2000,3000,4000),
                         interaction.depth = 1:2,
                         shrinkage = c(0.001,0.003,0.005),
                         n.minobsinnode = 1)
set.seed(1)
# Binomial loss function
gbmB.fit <- train(price.new~., housing,
                  subset = rowTrain,
                  tuneGrid = gbmB.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "bernoulli",
                  metric = "ROC",
                  verbose = FALSE)

ggplot(gbmB.fit, highlight = TRUE)
```

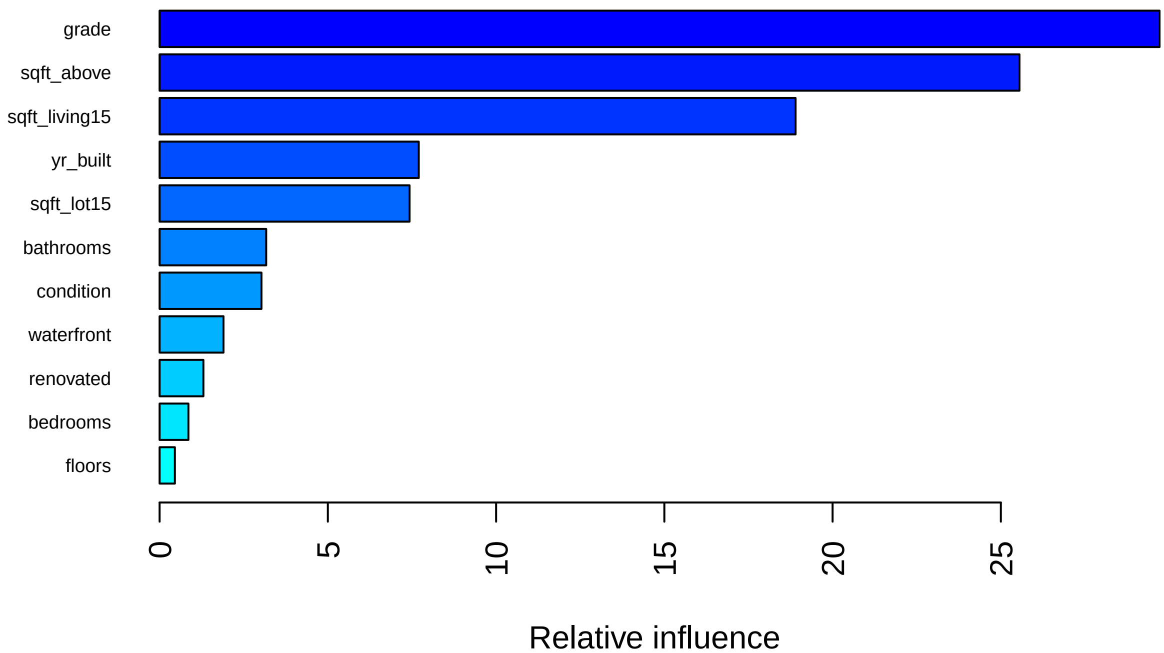



```r
# Test set performance
gbmB.pred.prob <- predict(gbmB.fit, newdata = housing[-rowTrain,], type = "prob")[,1]
gbmB.pred <- rep("Low", length(gbmB.pred.prob))
gbmB.pred[gbmB.pred.prob>0.5] <- "High"
confusionMatrix(data = as.factor(gbmB.pred),
                reference = housing$price.new[-rowTrain],
                positive = "High")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  121  23
##       Low    22 122
##
##                Accuracy : 0.8438
##                  95% CI : (0.7966, 0.8837)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6875
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8462
##             Specificity : 0.8414
##          Pos Pred Value : 0.8403
```

```
##          Neg Pred Value : 0.8472
##              Prevalence : 0.4965
##          Detection Rate : 0.4201
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : High
##
```

```
# 0.8333

#Variable importance
summary(gbmB.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```



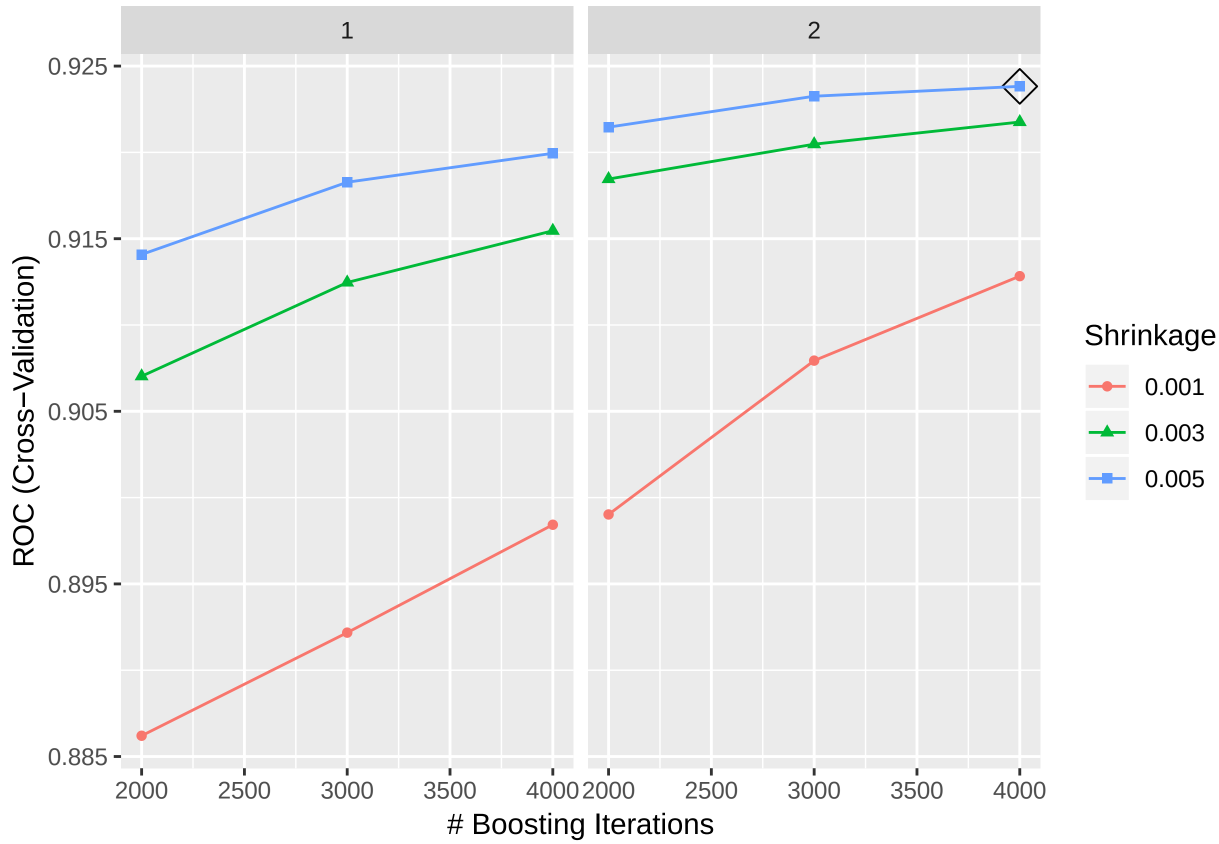

```
##                           var    rel.inf
## grade                   grade 29.7155156
## sqft_above         sqft_above 25.5522851
## sqft_living15 sqft_living15 18.8998282
## yr_built             yr_built  7.7012347
## sqft_lot15         sqft_lot15  7.4277080
## bathrooms           bathrooms  3.1671985
## condition           condition  3.0267079
## waterfront         waterfront  1.8977893
## renovated           renovated  1.3013290
## bedrooms             bedrooms  0.8561193
## floors                 floors  0.4542846
```

The top three important variable are `grade`, `sqft_above`, and `sqft_living15`.

AdaBoost

```
gbmA.grid <- expand.grid(n.trees = c(2000,3000,4000),
                         interaction.depth = 1:2,
```

```r
                         shrinkage = c(0.001,0.003,0.005),
                         n.minobsinnode = 1)
set.seed(1)
# Adaboost loss function
gbmA.fit <- train(price.new~., housing,
                  subset = rowTrain,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

ggplot(gbmA.fit, highlight = TRUE)
```

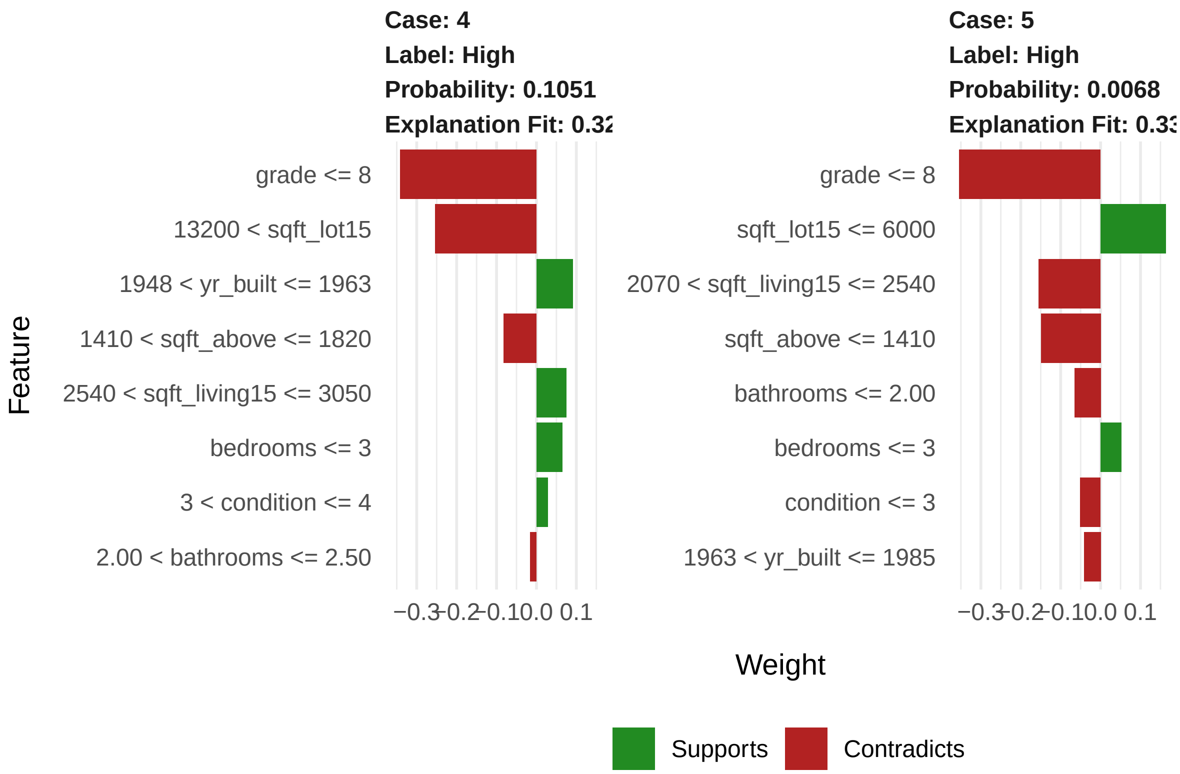



```r
# Test set performance
gbmA.pred.prob <- predict(gbmA.fit, newdata = housing[-rowTrain,], type = "prob")[,1]
gbmA.pred <- rep("Low", length(gbmA.pred.prob))
gbmA.pred[gbmA.pred.prob>0.5] <- "High"
confusionMatrix(data = as.factor(gbmA.pred),
                reference = housing$price.new[-rowTrain],
                positive = "High")
```

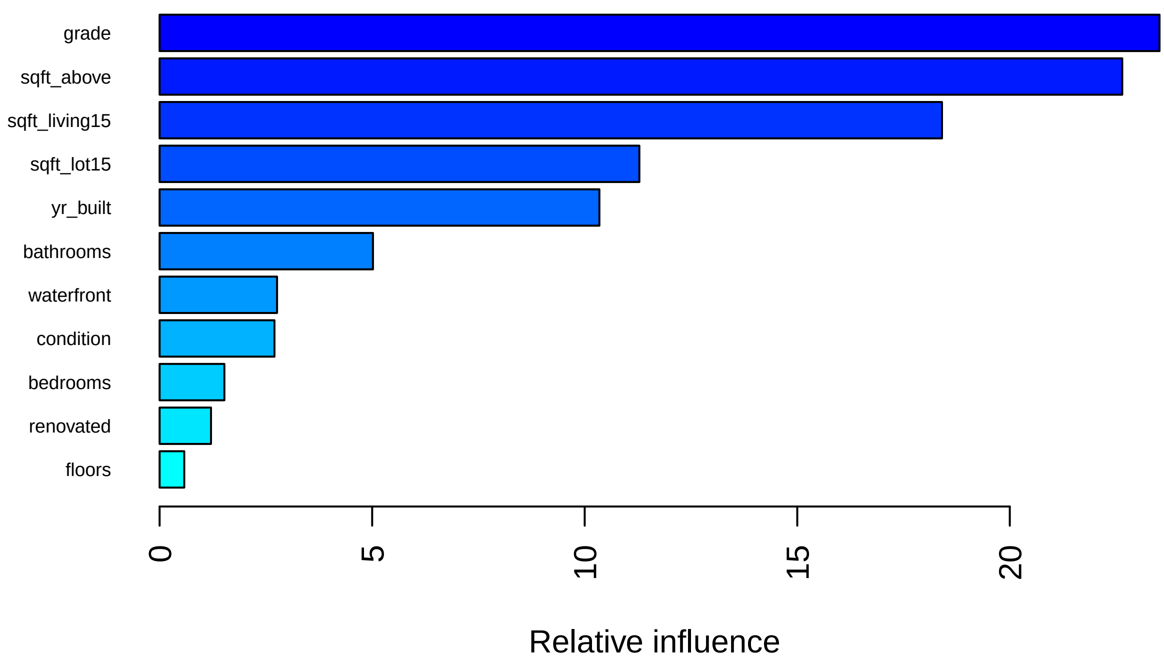
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
```

```
##       High  123  24
##       Low    20 121
##
##                Accuracy : 0.8472
##                  95% CI : (0.8004, 0.8867)
##     No Information Rate : 0.5035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6945
##
##  Mcnemar's Test P-Value : 0.6511
##
##             Sensitivity : 0.8601
##             Specificity : 0.8345
##          Pos Pred Value : 0.8367
##          Neg Pred Value : 0.8582
##              Prevalence : 0.4965
##          Detection Rate : 0.4271
##    Detection Prevalence : 0.5104
##       Balanced Accuracy : 0.8473
##
##        'Positive' Class : High
##
```

```
# 0.8333

# Explain your prediction
new_obs <- housing[-rowTrain,-12][1:2,]
explainer.gbm <- lime(housing[rowTrain,-12], gbmA.fit)
explanation.gbm <- explain(new_obs, explainer.gbm, n_features = 8,
                           labels = "High")
plot_features(explanation.gbm)
```




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



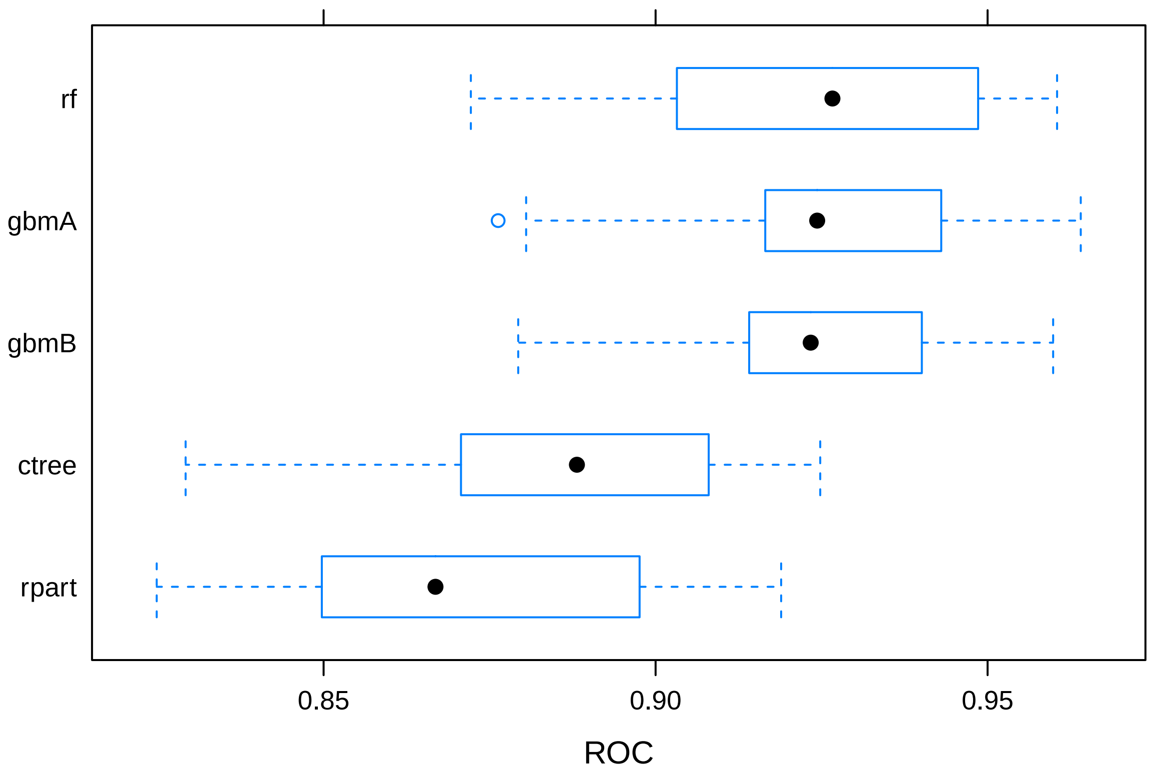

```
##                         var    rel.inf
## grade                 grade 23.5222429
## sqft_above       sqft_above 22.6450071
## sqft_living15 sqft_living15 18.4050638
## sqft_lot15       sqft_lot15 11.2857504
## yr_built           yr_built 10.3458517
```

```
## bathrooms         bathrooms  5.0192286
## waterfront       waterfront  2.7627767
## condition         condition  2.7019817
## bedrooms           bedrooms  1.5235294
## renovated         renovated  1.2087462
## floors               floors  0.5798215
```

The top three important variable are also `grade`, `sqft_above`, and `sqft_living15`.

Resamples

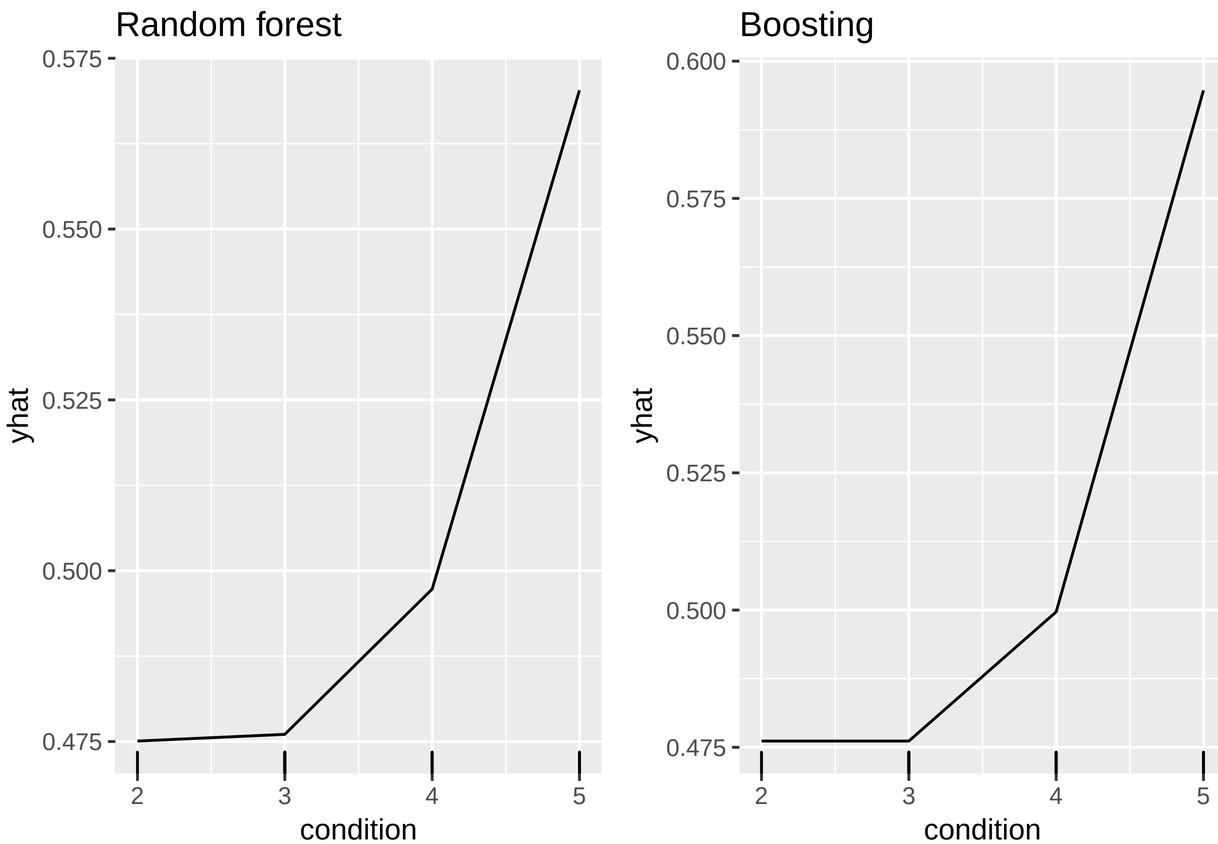
```
resamp.tree <- resamples(list(rf = rf.fit,
                         gbmA = gbmA.fit,
                         gbmB = gbmB.fit,
                         rpart = rpart.fit,
                         ctree = ctree.fit))
bwplot(resamp.tree, metric = "ROC")
```



Variable importance - PDP

```
pdp.rf <- rf.fit %>%
  partial(pred.var = "condition",
          grid.resolution = 100,
          prob = TRUE) %>%
  autoplot(rug = TRUE, train = housing[rowTrain,]) +
  ggtitle("Random forest")
```

```r
pdp.gbm <- gbmA.fit %>%
  partial(pred.var = "condition",
          grid.resolution = 100,
          prob = TRUE) %>%
  autoplot(rug = TRUE, train = housing[rowTrain,]) +
  ggtitle("Boosting")

grid.arrange(pdp.rf, pdp.gbm, nrow = 1)
```



Test data performance (Tree)

```r
roc.rpart <- roc(housing$price.new[-rowTrain], rpart.pred.prob)
roc.ctree <- roc(housing$price.new[-rowTrain], ctree.pred.prob)
roc.rf <- roc(housing$price.new[-rowTrain], rf.pred.prob)
roc.gbmA <- roc(housing$price.new[-rowTrain], gbmA.pred.prob)
roc.gbmB <- roc(housing$price.new[-rowTrain], gbmB.pred.prob)


plot(roc.rpart)
plot(roc.ctree, add = TRUE, col = 2)
plot(roc.rf, add = TRUE, col = 3)
plot(roc.gbmA, add = TRUE, col = 4)
plot(roc.gbmB, add = TRUE, col = 5)
```

```r
auc <- c(roc.rpart$auc[1],  roc.ctree$auc[1],
         roc.rf$auc[1], roc.gbmA$auc[1], roc.gbmB$auc[1])

modelNames <- c("rpart_caret","ctree","rf","gbmA","gbmB")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:6, lwd = 2)
```