DS-6030 Homework Module 8

Tom Lever

07/16/2023

DS 6030 | Spring 2023 | University of Virginia

7. In the lab, we applied random forests to the Boston data using `mtry = 6` and using `ntree = 25` and `ntree = 500`.

   Create a plot displaying the test error resulting from random forests on this data set for a more comprehensive range of values for mtry and ntree. You can model your plot after Figure 8.10. Describe the results obtained.

```r
library(ISLR2)
library(randomForest)
```

```
# randomForest 4.7-1.1

# Type rfNews() to see new features/changes/bug fixes.
```

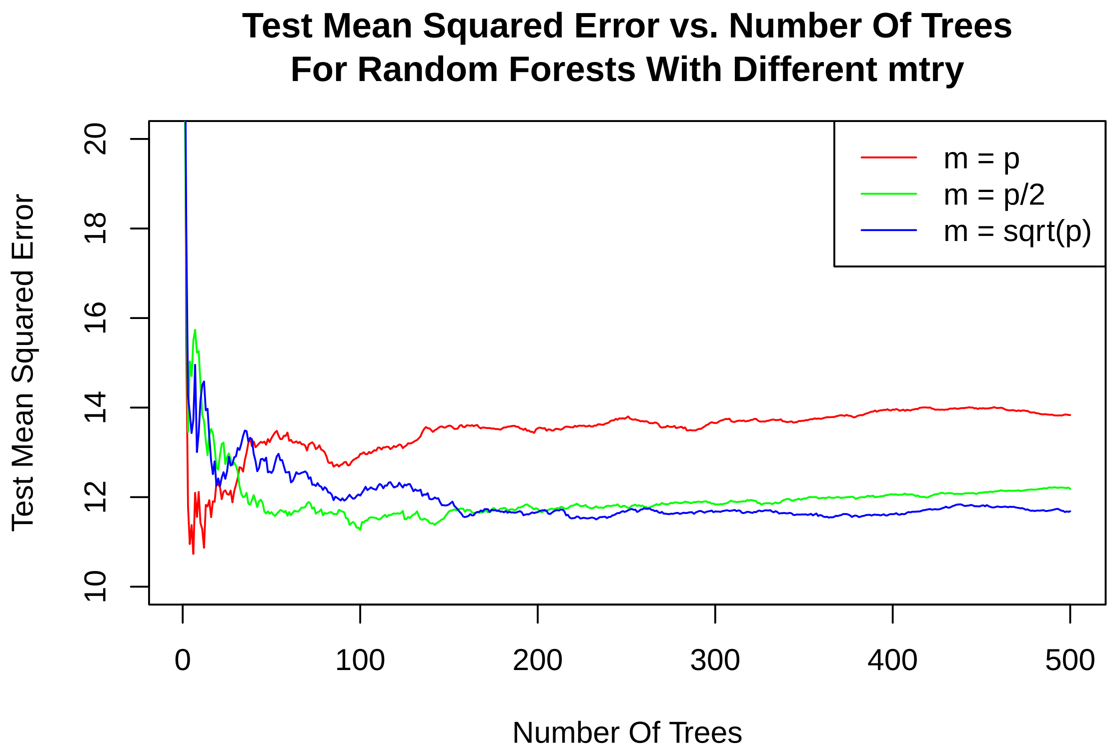
```r
library(TomLeversRPackage)

set.seed(1)
training_and_testing_data <- split_data_set_into_training_and_testing_data(
    Boston,
    proportion_of_training_data = 0.9
)
training_data <- training_and_testing_data$training_data
testing_data <- training_and_testing_data$testing_data
head(training_data, n = 3)
```

```
#        crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
# 505 0.10959  0 11.93    0 0.573 6.794 89.3 2.3889   1 273    21.0  6.48 22.0
# 324 0.28392  0  7.38    0 0.493 5.708 74.3 4.7211   5 287    19.6 11.74 18.5
# 167 2.01019  0 19.58    0 0.605 7.929 96.2 2.0459   5 403    14.7  3.70 50.0
```

```r
index_of_column_medv <- get_index_of_column_of_data_frame(training_data, "medv")
data_frame_of_training_predictors <- training_data[, -index_of_column_medv]
number_of_predictors <- ncol(data_frame_of_training_predictors)
data_frame_of_training_response_values <- training_data[, index_of_column_medv]
data_frame_of_testing_predictors <- testing_data[, -index_of_column_medv]
data_frame_of_testing_response_values <- testing_data[, index_of_column_medv]
randomForest_for_mtry_equal_to_number_of_predictors <- randomForest(
    x = data_frame_of_training_predictors,
```

```r
    y = data_frame_of_training_response_values,
    xtest = data_frame_of_testing_predictors,
    ytest = data_frame_of_testing_response_values,
    mtry = number_of_predictors,
    ntree = 500
)
randomForest_for_mtry_equal_to_half_number_of_predictors <- randomForest(
    x = data_frame_of_training_predictors,
    y = data_frame_of_training_response_values,
    xtest = data_frame_of_testing_predictors,
    ytest = data_frame_of_testing_response_values,
    mtry = number_of_predictors / 2,
    ntree = 500
)
randomForest_for_mtry_equal_to_square_root_of_number_of_predictors <- randomForest(
    x = data_frame_of_training_predictors,
    y = data_frame_of_training_response_values,
    xtest = data_frame_of_testing_predictors,
    ytest = data_frame_of_testing_response_values,
    mtry = sqrt(number_of_predictors),
    ntree = 500
)
plot(
    x = 1:500,
    y = randomForest_for_mtry_equal_to_number_of_predictors$test$mse,
    ylim = c(10, 20),
    col = "red",
    type = "l",
    xlab = "Number Of Trees",
    ylab = "Test Mean Squared Error",
    main = "Test Mean Squared Error vs. Number Of Trees\nFor Random Forests With Different mtry"
)
lines(
    x = 1:500,
    y = randomForest_for_mtry_equal_to_half_number_of_predictors$test$mse,
    col = "green"
)
lines(
    x = 1:500,
    y = randomForest_for_mtry_equal_to_square_root_of_number_of_predictors$test$mse,
    col = "blue"
)
legend(
    x = "topright",
    legend = c("m = p", "m = p/2", "m = sqrt(p)"),
    col = c("red", "green", "blue"),
    lty = 1
)
```



Above is a plot of Test Mean Squared Error for random forests predicting median value of owner-occupied homes in thousands of dollars based on the other variables of data set `ISLR2::Boston`. Variable *mtry* represents the number of variables considered at each split. Red, green, and blue curves correspond to random forests with *mtry* equal to the number of predictors, half the number of predictors, and the square root of the number of predictors, respectively. For each curve, Test Mean Squared Error decreases exponentially with number of trees. A random forest with *mtry* equal to the number of predictors and a number of trees less than 25 has the lowest Test Mean Squared Error and performs best.

8. This question uses the `Caravan` data set.

   (a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

```
set.seed(1)
Caravan$Purchase <- ifelse(Caravan$Purchase == "Yes", 1, 0)
training_data <- Caravan[1:1000, ]
testing_data <- Caravan[-c(1:1000), ]
```

   (b) Fit a boosting model to the training set with `Purchase` as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

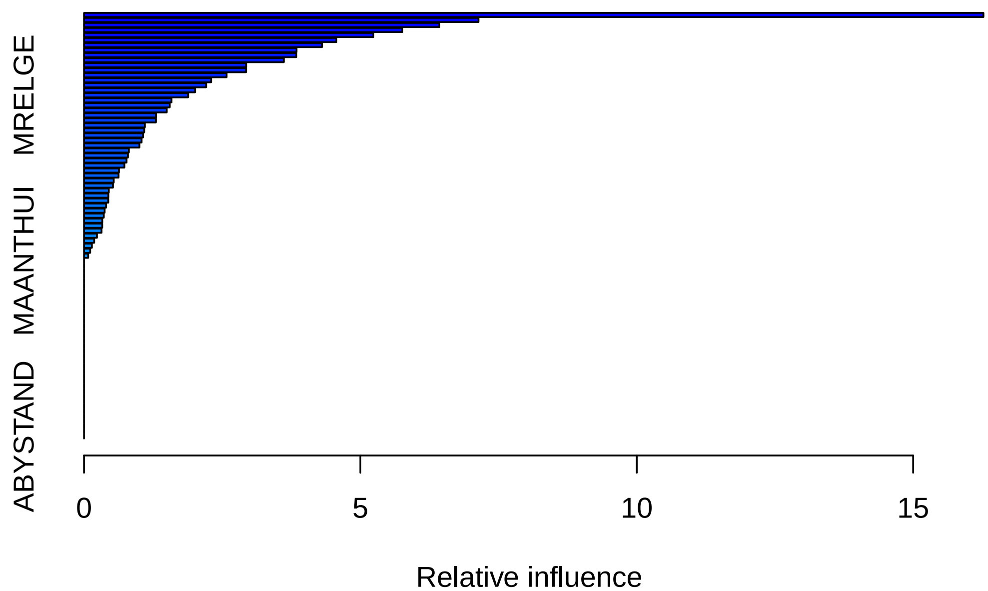
```
the_gbm.object <- gbm::gbm(
    formula = Purchase ~ .,
    distribution = "adaboost",
    data = training_data,
    n.trees = 1000,
```

```r
    shrinkage = 0.01
)
```

```
# Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
# : variable 50: PVRAAUT has no variation.

# Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution,
# : variable 71: AVRAAUT has no variation.
```

```r
summary(the_gbm.object)
```




```
#                 var     rel.inf
# PPERSAUT PPERSAUT 16.27175732
# MKOOPKLA MKOOPKLA  7.13694190
# MBERMIDD MBERMIDD  6.42793466
# MOPLHOOG MOPLHOOG  5.75847398
# PBRAND     PBRAND  5.23437327
# MGODGE     MGODGE  4.56525360
# MINK3045 MINK3045  4.30566970
# MINKM30   MINKM30  3.84523657
# MAUT1       MAUT1  3.84089359
# MSKC         MSKC  3.61528574
# MBERARBG MBERARBG  2.93289220
# MAUT2       MAUT2  2.93161159
# MOSTYPE   MOSTYPE  2.58027508
# MSKA         MSKA  2.29951574
# PWAPART   PWAPART  2.20956399
# MBERHOOG MBERHOOG  2.00974939
# MRELGE     MRELGE  1.88421014
```

```
# MGODOV     MGODOV  1.58363134
# MINKGEM   MINKGEM  1.55185993
# PBYSTAND PBYSTAND  1.49720553
# MGODPR     MGODPR  1.30330440
# ABRAND     ABRAND  1.30150833
# MZFONDS   MZFONDS  1.10115776
# PMOTSCO   PMOTSCO  1.08936547
# MSKD         MSKD  1.06835520
# MHHUUR     MHHUUR  1.04244317
# MSKB1       MSKB1  1.00244172
# MSKB2       MSKB2  0.81205672
# MBERBOER MBERBOER  0.79494102
# MAUT0       MAUT0  0.76986818
# MINK4575 MINK4575  0.73016367
# MRELOV     MRELOV  0.63206423
# MOPLMIDD MOPLMIDD  0.62520980
# MHKOOP     MHKOOP  0.53804087
# MOSHOOFD MOSHOOFD  0.52356550
# MGEMOMV   MGEMOMV  0.44888986
# MFWEKIND MFWEKIND  0.44018334
# MGODRK     MGODRK  0.43872282
# MRELSA     MRELSA  0.39937493
# MGEMLEEF MGEMLEEF  0.37354662
# MFGEKIND MFGEKIND  0.35808623
# MFALLEEN MFALLEEN  0.33030603
# APERSAUT APERSAUT  0.32944210
# MINK7512 MINK7512  0.31701634
# MBERARBO MBERARBO  0.23420446
# MOPLLAAG MOPLLAAG  0.18394434
# MINK123M MINK123M  0.14264058
# MBERZELF MBERZELF  0.11140151
# MZPART     MZPART  0.07541953
# MAANTHUI MAANTHUI  0.00000000
# PWABEDR   PWABEDR  0.00000000
# PWALAND   PWALAND  0.00000000
# PBESAUT   PBESAUT  0.00000000
# PVRAAUT   PVRAAUT  0.00000000
# PAANHANG PAANHANG  0.00000000
# PTRACTOR PTRACTOR  0.00000000
# PWERKT     PWERKT  0.00000000
# PBROM       PBROM  0.00000000
# PLEVEN     PLEVEN  0.00000000
# PPERSONG PPERSONG  0.00000000
# PGEZONG   PGEZONG  0.00000000
# PWAOREG   PWAOREG  0.00000000
# PZEILPL   PZEILPL  0.00000000
# PPLEZIER PPLEZIER  0.00000000
# PFIETS     PFIETS  0.00000000
# PINBOED   PINBOED  0.00000000
# AWAPART   AWAPART  0.00000000
# AWABEDR   AWABEDR  0.00000000
# AWALAND   AWALAND  0.00000000
# ABESAUT   ABESAUT  0.00000000
# AMOTSCO   AMOTSCO  0.00000000
```

```
# AVRAAUT   AVRAAUT  0.00000000
# AAANHANG AAANHANG  0.00000000
# ATRACTOR ATRACTOR  0.00000000
# AWERKT     AWERKT  0.00000000
# ABROM       ABROM  0.00000000
# ALEVEN     ALEVEN  0.00000000
# APERSONG APERSONG  0.00000000
# AGEZONG   AGEZONG  0.00000000
# AWAOREG   AWAOREG  0.00000000
# AZEILPL   AZEILPL  0.00000000
# APLEZIER APLEZIER  0.00000000
# AFIETS     AFIETS  0.00000000
# AINBOED   AINBOED  0.00000000
# ABYSTAND ABYSTAND  0.00000000
```

According to Understanding Gradient Boosting Machines, an "important feature in the gbm modelling is the Variable Importance. Applying the summary function to a gbm output produces both a Variable Importance Table and a Plot of the model. This table [above] ranks the individual variables based on their relative influence, which is a measure indicating the relative importance of each variable in training the model."

$PPERSAUT$ and $MKOOPKLA$ appear to our most important predictors.

(c) Use the boosting model to predict the response on the test data. Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %. Form a confusion matrix. What fraction of the people predicted to make a purchase do in fact make one? How does this compare with the results obtained from applying KNN or logistic regression to this data set?

```
vector_of_predicted_probabilities <- predict(
    object = the_gbm.object,
    testing_data,
    n.trees = 1000,
    type = "response"
)
vector_of_predictions <- ifelse(
    test = vector_of_predicted_probabilities > 0.2,
    yes = 1,
    no = 0
)
table(testing_data$Purchase, vector_of_predictions)
```

```
#    vector_of_predictions
#        0    1
#   0 4470   63
#   1  270   19
```

The fraction of the people predicted to make a purchase that do in fact make one and the precision of our Generalized Boosting Model for an AdaBoost distribution and threshold of 0.2 $PPV = \frac{TP}{TP+FP} = \frac{19}{19+63} = 0.232$.

```
logistic_regression_model <- glm(
    formula = Purchase ~ .,
    data = training_data,
    family = "binomial"
)
```

```
# Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
vector_of_predicted_probabilities <- predict(
    object = logistic_regression_model,
    testing_data,
    type = "response"
)
```

```
# Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
# prediction from rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```r
vector_of_predictions <- ifelse(
    test = vector_of_predicted_probabilities > 0.2,
    yes = 1,
    no = 0
)
table(testing_data$Purchase, vector_of_predictions)
```

```
#    vector_of_predictions
#        0    1
#   0 4183  350
#   1  231   58
```

The precision of a logistic regression model with threshold 0.2 is $PPV = \frac{58}{58+350} = 0.142$. The rate of difference in precision between our Generalized Boosting Model and our logistic regression model is $\frac{0.232-0.142}{0.142} = 0.634$.