R Notebook

Michelle Cui and Sixian Ju

Contents

1. Logistic Regression **2**

train data . . . 2

errors for logistic regression . . . 3

2. LDA and QDA **3**

3. KNN for classification **4**

4. Classification and Receiver Operating Characteristics(ROC) curves and AUC **5**

5. tree **6**

decision tree . . . 6

prune tree . . . 7

errors for prune tree . . . 7

bagging . . . 8

errors for bagging . . . 9

random forest . . . 10

errors for random forest . . . 11

boosting . . . 11

errors for boosting . . . 12

```
library(readr)
library(tidyverse)
library(caret)
library(ggplot2)
library(pROC)
library(MASS)
library(leaps)
library(egg)
library(glmnet)
library(plotmo)
library(tree)
library(randomForest)
library(gbm)
```

```
wine <- read_delim("winequality-white.csv", delim = ";", escape_double = FALSE, trim_ws = TRUE)
colnames(wine) <- c("fixed_ac", "volatile_ac", "citric_ac", "residual_sugar", "chloride", "free_so2", "
wine.quality <- wine %>%
  mutate(excellent = case_when(quality <= 6 ~ FALSE,
                           quality > 6 ~ TRUE))
wine.quality$excellent <- as.factor(wine.quality$excellent)
#split test and training
wine.q <- wine.quality %>%
  mutate(quality = ifelse(wine$quality <= 5, "poor",
                           ifelse(wine$quality <= 7, "average", "excellent" )))

set.seed(0)
n_all <- nrow(wine)
tr_ind <- sample(n_all, round(n_all/2))
wine_train <- wine.q[tr_ind, ]
wine_test <- wine.q[-tr_ind, ]
colnames(wine_test)[13] <- "excellent"
colnames(wine_train)[13] <- "excellent"
fit_std <- preProcess(wine_train, method = "scale")
wine_train <- predict(fit_std, newdata = wine_train )
wine_test <- predict(fit_std, newdata = wine_test)
wine_train$quality <- as.factor(wine_train$quality)
wine_test$quality <- as.factor(wine_test$quality)
##scale
wine_sc <- rbind(wine_train, wine_test)
```

1. Logistic Regression

train data

```
glmod.train <- glm(excellent ~.-quality, wine_train, family = "binomial")
summary(glmod.train)
```

```
##
## Call:
## glm(formula = excellent ~ . - quality, family = "binomial", data = wine_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0221  -0.6685  -0.4046  -0.1503   2.9342
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    689.04626  135.72505   5.077 3.84e-07 ***
## fixed_ac         0.53957    0.11210   4.813 1.49e-06 ***
## volatile_ac     -0.48187    0.07539  -6.392 1.64e-10 ***
## citric_ac       -0.13476    0.06970  -1.933   0.0532 .
## residual_sugar   1.68357    0.26388   6.380 1.77e-10 ***
## chloride        -0.29461    0.12699  -2.320   0.0203 *
## free_so2         0.08868    0.07636   1.161   0.2455
```

```
## total_so2        0.05206    0.09214   0.565   0.5720
## density         -2.15925    0.41607  -5.190 2.11e-07 ***
## ph               0.57710    0.09590   6.018 1.77e-09 ***
## sulphates        0.25770    0.05612   4.592 4.39e-06 ***
## alcohol          0.17754    0.19789   0.897   0.3697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2519.2  on 2448  degrees of freedom
## Residual deviance: 2019.7  on 2437  degrees of freedom
## AIC: 2043.7
##
## Number of Fisher Scoring iterations: 6
```

errors for logistic regression

```
pred_exl_train <- predict(glmod.train, newdata = wine_train, type = "response")
exl_train_label <- ifelse(pred_exl_train > 0.5, TRUE, FALSE)
table(predict = exl_train_label, train = wine_train$excellent)
```

```
##        train
## predict FALSE TRUE
##   FALSE  1844  363
##   TRUE     90  152
```

```
train_error_exl <- mean(exl_train_label != wine_train$excellent)
pred_exl_test <- predict(glmod.train, newdata = wine_test, type = "response")
exl_test_label <- ifelse(pred_exl_test > 0.5, TRUE, FALSE)
table(predict = exl_test_label, test = wine_test$excellent)
```

```
##        test
## predict FALSE TRUE
##   FALSE  1785  400
##   TRUE    119  145
```

```
test_error_exl <- mean(exl_test_label != wine_test$excellent)
```

Our train error for logistic regression is 0.1849735. And test error is 0.2119232.

2. LDA and QDA

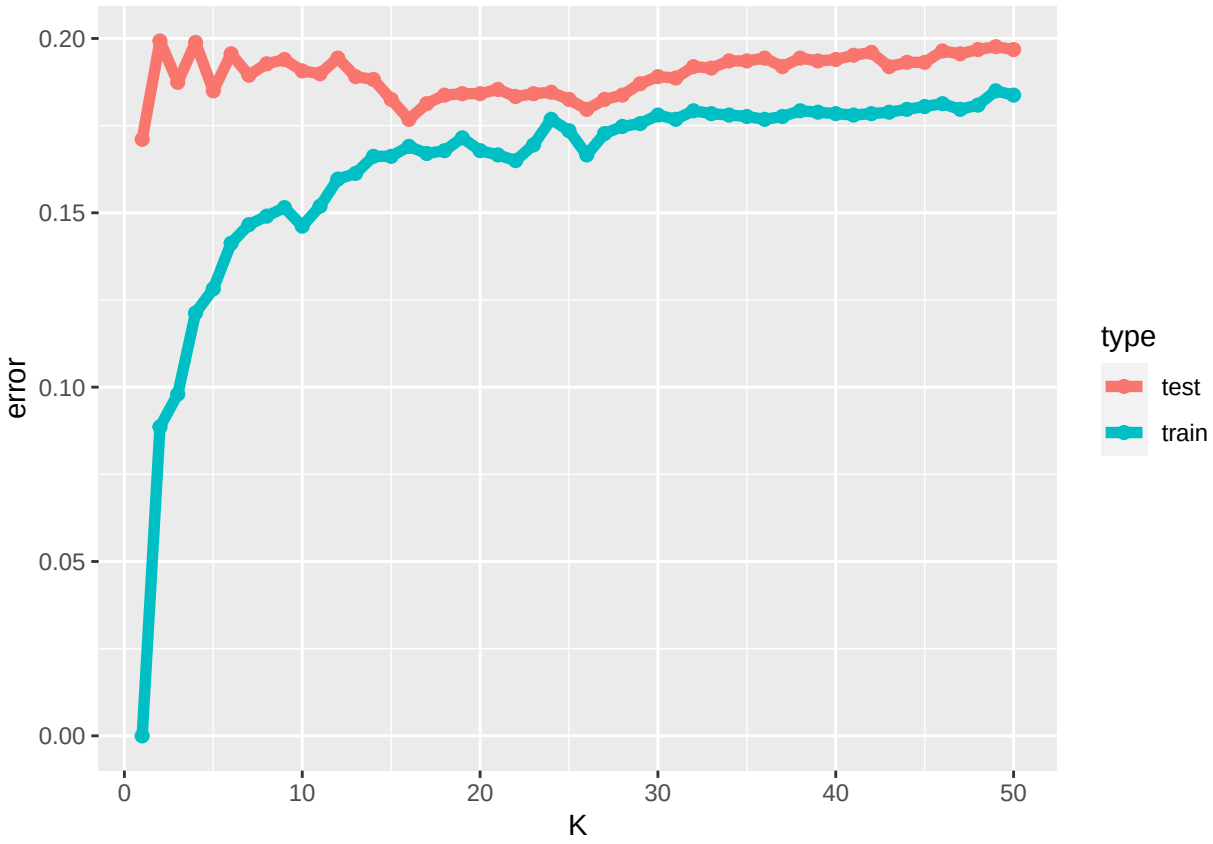
```
## LDA
fit_lda <- lda(excellent ~. -quality, wine_train)
### train error
lda.train.pred <- predict(fit_lda, newdata = wine_train)
```

```r
lda.train.pred.class <- lda.train.pred$class
lda.train.error <- mean(lda.train.pred.class != wine_train$excellent)
### test error
lda.test.pred <- predict(fit_lda, newdata = wine_test)
lda.test.pred.class <- lda.test.pred$class
lda.test.error <- mean(lda.test.pred.class != wine_test$excellent)

## QDA
fit_qda <- qda(excellent ~. -quality, wine_train)
### training error
qda.train.pred <- predict(fit_qda, newdata = wine_train)
qda.train.pred.class <- qda.train.pred$class
qda.train.error <- mean(qda.train.pred.class != wine_train$excellent)
### testing error
qda.test.pred <- predict(fit_qda, newdata = wine_test)
qda.test.pred.class <- qda.test.pred$class
qda.test.error <- mean(qda.test.pred.class != wine_test$excellent)
```

type	train error	test error
LDA	0.1886484	0.2078399
QDA	0.2670478	0.2711311

3. KNN for classification

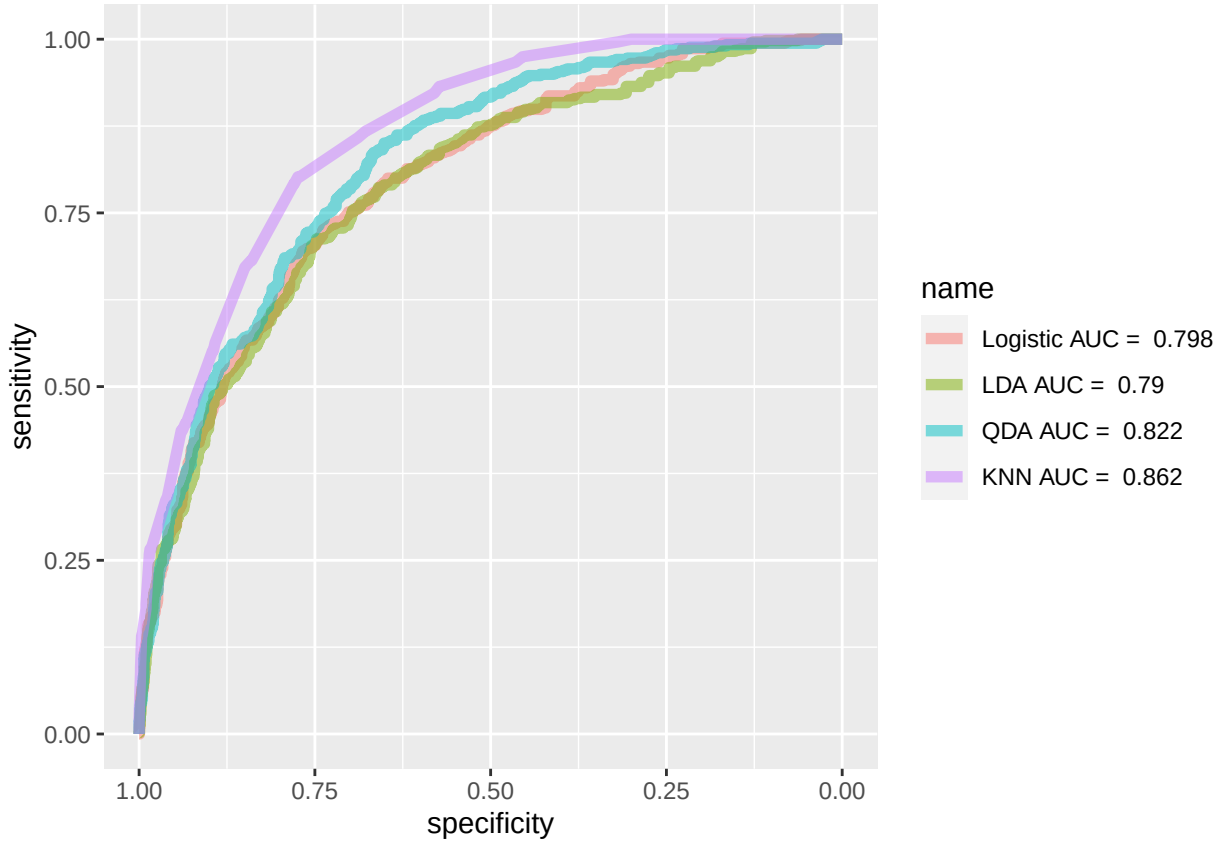
```r
k_seq <- c(1:50)
train_error_seq <- test_error_seq <- NULL
for (i in seq_along(k_seq)) {
  k <- k_seq[i]
  fit_knn <- knn3(excellent ~. -quality, wine_train, k = k)
  pred_knn_train <- predict(fit_knn, newdata = wine_train, type = "class")
  train_error_seq[i] <- mean(pred_knn_train != wine_train$excellent)
  test <- predict(fit_knn, newdata = wine_test, type = "class")
  test_error_seq[i] <- mean(test != wine_test$excellent)
}

knn_df <- rbind(data.frame(K = k_seq, error = train_error_seq, type = "train"),
                data.frame(K = k_seq, error = test_error_seq, type = "test"))
ggplot(knn_df, mapping = aes(x = K, y = error, color = type)) +
  geom_point(size = 2) +
  geom_line(size = 2)
```



4. Classification and Receiver Operating Characteristics(ROC) curves and AUC

```
### auc_lda
pred_lda <- predict(fit_lda)$posterior[,2]
roc_lda <- roc(wine_train$excellent, pred_lda)
auc_lda <- auc(roc_lda)
### auc_qda
pred_qda <- predict(fit_qda)$posterior[,2]
roc_qda <- roc(wine_train$excellent, pred_qda)
auc_qda <- auc(roc_qda)
### logistic regression
roc_logi <- roc(wine_train$excellent, pred_exl_train)
auc_logi <- auc(roc_logi)
### knn
fit_knn <- knn3(excellent ~. -quality, wine_train, k = 16, prob = TRUE)
pred_knn <- predict(fit_knn, newdata = wine_train, type = "prob")
roc_knn <- roc(wine_train$excellent, pred_knn[,2])
auc_knn <- auc(roc_knn)

rocobj <- list(Logistic = roc_logi, LDA = roc_lda, QDA = roc_qda, KNN = roc_knn)
methods_auc <- paste(c("Logistic", "LDA", "QDA", "KNN"), "AUC = ",
                     round(c(auc_logi, auc_lda, auc_qda, auc_knn), 3))
```

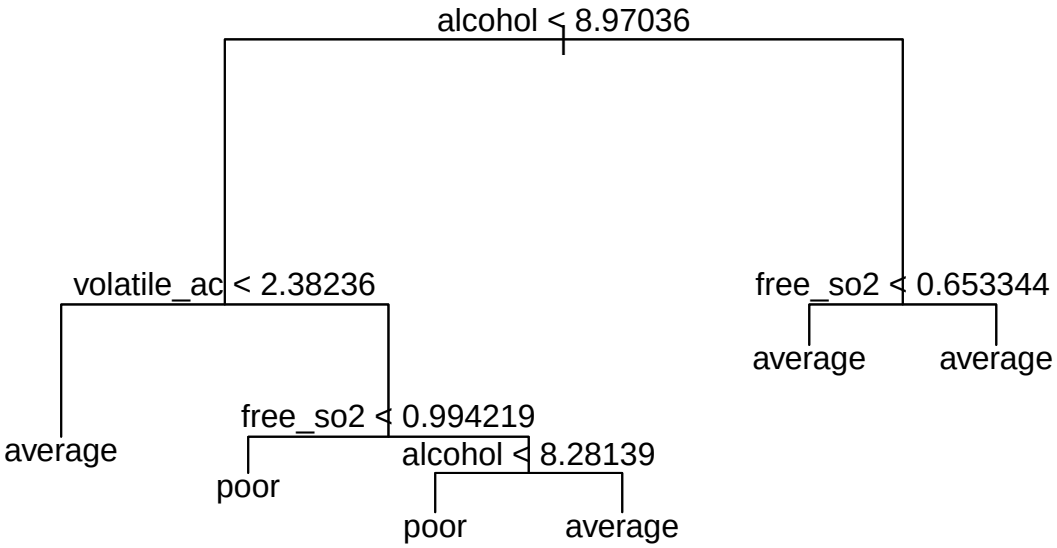
```
ggroc(rocobj, size = 2, alpha = 0.5) +
  scale_color_discrete(labels = methods_auc)
```



5. tree

decision tree

```
fit.tree <- tree(quality ~ fixed_ac + volatile_ac + citric_ac + residual_sugar + chloride + free_so2 + 
set.seed(0)
cv.type <- cv.tree(fit.tree)

plot(fit.tree)
text(fit.tree)
```

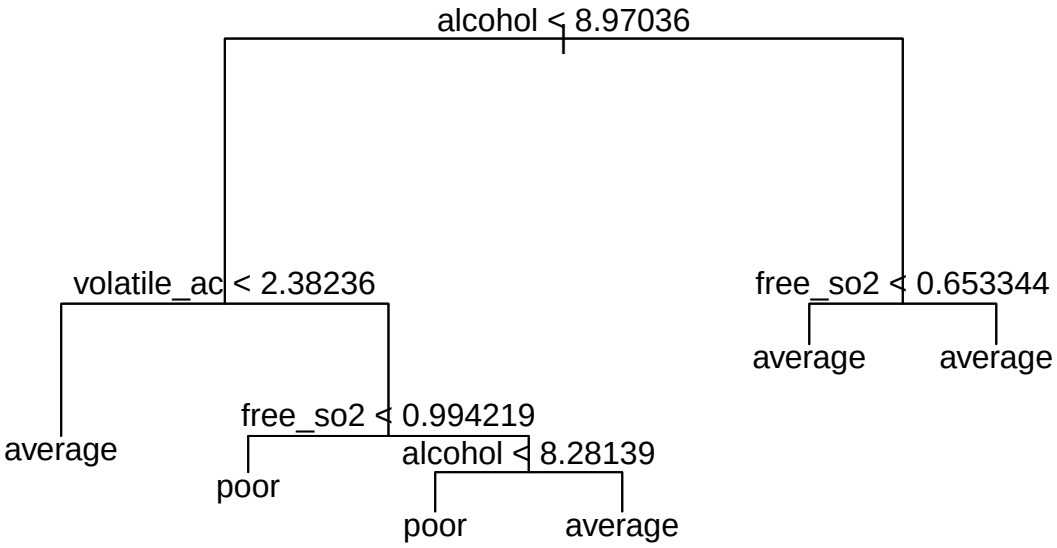


prune tree

```
(bestsize_tree_type <- cv.type$size[which.min(cv.type$dev)])
```

```
## [1] 6
```

```
prune_type <- prune.tree(fit.tree, best = bestsize_tree_type)

plot(prune_type)
text(prune_type)
```



errors for prune tree

```
##prune
### training error
pred_type_train <- predict(prune_type, newdata = wine_train, type = "class")
prune.train.error <- mean(pred_type_train != wine_train$quality)
### test error
pred_type_test <- predict(prune_type, newdata = wine_test, type = "class")
prune.test.error <- mean(pred_type_test != wine_test$quality)
```

bagging

```
p <- ncol(wine.quality) - 1
bag_fit <- randomForest(quality ~ fixed_ac + volatile_ac + citric_ac + residual_sugar + chloride + free
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
bag_fit
```

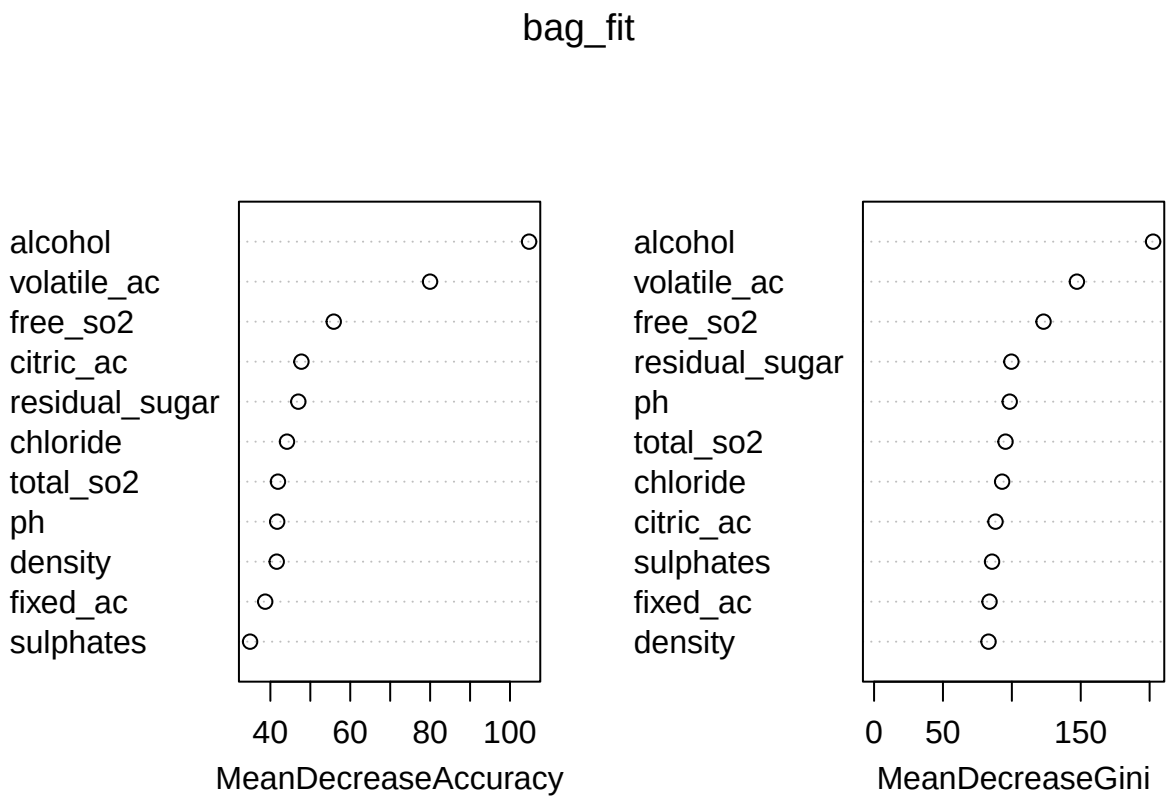
```
##
## Call:
##  randomForest(formula = quality ~ fixed_ac + volatile_ac + citric_ac +      residual_sugar + chlorid
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##         OOB estimate of  error rate: 22.01%
## Confusion matrix:
##           average excellent poor class.error
## average      1336         4  197   0.1307742
## excellent      64        17    0   0.7901235
## poor          273         1  557   0.3297232
```

```
importance(bag_fit)
```

```
##                 average excellent     poor MeanDecreaseAccuracy
## fixed_ac       28.69339 14.496416 26.17894             38.70484
## volatile_ac    55.56244 32.050309 55.72244             79.98069
## citric_ac      36.35715 15.087495 27.32360             47.77326
## residual_sugar 36.46697 10.344337 27.66559             46.99954
## chloride       29.21661 12.678412 28.29627             44.15617
## free_so2       40.21735 14.966357 35.28244             55.86153
## total_so2      24.92937 18.244959 25.41028             41.89533
## density        31.29106  8.517527 15.75913             41.58878
## ph             25.79126 15.912904 32.65082             41.70324
## sulphates      23.59093 17.925353 25.19618             34.91962
## alcohol        55.74903 44.695047 85.76612            104.78185
##                MeanDecreaseGini
## fixed_ac               83.80294
## volatile_ac           147.21572
```

```
## citric_ac              88.12772
## residual_sugar         99.66090
## chloride               92.95145
## free_so2              122.99952
## total_so2              95.40088
## density                83.06712
## ph                     98.43153
## sulphates              85.63093
## alcohol               202.52648
```

```
varImpPlot(bag_fit)
```



errors for bagging

```
##bagging
### training error
train.bag.type <- predict(bag_fit, newdata = wine_train, type = "class")
bag.train.error <- mean(train.bag.type != wine_train$quality)
### test error
test.bag.type <- predict(bag_fit, newdata = wine_test)
bag.test.error <- mean(test.bag.type != wine_test$quality)
```

random forest

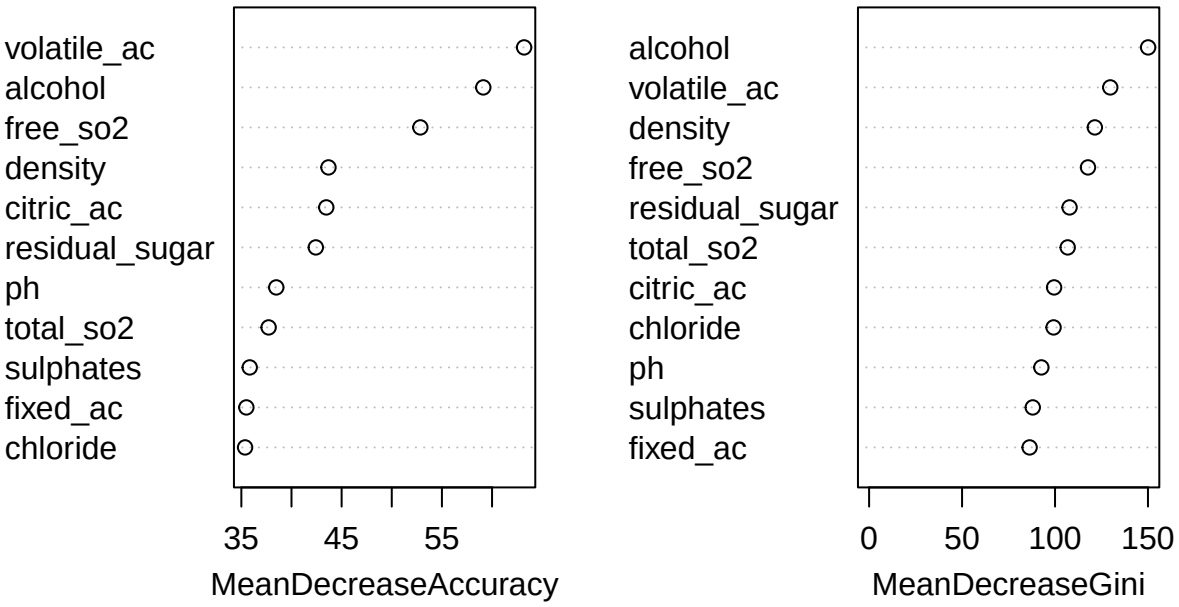
```
set.seed(0)
rf.type <- randomForest(quality ~ fixed_ac + volatile_ac + citric_ac + residual_sugar + chloride + free_
rf.type
```

```
##
## Call:
##  randomForest(formula = quality ~ fixed_ac + volatile_ac + citric_ac +      residual_sugar + chloride
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 21.31%
## Confusion matrix:
##           average excellent poor class.error
## average      1352         2  183   0.1203643
## excellent      64        17    0   0.7901235
## poor          273         0  558   0.3285199
```

```
importance(rf.type)
```

```
##                 average excellent     poor MeanDecreaseAccuracy
## fixed_ac       23.50271  17.17611 24.11913             35.50043
## volatile_ac    44.96321  26.71236 48.53512             63.19765
## citric_ac      35.36026  16.57741 27.56267             43.47083
## residual_sugar 31.75171  13.21749 27.87131             42.42474
## chloride       26.44891  17.97996 27.39499             35.37360
## free_so2       34.96274  14.14561 36.87276             52.85003
## total_so2      24.61444  18.49637 26.33265             37.72086
## density        30.78403  20.78773 24.96956             43.68921
## ph             25.17415  16.83690 28.87245             38.48407
## sulphates      24.21046  18.55926 25.37707             35.83700
## alcohol        36.04488  33.71202 55.45350             59.12387
##                MeanDecreaseGini
## fixed_ac               86.34029
## volatile_ac           129.71743
## citric_ac              99.50221
## residual_sugar        107.81592
## chloride               99.22287
## free_so2              117.72976
## total_so2             106.81997
## density               121.41686
## ph                     92.64881
## sulphates              88.02398
## alcohol               150.08665
```

```
varImpPlot(rf.type)
```

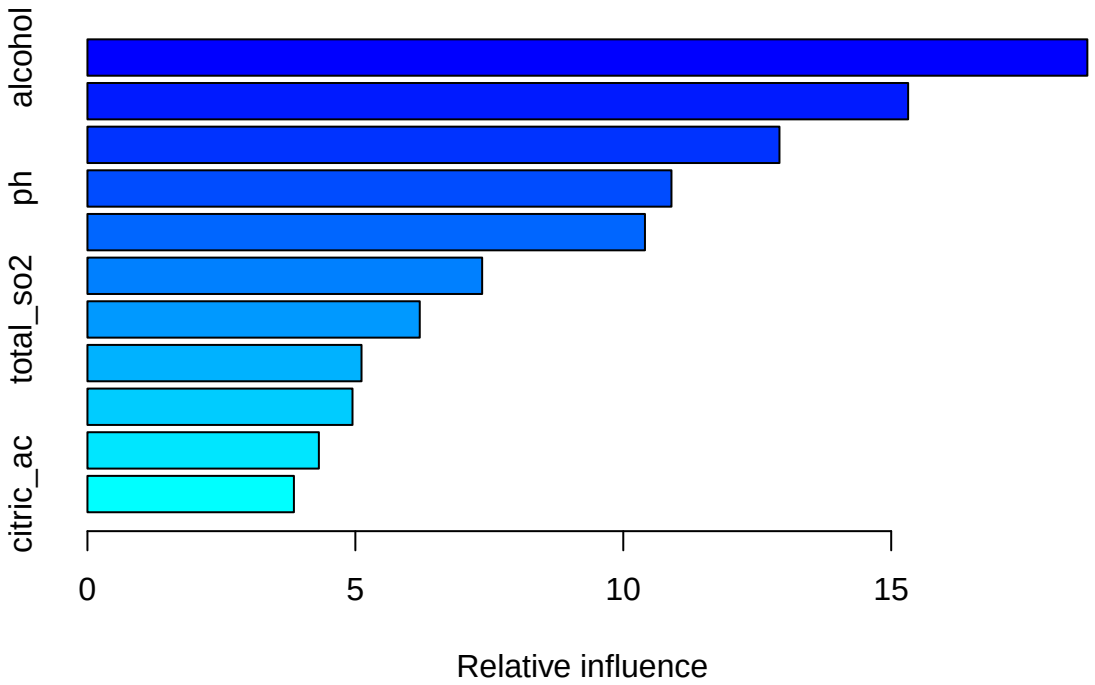


errors for random forest

```
#random forest
### training error
train.rf.type <- predict(rf.type, newdata = wine_train)
rf.train.error <- mean(train.rf.type != wine_train$quality)
### test error
test.rf.type <- predict(rf.type, newdata = wine_test)
rf.test.error <- mean(test.rf.type != wine_test$quality)
```

boosting

```
wine_tr <- wine_train %>%
  mutate(excellent = ifelse(wine_train$excellent == "TRUE", 1, 0))
set.seed(0)
boost.type <- gbm(excellent ~ fixed_ac + volatile_ac + citric_ac + residual_sugar + chloride + free_so2
                  distribution = "bernoulli", n.trees = 5000,
                  interaction.depth = 1, cv.folds = 5, shrinkage = 0.2)
best_n_type <- which.min(boost.type$cv.error)
summary(boost.type)
```




```
##                          var   rel.inf
## alcohol              alcohol 18.661096
## density              density 15.316223
## sulphates          sulphates 12.915238
## ph                        ph 10.899315
## chloride            chloride 10.404572
## volatile_ac      volatile_ac  7.366672
## total_so2          total_so2  6.201768
## residual_sugar residual_sugar  5.115241
## fixed_ac            fixed_ac  4.946826
## free_so2            free_so2  4.319353
## citric_ac          citric_ac  3.853695
```

errors for boosting

type	train error	test error
logistic regression	0.1849735	0.2119232
knn(16)	0.1690486	0.1768069
LDA	0.1886484	0.2078399
QDA	0.2670478	0.2711311
prune	0.2866476	0.2780727
bagging	0	0.2127399
random forest	0	0.2033483
boosting	0.1465904	0.1935484