code

Qi Yuchen, yq2279

2020/5/16

Exploratory analysis

data imput

First, we identify the missing values in the dataset. As is shown in the table below, there is no variable containing missing data.

```
df.raw = read_csv("winequality-red.csv")
```

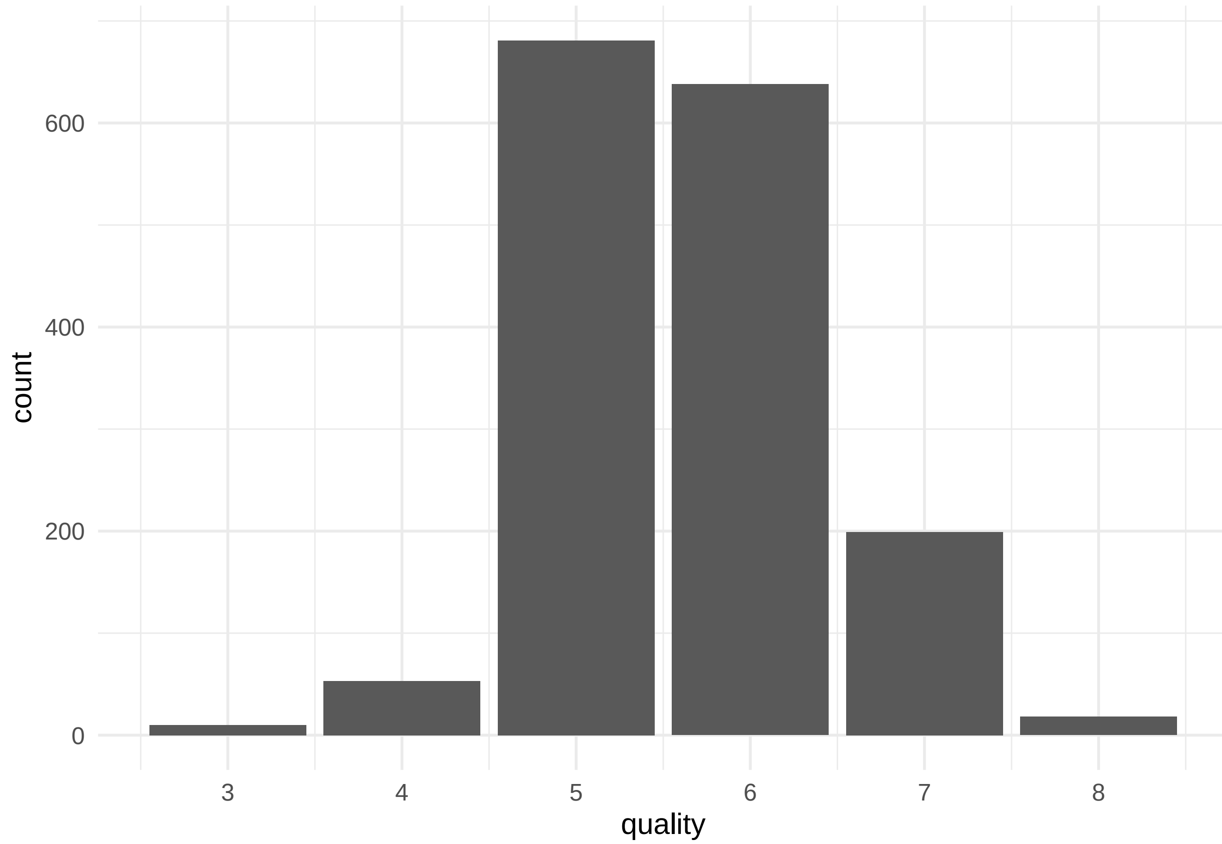
```
## Parsed with column specification:
## cols(
##   `fixed acidity` = col_double(),
##   `volatile acidity` = col_double(),
##   `citric acid` = col_double(),
##   `residual sugar` = col_double(),
##   chlorides = col_double(),
##   `free sulfur dioxide` = col_double(),
##   `total sulfur dioxide` = col_double(),
##   density = col_double(),
##   pH = col_double(),
##   sulphates = col_double(),
##   alcohol = col_double(),
##   quality = col_double()
## )
```

```
# check NA data
df.na = is.na(df.raw)
var.na = colSums(df.na)
var.na
```

```
##        fixed acidity     volatile acidity          citric acid
##                    0                    0                    0
##       residual sugar            chlorides  free sulfur dioxide
##                    0                    0                    0
## total sulfur dioxide              density                   pH
##                    0                    0                    0
##            sulphates              alcohol              quality
##                    0                    0                    0
```

Then we check the response variable `quality`. As the distribution is not balanced and we are intereted in whether the wine is good or not, we divide it into two classes, poor (quality < 5.5) or good (quality > 5.5). After cleaning the names of the variables, we divide the data into traning and testing data.

```
df.raw %>%
  ggplot(aes(x = quality)) +
  geom_bar() +
  scale_x_continuous(breaks = seq(3,8,1))
```




```
df = df.raw %>%
  janitor::clean_names() %>%
  mutate(quality = as.factor(quality)) %>%
  mutate(quality = recode(quality, '3' = "poor",'4' = "poor",'5' = "poor",'6' = "good",'7' =  "good",'8
```

```
## Warning in FUN(X[[i]], ...): strings not representable in native encoding
## will be translated to UTF-8
```

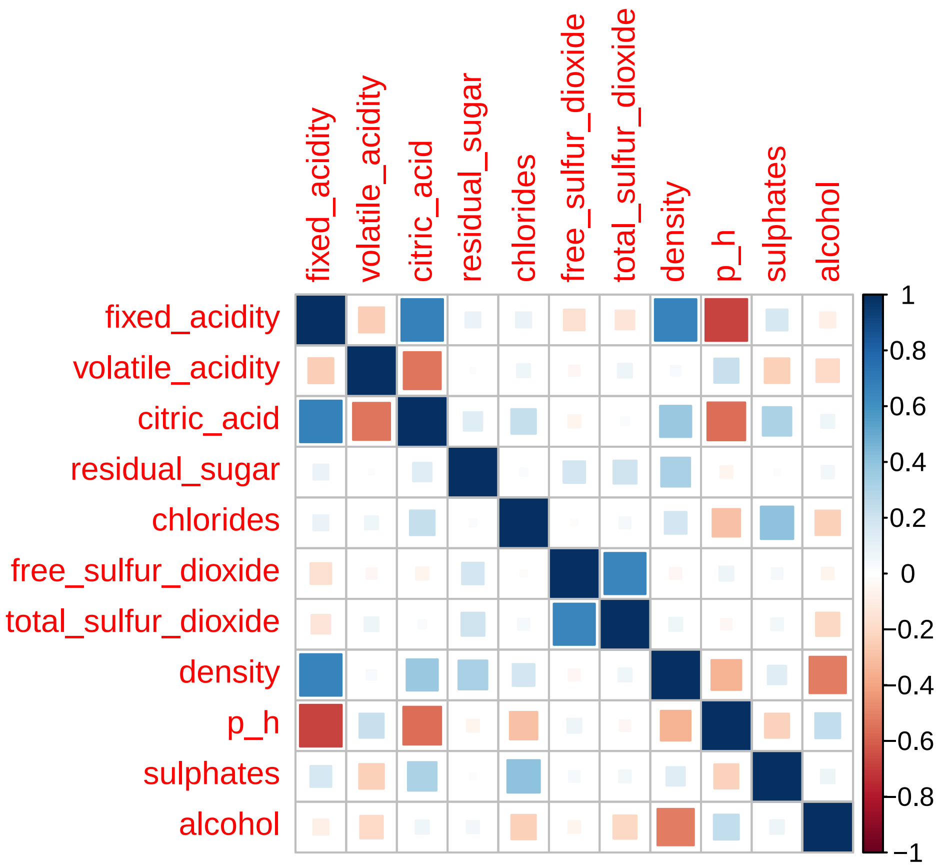
```
set.seed(1)
rowTrain <- createDataPartition(y = df$quality,
                                p = 2/3,
                                list = FALSE)
df.train = df[rowTrain,]
x = model.matrix(quality~., df.train)[,-1]
y = df.train$quality

df.test = df[-rowTrain,]

summary(df$quality) # 6 levels
```

```
## poor good
##  744  855
```

Correlations

11 predictors are all numerical variables. There is no strong correlation (>0.7) between them.

```
var.numerical = df.train %>% dplyr::select_if(is.numeric) %>% as.matrix()
var.cor = cor(var.numerical)

corrplot(cor(var.numerical), method = "square", type = "full")
```

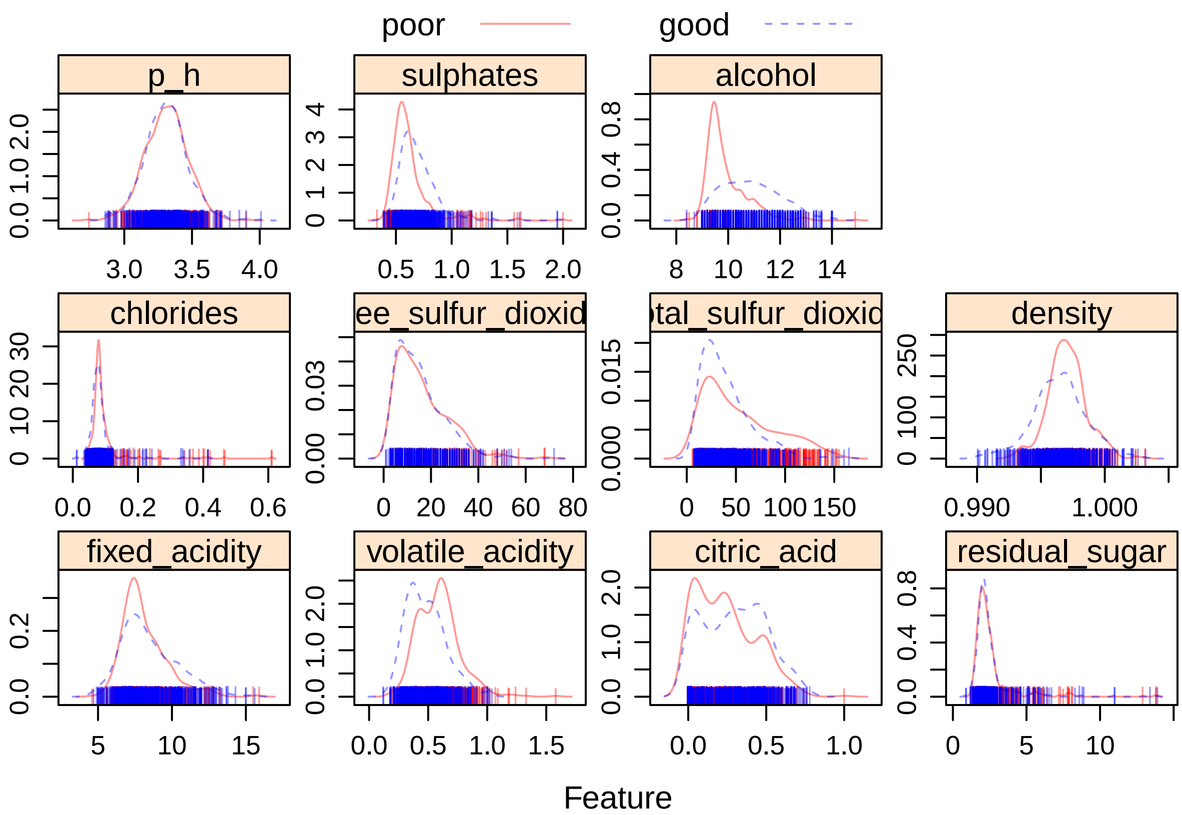



```
which((var.cor > 0.7 & var.cor < 1), arr.ind = TRUE)
```

```
##      row col
```

Scatter plot

```
theme1 <- transparentTheme(trans = .4)
trellis.par.set(theme1)

featurePlot(x,
            y,
            scales = list(x=list(relation="free"),
                          y=list(relation="free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



Models

```
ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
```

LDA and QDA

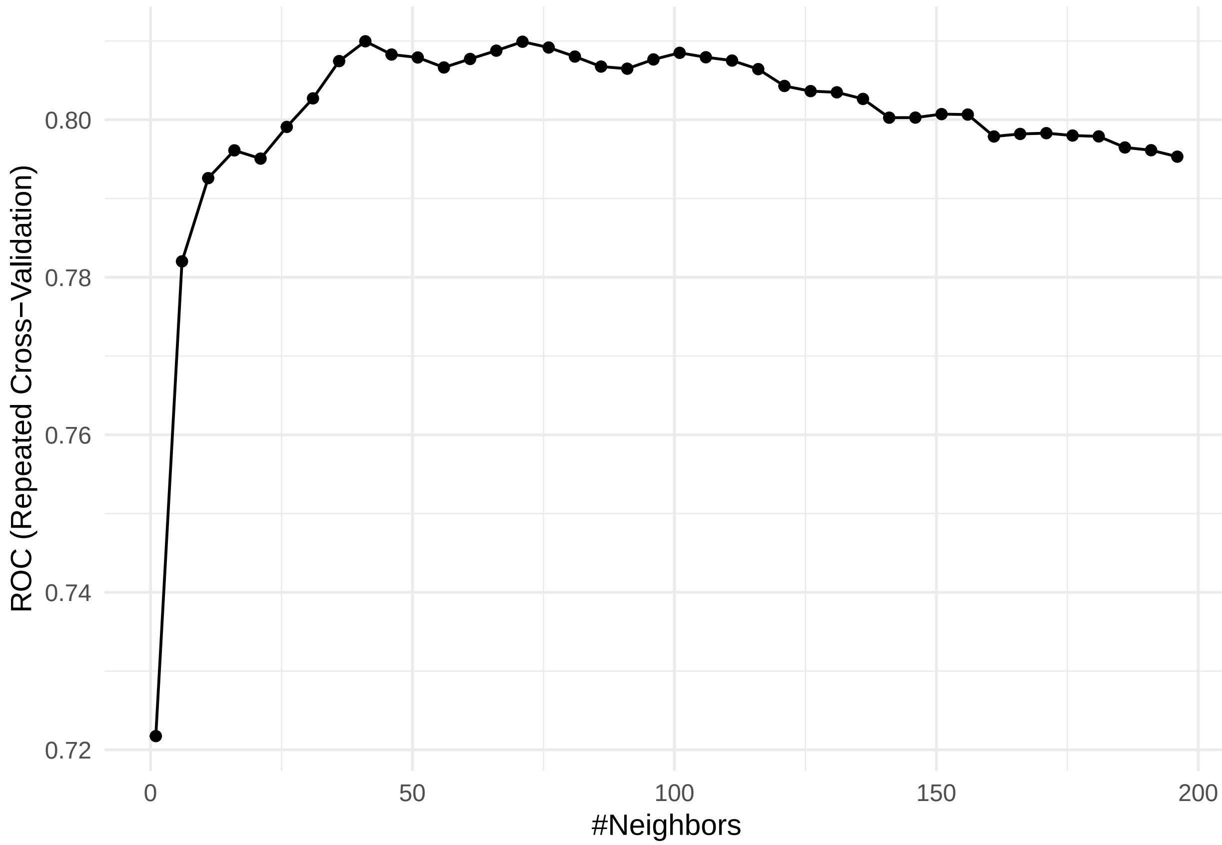
```
set.seed(1)
model.lda <- train(quality~., df,
                   subset = rowTrain,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
set.seed(1)
model.qda <- train(quality~., df,
                   subset = rowTrain,
                   method = "qda",
                   metric = "ROC",
                   trControl = ctrl)
```

KNN

```r
set.seed(1)
knn.fit <- train(quality~., df,
                 subset = rowTrain,
                 method = "knn",
                 preProcess = c("center","scale"),
                 tuneGrid = data.frame(k = seq(1,200,by=5)),
                 trControl = ctrl)
```

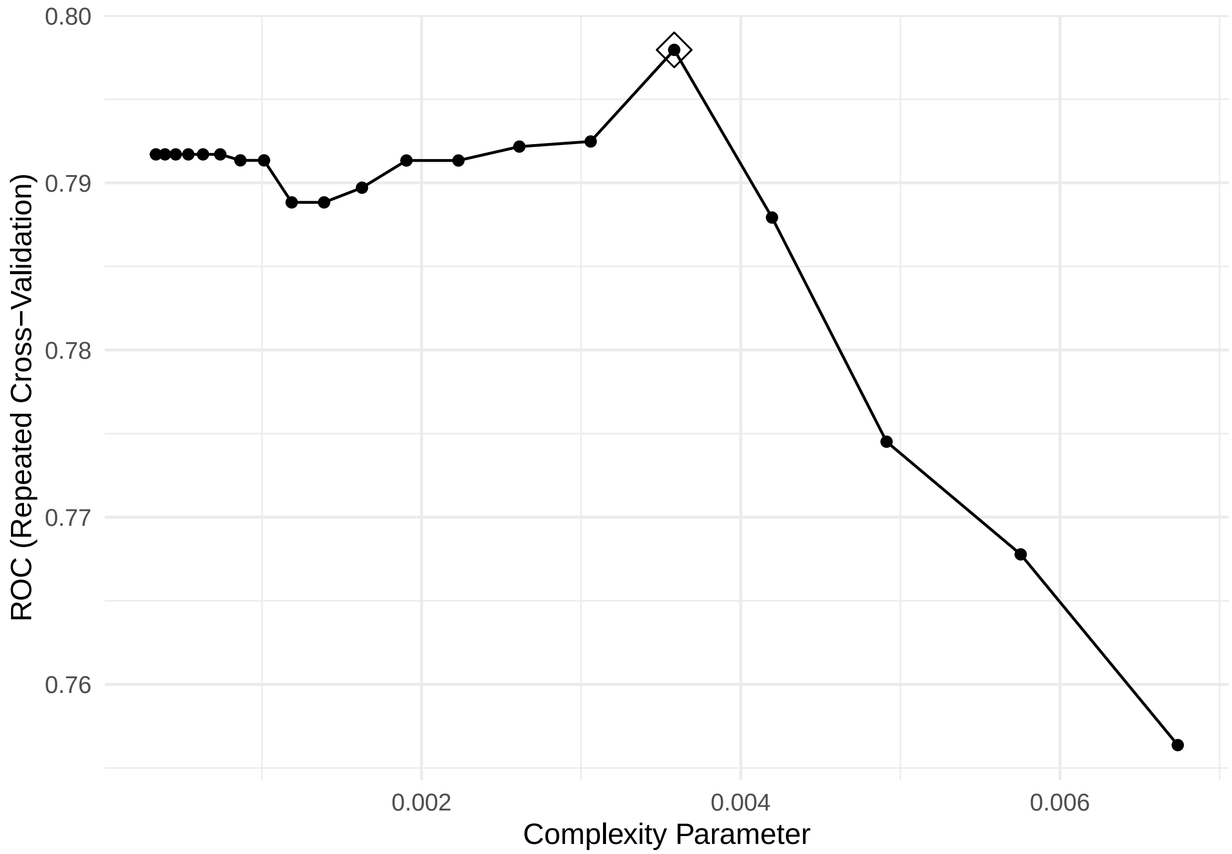
```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was
## not in the result set. ROC will be used instead.
```

```r
ggplot(knn.fit)
```

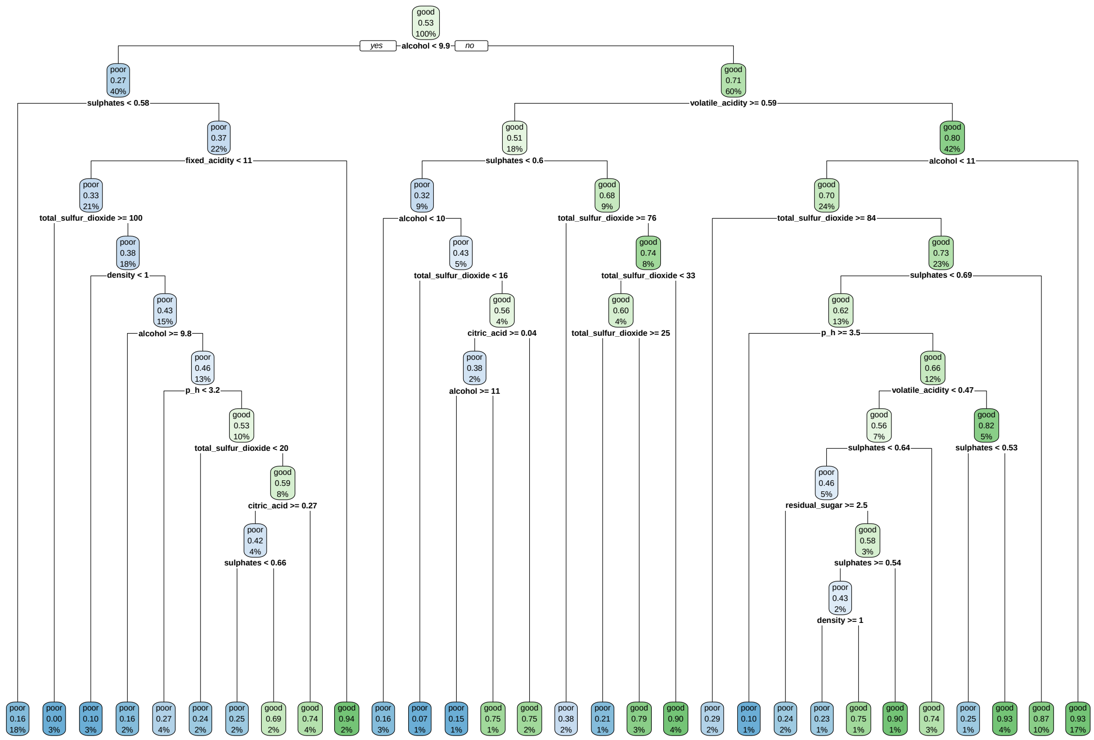


Classification tree

```r
set.seed(1)
rpart.fit <- train(quality~., df,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-8,-5, len = 20))),
                   trControl = ctrl,
                   metric = "ROC")
ggplot(rpart.fit, highlight = TRUE)
```

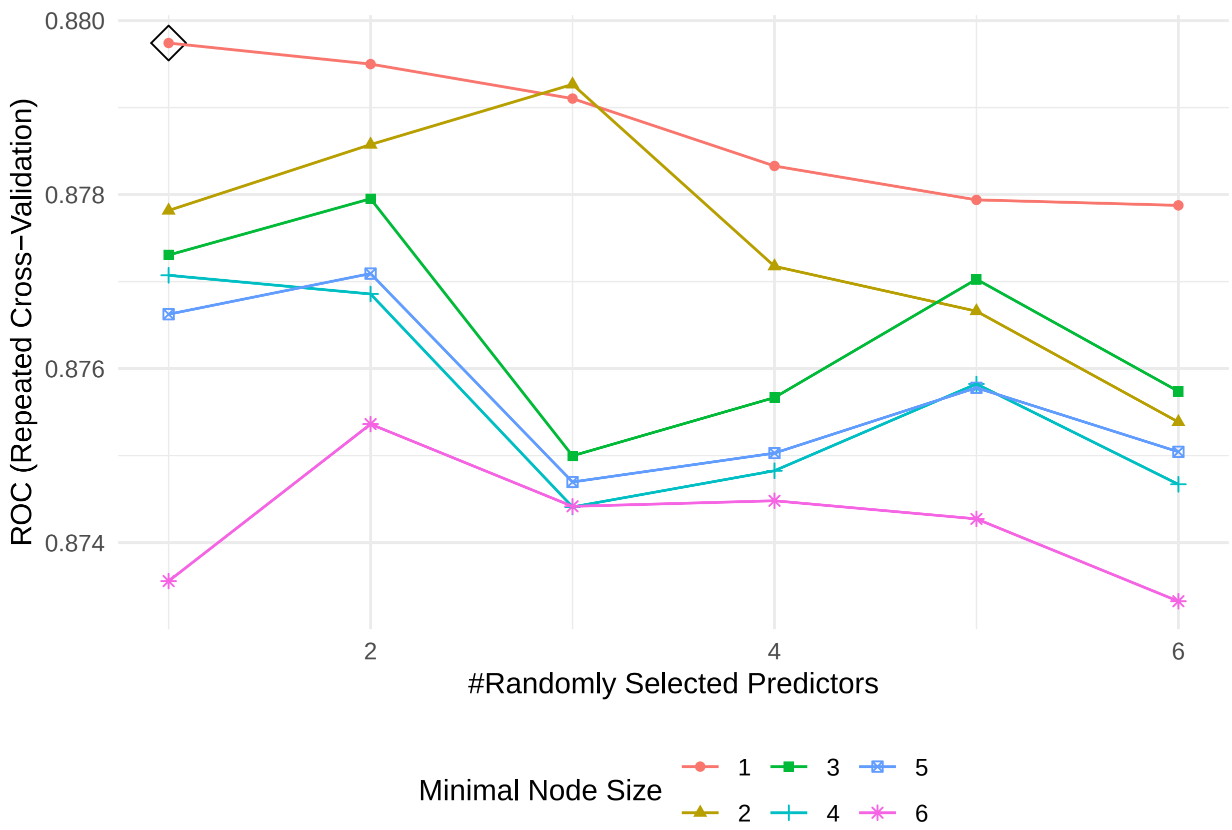



```r
rpart.plot(rpart.fit$finalModel)
```



Random forests

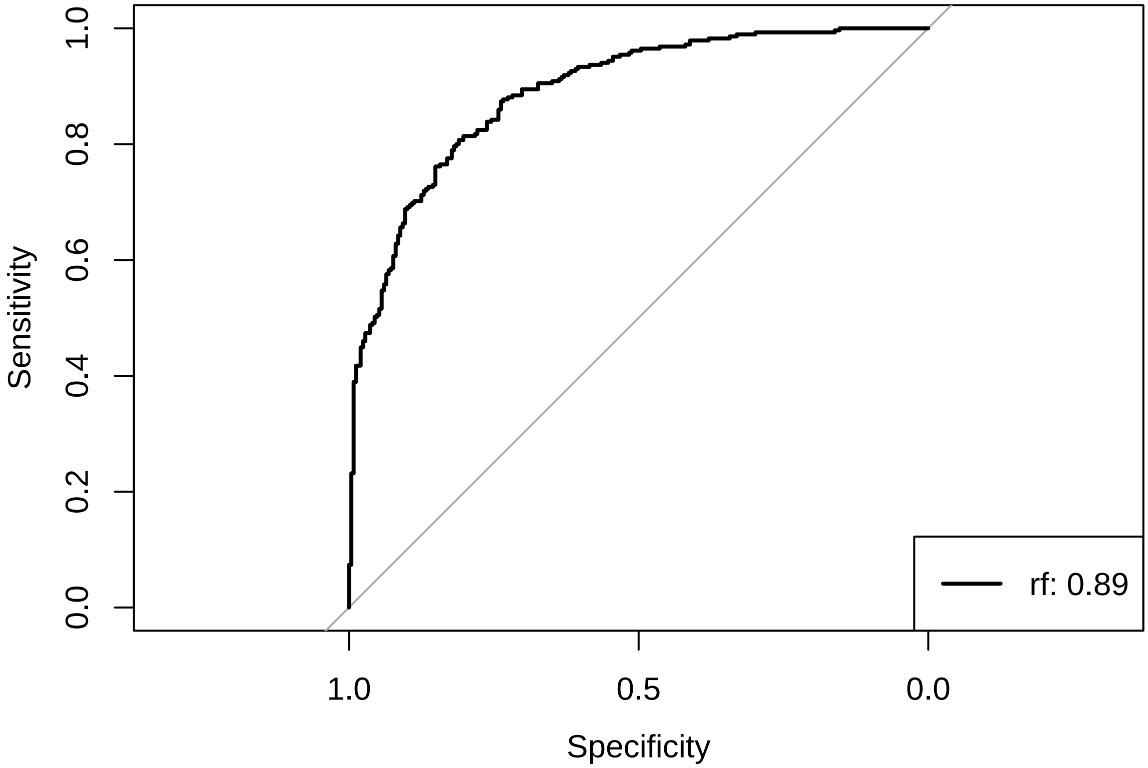
```
rf.grid <- expand.grid(mtry = 1:6,
                       splitrule = "gini",
                       min.node.size = 1:6)
set.seed(1)
rf.fit <- train(quality~., df,
                subset = rowTrain,
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",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

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




```r
rf.pred <- predict(rf.fit, newdata = df[-rowTrain,], type = "prob")[,1]
roc.rf <- roc(df$quality[-rowTrain], rf.pred)
```

```
## Setting levels: control = poor, case = good
## Setting direction: controls > cases
```

```r
plot(roc.rf)
auc <- roc.rf$auc[1]
modelNames <- "rf"
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:6, lwd = 2)
```



performance

```r
resamp <- resamples(list(rf = rf.fit,
                         knn = knn.fit,
                         lda = model.lda,
                         qda = model.qda,
                         rpart = rpart.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: rf, knn, lda, qda, rpart
## Number of resamples: 10
##
## ROC
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf    0.8234873 0.8462281 0.8820265 0.8797422 0.9019298 0.9364912    0
## knn   0.7123165 0.7652873 0.8052077 0.8099660 0.8519298 0.9117436    0
## lda   0.7372001 0.7773595 0.8102291 0.8115059 0.8524561 0.8729825    0
## qda   0.7207304 0.7642571 0.7865915 0.7834987 0.8044737 0.8494737    0
## rpart 0.7182241 0.7735383 0.7973648 0.7979520 0.8437397 0.8745614    0
##
## Sens
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
## rf    0.7142857 0.7437755 0.7700000 0.7802041 0.7950000 0.8979592    0
## knn   0.6326531 0.6633673 0.6869388 0.6997959 0.7000000 0.8979592    0
## lda   0.6734694 0.7236735 0.7500000 0.7480816 0.7600000 0.8571429    0
## qda   0.5800000 0.6450000 0.6667347 0.6655918 0.6984694 0.7142857    0
## rpart 0.6600000 0.6751020 0.7069388 0.7297551 0.7350000 0.9000000    0
##
## Spec
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf    0.7719298 0.8070175 0.8508772 0.8368421 0.8596491 0.8947368    0
## knn   0.6491228 0.7017544 0.7807018 0.7596491 0.8157895 0.8421053    0
## lda   0.5789474 0.7061404 0.7543860 0.7456140 0.8026316 0.8421053    0
## qda   0.6666667 0.7412281 0.7543860 0.7754386 0.8201754 0.8771930    0
## rpart 0.6140351 0.6622807 0.7368421 0.7385965 0.8070175 0.9122807    0
```