R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the Run button within the chunk or by placing your cursor inside it and pressing Cmd+Shift+Enter.

— Random Forests —

— Data Preparation —

Let's first try random forest regression:

```
# Import random forest library
library(randomForest)

# Preprocessing steps

# Read read and delete error observer
red = read.csv("winequality-red.csv", header = TRUE, sep=";")
red = na.omit(red)
red$X = NULL
red$fixed.acidity = NULL

white = read.csv("winequality-white.csv", header = TRUE, sep=";")
white = na.omit(white)
white$X = NULL
white$fixed.acidity = NULL

# Stratified partition 75 percent distribution (same # and mix as 1)
red_idx = sample(1:NROW(red), .75)
white_idx = sample(1:NROW(white), .75)

# Split into training and test sets (80-20 split)
red_train = red[red_idx,]
red_test = red[-red_idx,]

# White dataset
train_white_idx = sample(1:NROW(white)*.8 * 0 + 1:NROW(white) * 1)
test_white_idx = -train_white_idx
train_white_matrix = white[train_white_idx,]
test_white_matrix = white[-train_white_idx,]

train_white_x = train_white_matrix[,1:10]
train_white_y = train_white_matrix[,11]
test_white_x = test_white_matrix[,1:10]
test_white_y = test_white_matrix[,11]

train_red_idx = sample(1:NROW(red)*.8 * 0 + 1:NROW(red) * 1)
test_red_idx = -train_red_idx
train_red_matrix = red[train_red_idx,]
test_red_matrix = red[-train_red_idx,]

train_red_x = train_red_matrix[,1:10]
train_red_y = train_red_matrix[,11]
test_red_x = test_red_matrix[,1:10]
test_red_y = test_red_matrix[,11]
```

```
dim(white)
```

```
[1] 4898   12
```

```
dim(red)
```

```
[1] 1599   12
```

— Method 1: Random Forest Regressor —

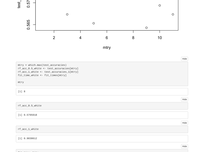
White Dataset:

```
train_percentage = length, 10)
test_percentage = length, 10)
fit_time = rbind, 10)
pred_time = rbind, 10)

for(i in 2:11)
{
  start = proc.time()
  fit = randomForest(x=train_white_x, y=train_white_y, importance=TRUE) # default: ntree=500
  fit_time[i] = proc.time() - start
  train_percentage[i] = mean(abs(predict(fit, train_white) - train_white_y) <= .5)
  test_percentage[i] = mean(abs(predict(fit, test_white) - test_white_y) <= .5)
}
```

```
number of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a multiple of
replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a
multiple of replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to
replace is not a multiple of replacement Singlethread of trees to replace is not a multiple of replacement
Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a
multiple of replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to
replace is not a multiple of replacement Singlethread
```

```
fit_time = pred_time = NULL
for(i in 2:11)
{
  fit_time = rbind(fit_time, fit[i])
  pred_time = rbind(pred_time, pred[i])
}
```

```
plot(fit_time, xlab='ntry', main='Fit Times for RF Regression for White Wines')
```




```
plot(test_percentage, xlab='ntry', main='Test Accuracy for RF Regression for White Wines')
```

```
ntry = which.max(test_percentage)
rf_acc_0.5 = test_percentage[ntry]
rf_acc_0.5_time = fit_time[ntry]
fit_time_white = fit_time[ntry]

ntry
```

```
[1] 9
```

```
rf_acc_0.5_tw
```

```
[1] 0.9347175
```

```
rf_acc_0.5_twt
```

```
[1] 0.9387273
```

```
fit_time_twt
```

```
[1] 1.403
```

Weighted Average Test performances for both red and white dataset:

Within absolute margin 0.5

```
(rf_acc_0.5_twt * length(test_red) + rf_acc_0.5_tw * length(test_white)) / (length(test_red) + length(test_white))
```

```
[1] 0.9338708
```

Within absolute margin 1

```
(rf_acc_1_twt * length(test_red) + rf_acc_1_tw * length(test_white)) / (length(test_red) + length(test_white))
```

```
[1] 0.9944154
```

— Data Preparation —

We will now do the classification task. The only difference is the code below is that the response variable (quality) is converted into a factor, which is important for random forest classification.

```
# Preprocessing steps

# Read read and delete error observer
red = read.csv("winequality-red.csv", header = TRUE, sep=";")
red = na.omit(red)
red$quality = factor(red$quality) # NEW: Convert into factor for classification tasks!
red$X = NULL
red$fixed.acidity = NULL

white = read.csv("winequality-white.csv", header = TRUE, sep=";")
white = na.omit(white)
white$quality = factor(white$quality) # NEW: Convert into factor for classification tasks!
white$X = NULL
white$fixed.acidity = NULL

# Stratified partition 75 percent distribution (same # and mix as 1)
red_idx = sample(1:NROW(red), .75)
white_idx = sample(1:NROW(white), .75)

# Split into training and test sets (80-20 split)
red_train = red[red_idx,]
red_test = red[-red_idx,]

# White dataset
train_white_idx = sample(1:NROW(white)*.8 * 0 + 1:NROW(white) * 1)
test_white_idx = -train_white_idx
train_white_matrix = white[train_white_idx,]
test_white_matrix = white[-train_white_idx,]

train_white_x = train_white_matrix[,1:10]
train_white_y = train_white_matrix[,11]
test_white_x = test_white_matrix[,1:10]
test_white_y = test_white_matrix[,11]

train_red_idx = sample(1:NROW(red)*.8 * 0 + 1:NROW(red) * 1)
test_red_idx = -train_red_idx
train_red_matrix = red[train_red_idx,]
test_red_matrix = red[-train_red_idx,]

train_red_x = train_red_matrix[,1:10]
train_red_y = train_red_matrix[,11]
test_red_x = test_red_matrix[,1:10]
test_red_y = test_red_matrix[,11]
```

— Method 2: Random Forest Classifier —

White Dataset:

```
train_percentage = length, 10)
test_percentage = length, 10)
pred = data.frame()

for(i in 2:11)
{
  start = proc.time()
  fit = randomForest(x=train_white_x, y=train_white_y, importance=TRUE) # default: ntree=500
  fit_time[i] = proc.time() - start
  train_percentage[i] = mean(predict(fit, train_white) == train_white_y)
  test_percentage[i] = mean(predict(fit, test_white) == test_white_y)
}
```

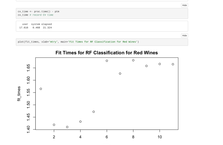
```
number of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a multiple of
replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to replace is not
a multiple of replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to
replace is not a multiple of replacement Singlethread of trees to replace is not a multiple of replacement
Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a
multiple of replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to
replace is not a multiple of replacement Singlethread
```

```
fit_time = pred = NULL
for(i in 2:11)
{
  fit_time = rbind(fit_time, fit[i])
  pred = rbind(pred, pred[i])
}
```

```
  user  system elapsed
  0.00   0.00   0.00
```

```
plot(fit_time, xlab='ntry', main='Fit Times for RF Classification for White Wines')
```

```
plot(test_percentage, xlab='ntry', main='Test Accuracy for RF Classification for White Wines')
```




```
ntry = which.max(test_percentage)
rf_acc = test_percentage[ntry]
fit_time_white = fit_time[ntry]

ntry
```

```
[1] 9
```

```
rf_acc_white
```

```
[1] 0.9428571
```

```
fit_time_white
```

```
[1] 0.767
```

Red Dataset:

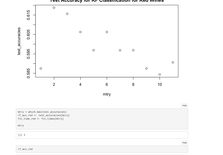
```
train_percentage = length, 10)
test_percentage = length, 10)
fit_time = rbind, 10)
pred = data.frame()

for(i in 2:11)
{
  start = proc.time()
  fit = randomForest(x=train_red_x, y=train_red_y, importance=TRUE) # default: ntree=500
  fit_time[i] = proc.time() - start
  train_percentage[i] = mean(predict(fit, train_red) == train_red_y)
  test_percentage[i] = mean(predict(fit, test_red) == test_red_y)
}
```

```
number of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a multiple of
replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to replace is not
a multiple of replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to
replace is not a multiple of replacement Singlethread of trees to replace is not a multiple of replacement
Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to replace is not a
multiple of replacement Singlethread of trees to replace is not a multiple of replacement Singlethread of trees to
replace is not a multiple of replacement Singlethread
```

```
fit_time = pred = NULL
for(i in 2:11)
{
  fit_time = rbind(fit_time, fit[i])
  pred = rbind(pred, pred[i])
}
```

```
  user  system elapsed
  0.00   0.00   0.00
```

```
plot(fit_time, xlab='ntry', main='Fit Times for RF Classification for Red Wines')
```




```
plot(test_percentage, xlab='ntry', main='Test Accuracy for RF Classification for Red Wines')
```

```
ntry = which.max(test_percentage)
rf_acc = test_percentage[ntry]
fit_time_red = fit_time[ntry]

ntry
```

```
[1] 2
```

```
rf_acc_red
```

```
[1] 0.6125
```

```
fit_time_red
```

```
[1] 0.018
```

Weighted Average Test performances for both red and white dataset:

```
(rf_acc_twt * length(test_red) + rf_acc_white * length(test_white)) / (length(test_red) + length(test_white))
```

```
[1] 0.9331006
```