Homework5_RF_R

Odo Luo

April 11, 2021

Homework 5: Random Forest

Use both R (randomForest package) and Python (sklearn module) to answer the following questions: 1. Use the provided iris data set with the four features: a. Sepal.Length b. Sepal.Width c. Petal.Length d. Petal.Width and only two target Species setosa and versicolor. 2. Create train and test data sets using a fixed (stratified) split of 80-20. 3. Find an appropriate number of trees by testing from 100 to 1000 in steps of 100 using all provided features and (stratified) 10-fold cross validation. 4. Plot the resulting errors for the different numbers of trees, interpret the results, and choose an appropriate number of trees for your random forest (this might not be the model with the lowest error, e. g. when the error reaches a plateau). Train the final random forest on the whole training set. 5. Use the final model to evaluate test loss. Plot the confusion matrix and ROC curve and interpret them. 6. Discuss which actions you could take and why if your model does not perform well enough.

Imports

```r
library(dplyr)
library(caret)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.0.5
```

Preprocessing

1 Load Data

```r
iris <- read.csv("iris.csv",header = TRUE) %>%
  as_tibble %>%
  filter(Species== "setosa" | Species == "versicolor" ) %>%
  mutate(id=row_number())

head(iris)
```

```
## # A tibble: 6 x 6
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species    id
##          <dbl>       <dbl>        <dbl>       <dbl> <chr>   <int>
## 1          5.1         3.5          1.4         0.2 setosa      1
## 2          4.9         3            1.4         0.2 setosa      2
## 3          4.7         3.2          1.3         0.2 setosa      3
## 4          4.6         3.1          1.5         0.2 setosa      4
## 5          5           3.6          1.4         0.2 setosa      5
## 6          5.4         3.9          1.7         0.4 setosa      6
```

2 Split Data

```
train <- iris %>%  sample_frac(.80)
test <-  anti_join(iris, train,'id')
train <- train %>% select(-id)
test <- test %>% select(-id)
```

Tuning

3 Finding optimal number of trees

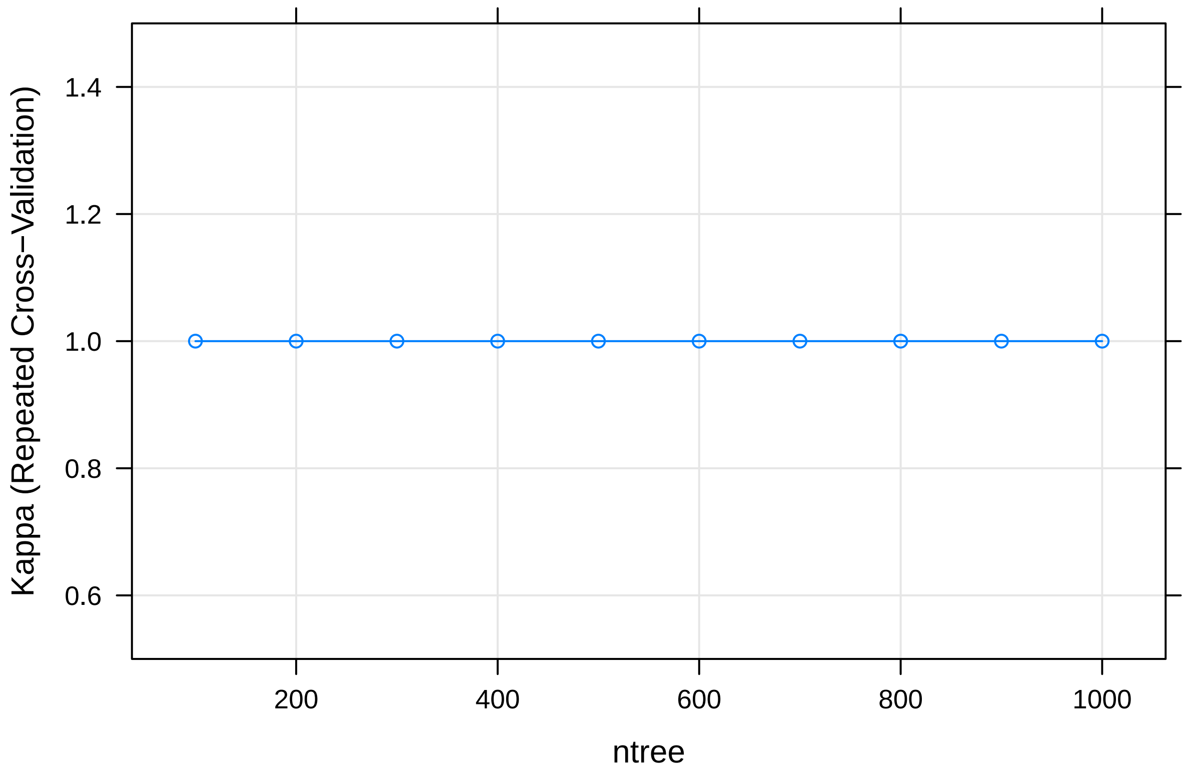
Code taken from: https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

```
customRF <- list(type = "Classification", library = "randomForest", loop = NULL)
customRF$parameters <- data.frame(parameter = c("mtry", "ntree"), class = rep("numeric", 2), label = c(
customRF$grid <- function(x, y, len = NULL, search = "grid") {}
customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {
  randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)
}
customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
   predict(modelFit, newdata)
customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
   predict(modelFit, newdata, type = "prob")
customRF$sort <- function(x) x[order(x[,1]),]
customRF$levels <- function(x) x$classes
```

```
control <- trainControl(method="repeatedcv", number=10, repeats=3,allowParallel = TRUE)
tunegrid <- expand.grid(.mtry=c(4), .ntree=seq(from = 100, to = 1000, by = 100))
set.seed(300)
custom <- train(Species~., data=train, method=customRF, metric="Kappa", tuneGrid=tunegrid, trControl=co
```

4 Plotting results

```
plot(custom)
```

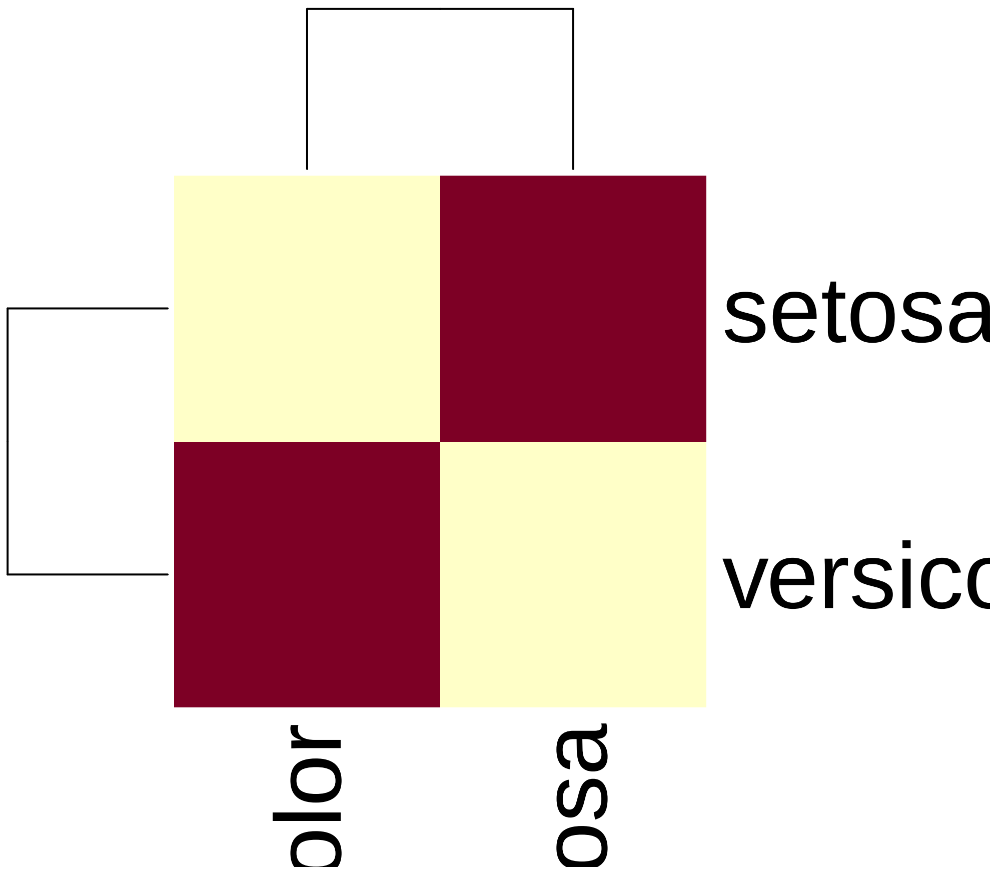


Final Model

5 Confusion Matrix and ROC

```
library(randomForest)
set.seed(300)
train$Species <- as.factor(train$Species)
test$Species <- as.factor(test$Species)
rf <- randomForest(Species ~ ., data = train, ntree=100)
pred<-predict(rf,test[-5])
```

```
cf <- confusionMatrix(pred,test$Species)
heatmap(cf$table)
```



Confusion Matrix

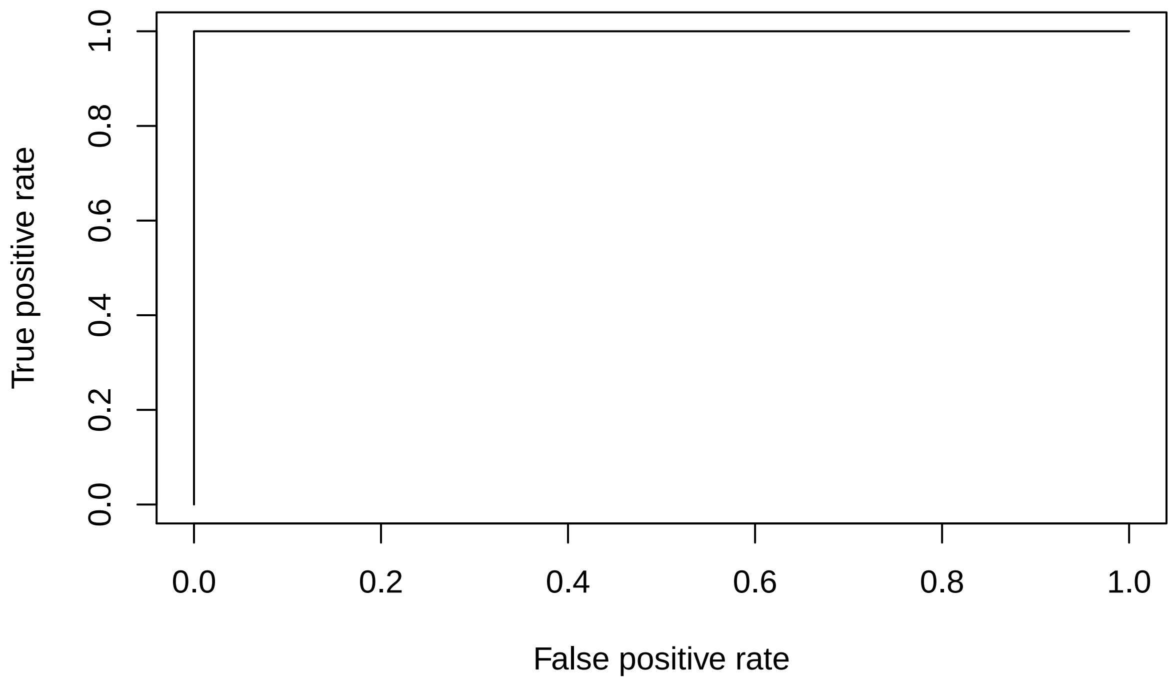
```
cf
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor
##   setosa         12          0
##   versicolor      0          8
##
##                Accuracy : 1
##                  95% CI : (0.8316, 1)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 3.656e-05
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.6
##          Detection Rate : 0.6
##    Detection Prevalence : 0.6
##       Balanced Accuracy : 1.0
##
```

```
##        'Positive' Class : setosa
##
```

```
p <- predict(rf,test[-5],type="prob")
predobj <- prediction(p[,2],test$Species, label.ordering=c("setosa","versicolor"))
performance <- performance(predobj,"tpr","fpr")

plot(performance)
```



ROC
## 6 Discussion

The model performs well enough. Since the performance reaches 100% accuracy indepent of number of the Trees, the lowest number of trees is taken. Higher number of trees increases complexity and evaluation time. Hence, it is a tradeof between complexity, time and performance. As letter stays the same there is no need for an increased number of trees.