Homework 07

STAT 430, Fall 2017

Due: Friday, November 3, 11:59 PM

You should use the `caret` package and training pipeline to complete this homework. **Any time you use the `train()` function, first run `set.seed(1337)`.**

```r
library(caret)
library(mlbench)
```

```r
library(randomForest)
library(gbm)
library(klaR)
```

Exercise 1 (Regression with `caret`)

[**10 points**] For this exercise we will train a number of regression models for the `Boston` data from the `MASS` package. Use `medv` as the response and all other variables as predictors. Use the test-train split given below. When tuning models and reporting cross-validated error, use 5-fold cross-validation.

```r
data(Boston, package = "MASS")
set.seed(42)
bstn_idx = createDataPartition(Boston$medv, p = 0.80, list = FALSE)
bstn_trn = Boston[bstn_idx, ]
bstn_tst = Boston[-bstn_idx, ]
```

Fit a total of five models:

- An additive linear regression
- A well tuned k-nearest neighbors model.
  - Do **not** scale the predictors.
  - Consider $k \in \{1, 5, 10, 15, 20, 25\}$
- Another well tuned k-nearest neighbors model.
  - **Do** scale the predictors.
  - Consider $k \in \{1, 5, 10, 15, 20, 25\}$
- A random forest
  - Use the default tuning parameters chosen by `caret`
- A boosted tree model
  - Use the provided tuning grid below

```r
gbm_grid = expand.grid(interaction.depth = c(1, 2, 3),
                       n.trees = (1:20) * 100,
                       shrinkage = c(0.1, 0.3),
                       n.minobsinnode = 20)
```

Provide plots of error versus tuning parameters for the two k-nearest neighbors models and the boosted tree model. Also provide a table that summarizes the cross-validated and test RMSE for each of the five (tuned) models.

Solution:

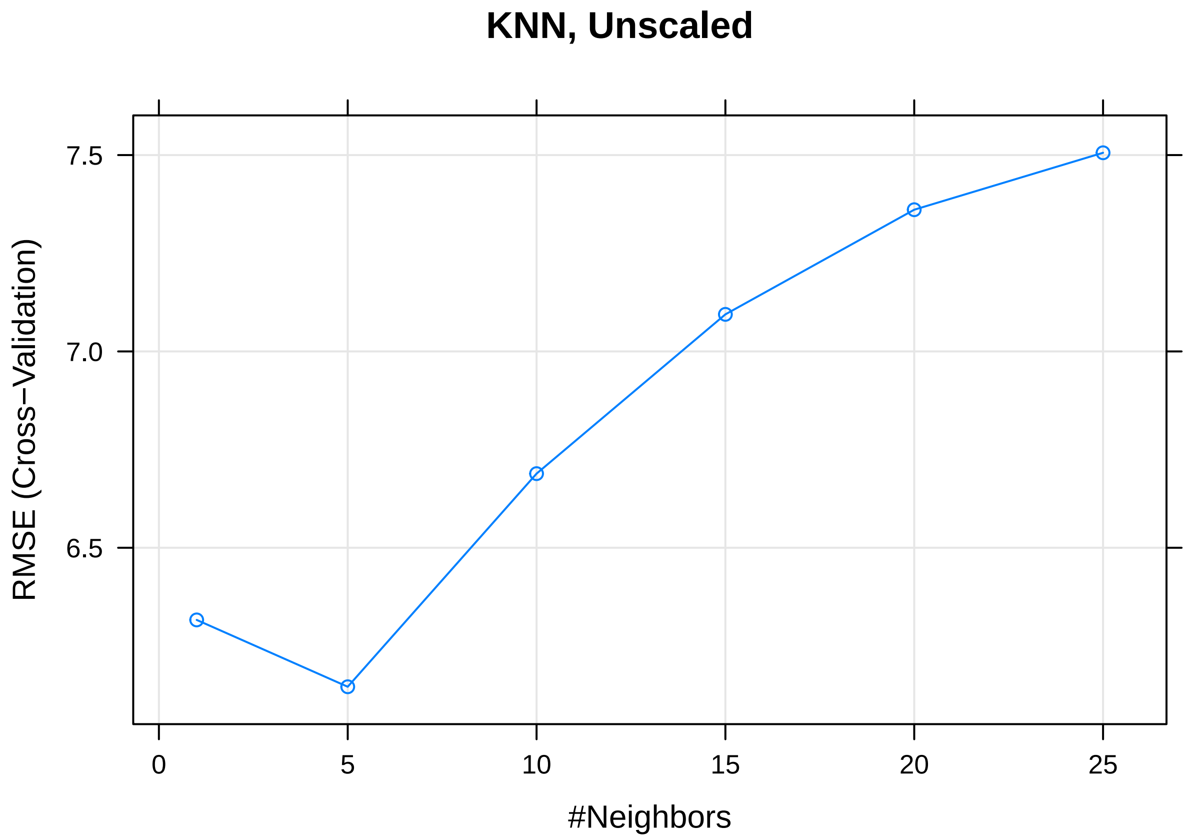
```r
set.seed(1337)
bstn_lm_mod = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "lm"
)
```

```r
set.seed(1337)
bstn_knnu_mod = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "knn",
  tuneGrid = expand.grid(k = c(1, 5, 10, 15, 20, 25))
)
```

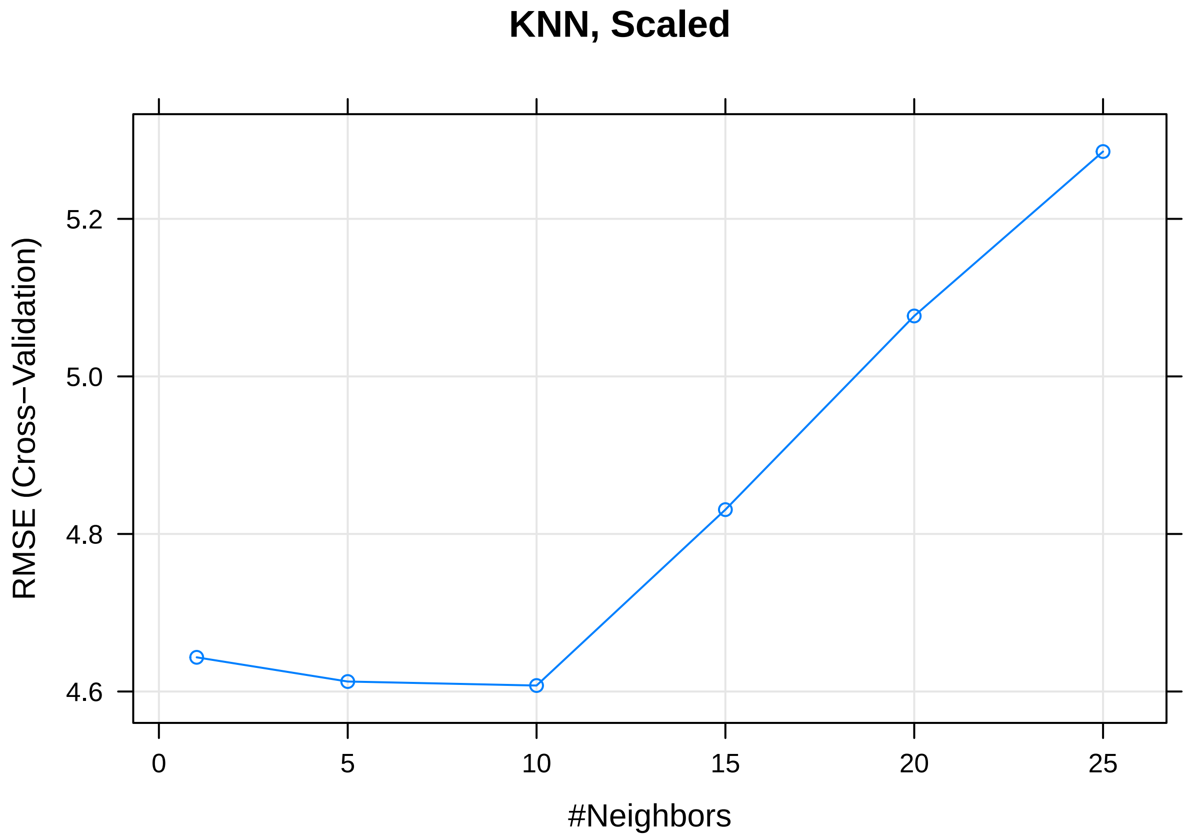
```r
set.seed(1337)
bstn_knns_mod = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale"),
  method = "knn",
  tuneGrid = expand.grid(k = c(1, 5, 10, 15, 20, 25))
)
```

```r
set.seed(1337)
bstn_rf_mod = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "rf"
)
```

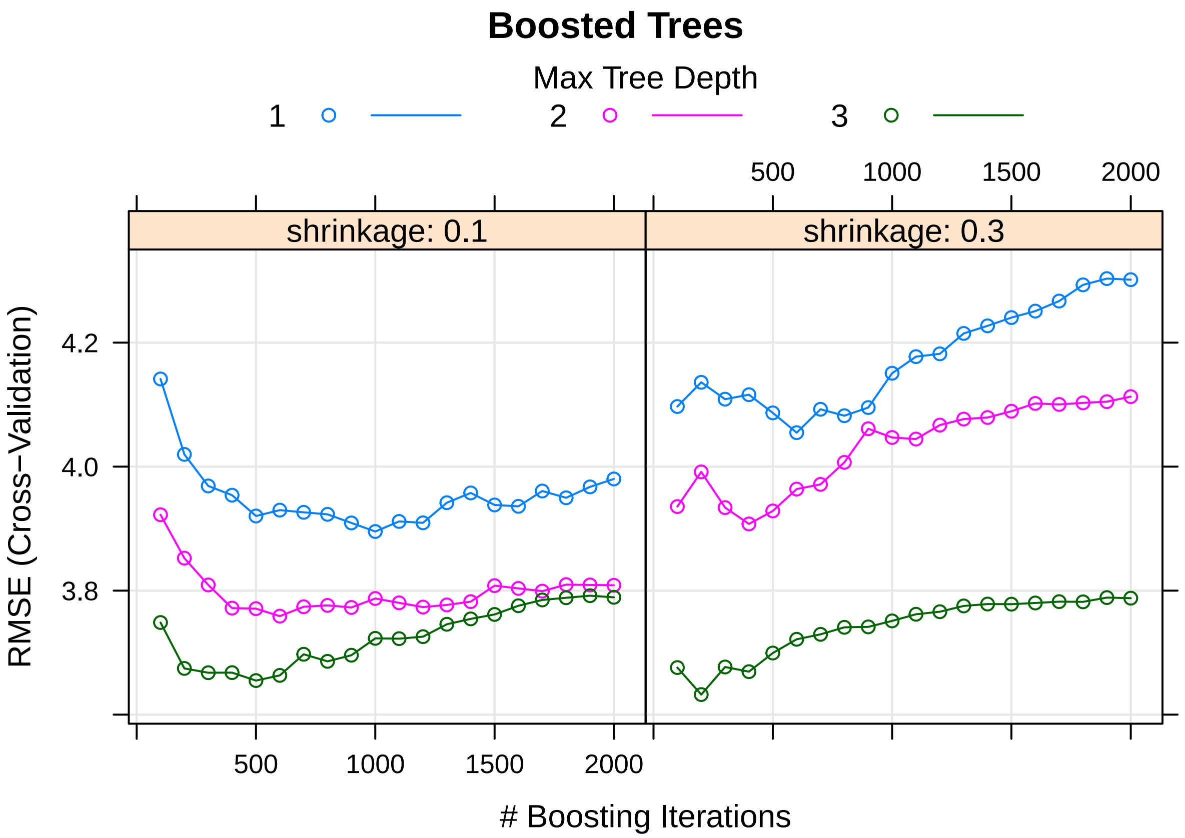
```r
set.seed(1337)
bstn_gbm_mod = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "gbm",
  tuneGrid = gbm_grid,
  verbose = FALSE
)
```



KNN, Unscaled



KNN, Scaled



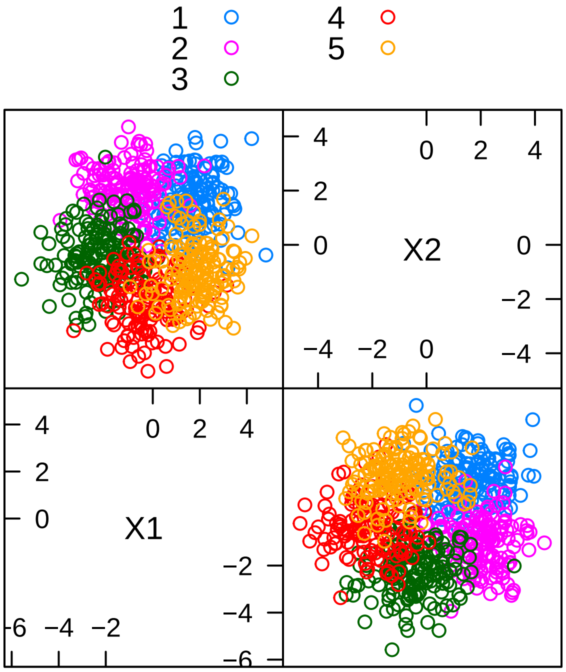
Method	CV RMSE	Test RMSE
Linear Regression	4.84	4.99
KNN, Unscaled	6.15	6.49
KNN, Scaled	4.61	5.46
Random Forest	3.28	3.03
Boosted Trees	3.63	3.67

Exercise 2 (Clasification with `caret`)

[**10 points**] For this exercise we will train a number of classifiers using the training data generated below. The categorical response variable is `classes` and the remaining variables should be used as predictors. When tuning models and reporting cross-validated error, use 10-fold cross-validation.

```
set.seed(42)
sim_trn = mlbench::mlbench.2dnormals(n = 750, cl = 5)
sim_trn = data.frame(
  classes = sim_trn$classes,
  sim_trn$x
)
```

```
caret::featurePlot(x = sim_trn[, -1],
            y = sim_trn$classes,
            plot = "pairs",
            auto.key = list(columns = 2))
```



Scatter Plot Matrix

Fit a total of four models:

- LDA
- QDA
- Naive Bayes
- Regularized Discriminant Analysis (RDA)
  - Use method **rda** with **caret** which requires the **klaR** package
  - Use the default tuning grid

Provide a plot of acuracy versus tuning parameters for the RDA model. Also provide a table that summarizes the cross-validated accuracy and their standard deviations for each of the four (tuned) models.

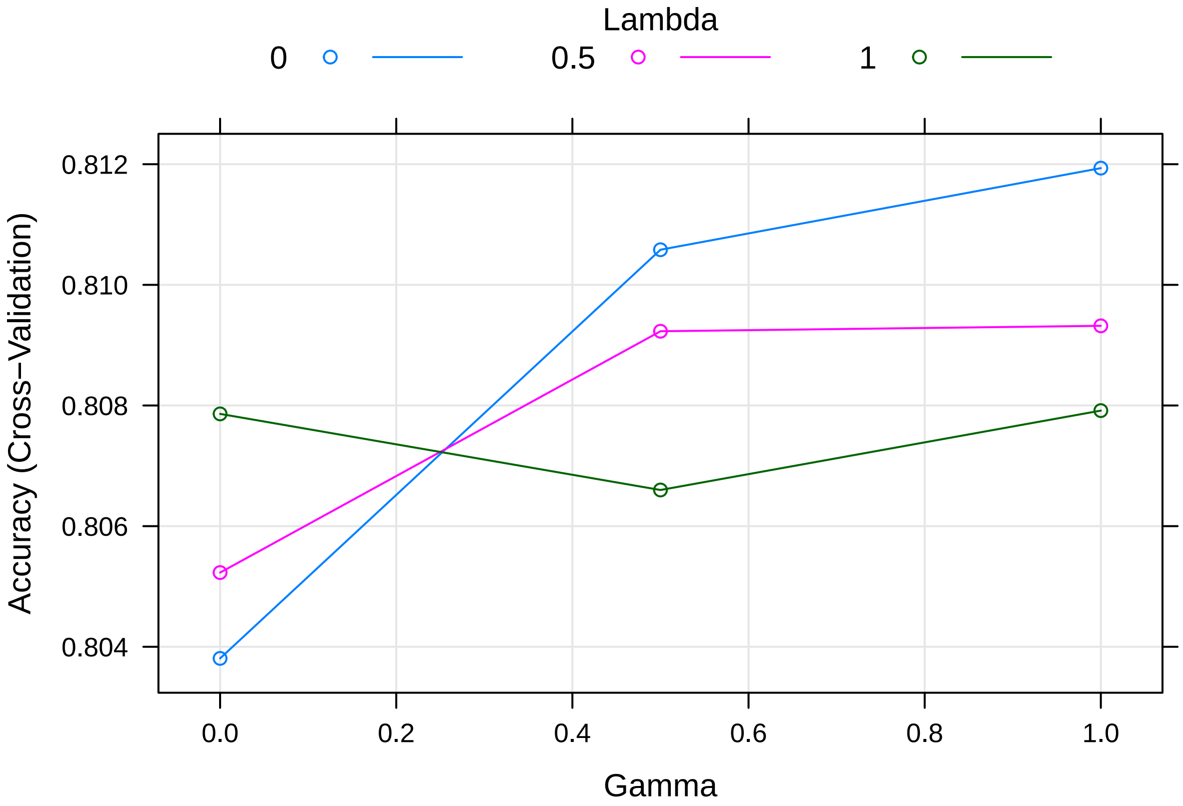
Solution:

```
set.seed(1337)
sim_lda_mod = train(
  classes ~ .,
  data = sim_trn,
  trControl = trainControl(method = "cv", number = 10),
  method = "lda"
)
```

```
set.seed(1337)
sim_qda_mod = train(
  classes ~ .,
  data = sim_trn,
  trControl = trainControl(method = "cv", number = 10),
  method = "qda"
)
```

```r
set.seed(1337)
sim_nb_mod = train(
  classes ~ .,
  data = sim_trn,
  trControl = trainControl(method = "cv", number = 10),
  method = "nb"
)
```

```r
set.seed(1337)
sim_rda_mod = train(
  classes ~ .,
  data = sim_trn,
  trControl = trainControl(method = "cv", number = 10),
  method = "rda"
)
```



Method	CV Acc	SD CV Acc
LDA	0.8079	0.0357
QDA	0.8038	0.0385
Naive Bayes	0.8118	0.0367
RDA	0.8119	0.0379

Exercise 3 (Concept Checks)

[**1 point each**] Answer the following questions based on your results from the three exercises.

Regression

(a) What value of k is chosen for KNN without predictor scaling?

```
bstn_knnu_mod$bestTune$k
```

```
## [1] 5
```

(b) What value of k is chosen for KNN **with** predictor scaling?

```
bstn_knns_mod$bestTune$k
```

```
## [1] 10
```

(c) What are the values of the tuning parameters chosen for the boosted tree model?

```
bstn_gbm_mod$bestTune
```

```
##     n.trees interaction.depth shrinkage n.minobsinnode
## 102     200                 3       0.3             20
```

(d) Which method achieves the lowest cross-validated error?

```
reg_results[reg_results$`CV RMSE` == min(reg_results$`CV RMSE`), ]
```

```
##          Method  CV RMSE Test RMSE
## 4 Random Forest 3.277206  3.033097
```

(e) Which method achieves the lowest test error?

```
reg_results[reg_results$`Test RMSE` == min(reg_results$`Test RMSE`), ]
```

```
##          Method  CV RMSE Test RMSE
## 4 Random Forest 3.277206  3.033097
```

Classification

(f) What are the values of the tuning parameters chosen for the RDA model?

```
sim_rda_mod$bestTune
```

```
##   gamma lambda
## 7     1      0
```

(g) Based on the scatterplot, which method, LDA or QDA, do you think is *more* appropriate? Explain.

LDA. The covariance seems to be the same in each class.

(h) Based on the scatterplot, which method, QDA or Naive Bayes, do you think is *more* appropriate? Explain.

Naive Bayes. The predictors seem to be independent in each class.

(i) Which model achieves the best cross-validated accuracy?

```
class_results[class_results$`CV Acc` == max(class_results$`CV Acc`), ]
```

```
##   Method   CV Acc  SD CV Acc
## 4    RDA 0.811935 0.03791713
```

(j) Do you believe the model in **(i)** is the model that should be chosen? Explain.

```
rda_res = class_results[class_results$`CV Acc` == max(class_results$`CV Acc`), ]
rda_res$`CV Acc` + rda_res$`SD CV Acc`
```

```
## [1] 0.8498521
```

No. The results of all the other model are within one SE. We should pick a less complex model, perhpas LDA.