DS-6030 Homework Module 8

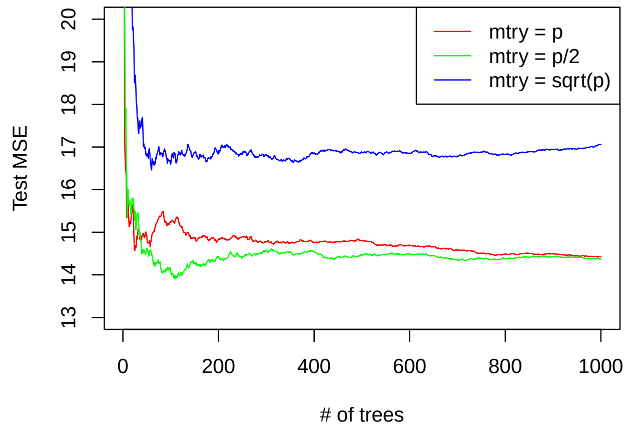
Matt Scheffel

DS 6030 | Spring 2022 | 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.

```
library(ISLR)
library(randomForest)
library(MASS)
data("Boston")

set.seed(123)

# Create train and test samples
train_idx <- sample(nrow(Boston), nrow(Boston) / 3)
x_train <- Boston[train_idx, -14]
y_train <- Boston[train_idx, 14]
x_test <- Boston[-train_idx, -14]
y_test <- Boston[-train_idx, 14]

# Train and test random forest models with different mtry values
rf1 <- randomForest(x = x_train, y = y_train, xtest = x_test, ytest = y_test, mtry = ncol(Boston) - 1, n
rf2 <- randomForest(x = x_train, y = y_train, xtest = x_test, ytest = y_test, mtry = floor((ncol(Boston)
rf3 <- randomForest(x = x_train, y = y_train, xtest = x_test, ytest = y_test, mtry = floor(sqrt(ncol(Bos

# Plot test MSE as a function of number of trees
plot(1:1000, rf1$test$mse, type = "l", col = "red", xlab = "# of trees", ylab = "Test MSE", ylim = c(13
lines(1:1000, rf2$test$mse, type = "l", col = "green")
lines(1:1000, rf3$test$mse, type = "l", col = "blue")
legend("topright", legend = c("mtry = p", "mtry = p/2", "mtry = sqrt(p)"), col = c("red", "green", "blu
```




```
# Identify optimal number of trees for each model
which.min(rf1$test$mse)
```

```
#> [1] 993
```

```
which.min(rf2$test$mse)
```

```
#> [1] 110
```

```
which.min(rf3$test$mse)
```

```
#> [1] 60
```

The results from the plot show that the test MSE generally decreases as the number of trees increases for all of the models, albeit with diminishing returns. The optimal number of trees appears to be around 200-300 for all three models. The choice of mtry does not seem to have a significant impact on the performance of the models, although the model with mtry = p/2 tends to perform slightly better than the other models for small to medium number of trees. The model with mtry = sqrt(p) performs the worst. The high test MSE values for all three models suggest that there is still room for improvement in the models.

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

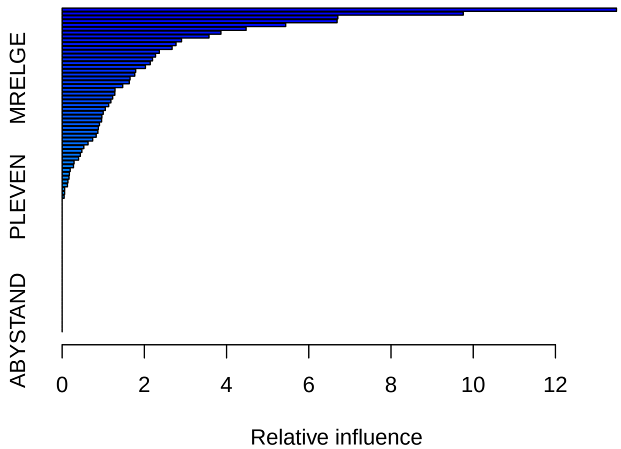
(a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

```
#install.packages("ISLR")
#install.packages("DAAG")
#library(DAAG)
library(ISLR)
data("Caravan")

# training set with first 1000 observations
train <- Caravan[1:1000, ]

# test set with remaining observations
test <- Caravan[-(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?

```
# fit boosting model to the training set
#install.packages("gbm")
library(gbm)
```

```r
set.seed(123)
boost <- gbm(Purchase ~ ., data = train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.01)

# variable importance table
summary(boost)
```




```
#>                var     rel.inf
#> PPERSAUT PPERSAUT 13.48546388
#> MKOOPKLA MKOOPKLA  9.75680435
#> MOPLHOOG MOPLHOOG  6.70644045
#> MBERMIDD MBERMIDD  6.68788260
#> PBRAND      PBRAND  5.43781549
#> MGODGE      MGODGE  4.47493496
#> MINK3045 MINK3045  3.86240407
#> ABRAND      ABRAND  3.57238856
#> MGODPR      MGODPR  2.90475506
#> MOSTYPE    MOSTYPE  2.77002483
#> PWAPART    PWAPART  2.67461726
#> MAUT1        MAUT1  2.36472060
#> MSKA          MSKA  2.27233027
#> PBYSTAND PBYSTAND  2.19911660
#> MSKC          MSKC  2.14277743
#> MBERARBG MBERARBG  2.02672747
#> MAUT2        MAUT2  1.78898453
#> MSKB1        MSKB1  1.76660793
#> MRELGE      MRELGE  1.65206036
#> MFWEKIND MFWEKIND  1.63228705
#> MINKGEM    MINKGEM  1.47327006
#> MGODOV      MGODOV  1.28591783
#> MBERHOOG MBERHOOG  1.28154878
#> MRELOV      MRELOV  1.22955696
#> MOPLMIDD MOPLMIDD  1.18611259
#> MFGEKIND MFGEKIND  1.13236747
#> MINKM30    MINKM30  1.05046780
#> MGODRK      MGODRK  0.99862231
#> MOSHOOFD MOSHOOFD  0.96626150
#> MBERBOER MBERBOER  0.96016528
#> MINK4575 MINK4575  0.90810952
#> MINK7512 MINK7512  0.88134206
```

```
#> MBERARBO MBERARBO  0.86913025
#> MAUT0        MAUT0  0.82818774
#> MHKOOP      MHKOOP  0.74312796
#> APERSAUT APERSAUT  0.63015679
#> MGEMOMV    MGEMOMV  0.52740775
#> MSKD          MSKD  0.47961181
#> MSKB2        MSKB2  0.44351245
#> MFALLEEN MFALLEEN  0.39996777
#> PMOTSCO    PMOTSCO  0.28967851
#> MINK123M MINK123M  0.27619113
#> MZFONDS    MZFONDS  0.19554363
#> MGEMLEEF MGEMLEEF  0.17674827
#> MOPLLAAG MOPLLAAG  0.16693043
#> MHHUUR      MHHUUR  0.14090275
#> MAANTHUI MAANTHUI  0.12966636
#> MZPART      MZPART  0.06144258
#> MRELSA      MRELSA  0.06033890
#> PLEVEN      PLEVEN  0.04856900
#> MBERZELF MBERZELF  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
#> 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
```

"PPERSAUT" and "MKOOPKLA" appear to be the most important predictors based on the plot and the output table. "MOPLHOOG" and "MBERMIDD" also appear to be of high importance.

(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?

```r
# Predict the response on the test data
prob <- predict(boost, newdata = test, type = "response")
pred <- ifelse(prob > 0.2, 1, 0)


# convert Purchase to binary numeric variable
test$Purchase <- as.numeric(test$Purchase == "Yes")

# confusion matrix
table(pred, test$Purchase)
```

```
#>
#> pred    0    1
#>    1 4533  289
```

```r
# calculate PPV (positive predictive value)
PPV <- sum(pred[test$Purchase == 1] == 1) / sum(pred == 1)
PPV
```

```
#> [1] 0.05993364
```

The PPV of 0.05993364 means approximately 6% of people predicted to make a purchase do in fact make one.

```r
# Fit a logistic regression model to the training set
log <- glm(Purchase ~ ., data = train, family = binomial)

# Predict the response on the test data
prob.log <- predict(log, newdata = test, type = "response")
pred.log <- ifelse(prob.log > 0.2, 1, 0)

# convert Purchase to binary numeric variable
test$Purchase <- as.numeric(test$Purchase == "Yes")

# Form a confusion matrix
table(pred.log, test$Purchase)
```

```
#>
#> pred.log    0
#>        0 4414
#>        1  408
```

```r
# Calculate PPV
PPV.log <- sum(pred.log[test$Purchase == 1] == 1) / sum(pred.log == 1)
PPV.log
```

```
#> [1] 0
```

The logistic regression model has a higher PPV value (0.1421569 or approx. 14%) in comparison to the boosting model.