Homework 8

STAT 430, Spring 2017

Due: Friday, April 7 by 11:59 PM

Please see the homework instructions document for detailed instructions and some grading notes. Failure to follow instructions will result in point reductions.

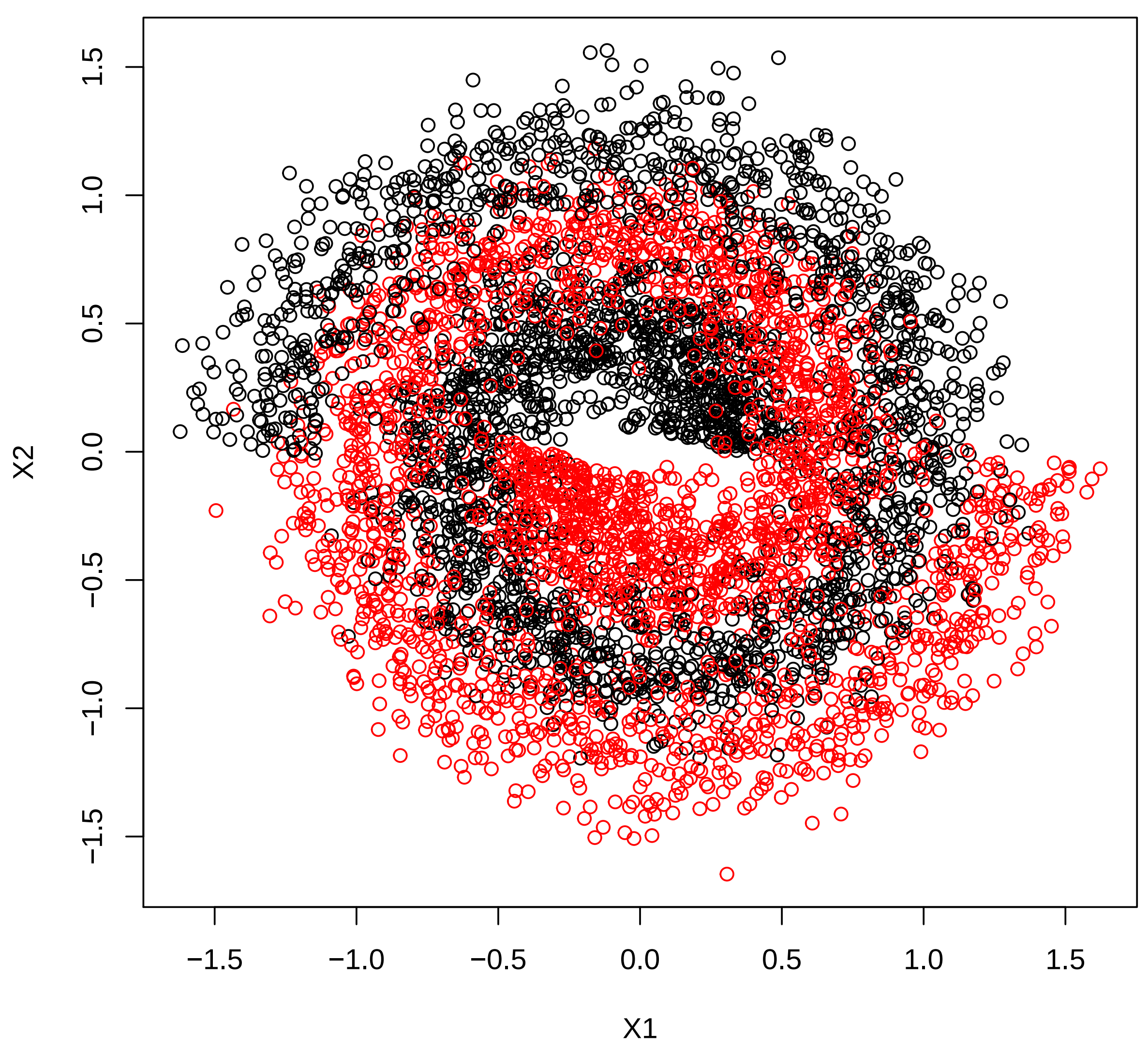
Exercise 1

[**12 points**] For this exercise we will create data via simulation, then asses how well certain methods perform. Use the code below to create a train and test dataset.

```r
library(mlbench)
set.seed(42)
sim_trn = mlbench.spirals(n = 5000, cycles = 1.5, sd = 0.15)
sim_trn = data.frame(sim_trn$x, class = as.factor(sim_trn$classes))
sim_tst = mlbench.spirals(n = 10000, cycles = 1.5, sd = 0.15)
sim_tst = data.frame(sim_tst$x, class = as.factor(sim_tst$classes))
```

The training data is plotted below, with colors indicating the `class` variable, which is the response.

```r
plot(sim_trn$X1, sim_trn$X2, col = sim_trn$class,
     xlab = "X1", ylab = "X2")
```



Before proceeding further, set a seed equal to your UIN.

```r
uin = 123456789
set.seed(uin)
```

We'll use the following to define 5-fold cross-validation for use with `train()` from `caret`.

```r
library(caret)
cv_5 = trainControl(method = "cv", number = 5)
```

We now tune two models with `train()`. First, a logistic regression using `glm`. (This actually isn't "tuned" as there are not parameters to be tuned, but we use `train()` to perform cross-validation.) Second we tune a single decision tree using `rpart`.

We store the results in `sim_glm_cv` and `sim_tree_cv` respectively, but we also wrap both function calls with `system.time()` in order to record how long the tuning process takes for each method.

```r
glm_cv_time = system.time({
  sim_glm_cv  = train(
    class ~ .,
    data = sim_trn,
    trControl = cv_5,
    method = "glm")
})

tree_cv_time = system.time({
  sim_tree_cv = train(
    class ~ .,
    data = sim_trn,
    trControl = cv_5,
    method = "rpart")
})
```

We see that both methods are tuned via cross-validation in a similar amount of time.

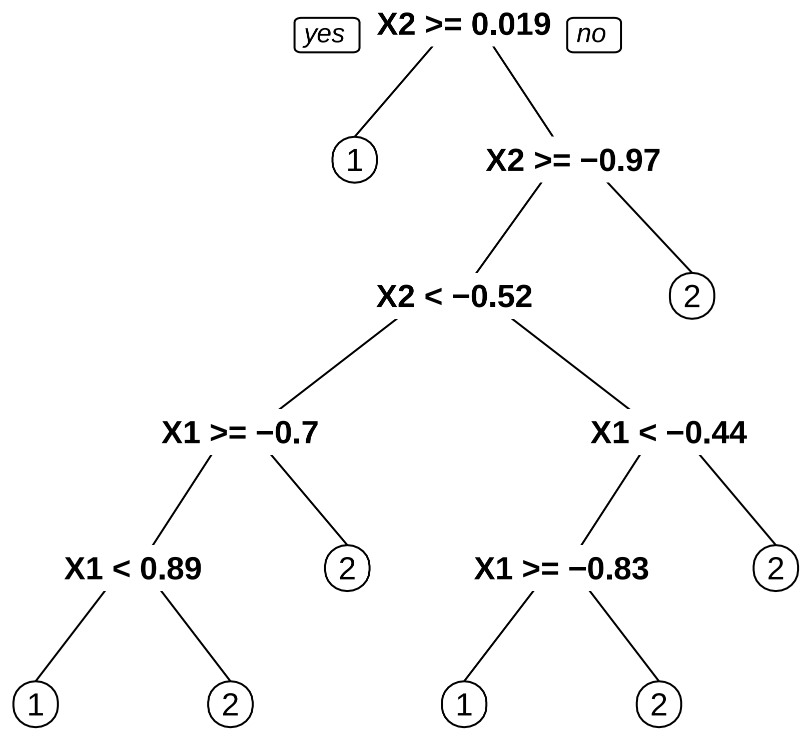
```r
glm_cv_time["elapsed"]
```

```
## elapsed
##    0.98
```

```r
tree_cv_time["elapsed"]
```

```
## elapsed
##    0.72
```

```r
library(rpart.plot)
prp(sim_tree_cv$finalModel)
```



Repeat the above analysis using a random forest, twice. The first time use 5-fold cross-validation. The second time, tune the model using OOB samples. We only have two predictors here, so, for both, use the following tuning grid.

```
rf_grid = expand.grid(mtry = c(1, 2))
```

(a) Compare the time taken to tune each model. Is the difference between the OOB and CV result for the random forest similar to what you would have expected?

(b) Compare the tuned value of `mtry` for each of the random forests tuned. Do they choose the same model?

(c) Report the CV and OOB accuracy for the random forests.

(d) Compare the test accuracy of each of the four procedures considered. Briefly explain these results.

Exercise 2

[**12 points**] For this question we will predict the `Salary` of `Hitters`. (`Hitters` is also the name of the dataset.) We first remove the missing data:

```
library(ISLR)
Hitters = na.omit(Hitters)
```

After changing `uin` to your UIN, use the following code to test-train split the data.

```
uin = 123456789
set.seed(uin)
hit_idx = createDataPartition(Hitters$Salary, p = 0.6, list = FALSE)
```

```
hit_trn = Hitters[hit_idx,]
hit_tst = Hitters[-hit_idx,]
```

(a) Tune a boosted tree model using the following tuning grid and 5-fold cross-validation. Create a plot that shows the tuning results.

```
gbm_grid = expand.grid(interaction.depth = c(1, 2),
                       n.trees = c(500, 1000, 1500),
                       shrinkage = c(0.001, 0.01, 0.1),
                       n.minobsinnode = 10)
```

(b) What is the CV-RMSE and tuning parameters of the tuned boosted tree model?

(c) According to the boosted model, what are the two most important predictors?

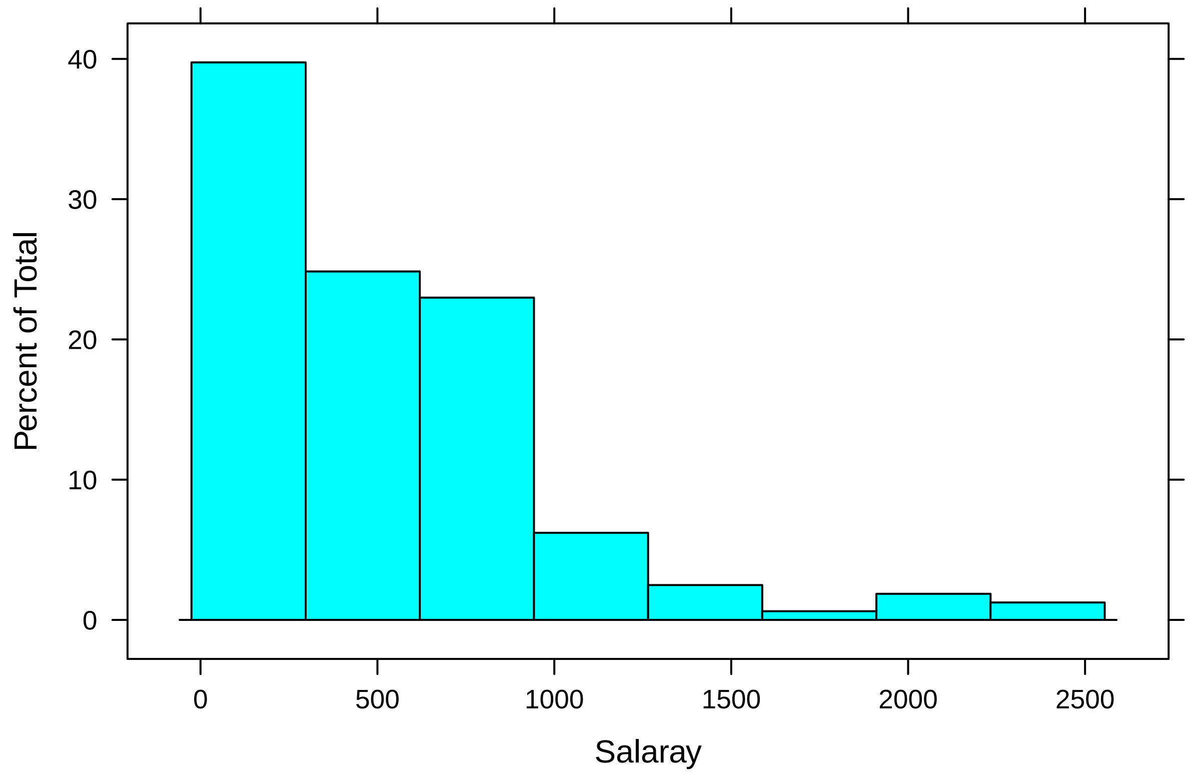
(d) Tune a random forest using OOB resampling and **all** possible values of `mtry`. Report the best value of `mtry` as well as the OOB RMSE for both the best model as well as the bagged model. (It is possible, but unlikely that they are the same. If they are the same, simply report that the best model is the bagged model and report only that model.)

(e) Report the test RMSE for the tuned boosted tree model, the tuned random forest, and a bagged tree model.

Exercise 3

[**6 points**] Continue with the data from Exercise 2. The book, ISL, suggests log transforming the response, `Salary`, before fitting a random forest. Is this necessary? Re-tune a random forest as you did in Exercise 2, except with a log transformed response. Report test RMSE for both the untransformed and transformed model. Based on these results, do you think the transformation was necessary?

```
histogram(hit_trn$Salary, xlab = "Salaray")
```



Percent of Total
40
30
20
10
0
0
500
1000
1500
2000
2500
Salaray