Homework 07

STAT 430, Fall 2017

Due: Friday, November 3, 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.

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)`.**

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

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.

```
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

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

```
set.seed(1337)
m1 = train(form = medv ~ ., data = bstn_trn,
           trControl = trainControl(method = "cv", number = 5),
           method = 'lm')
set.seed(1337)
m2 = train(
  medv ~ .,
```

```r
  data = bstn_trn,
  method = "knn",
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = expand.grid(k = c(1, 5, 10, 15, 20, 25))
  )
set.seed(1337)
m3 = train(medv ~ .,
  data = bstn_trn,
  method = "knn",
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = expand.grid(k = c(1,5,10,15,20,25)),
  preProcess = c('center','scale')
)
set.seed(1337)
m4 = train(medv ~., data = bstn_trn, method = 'rf',
           trControl = trainControl(method = 'cv', number = 5))
```

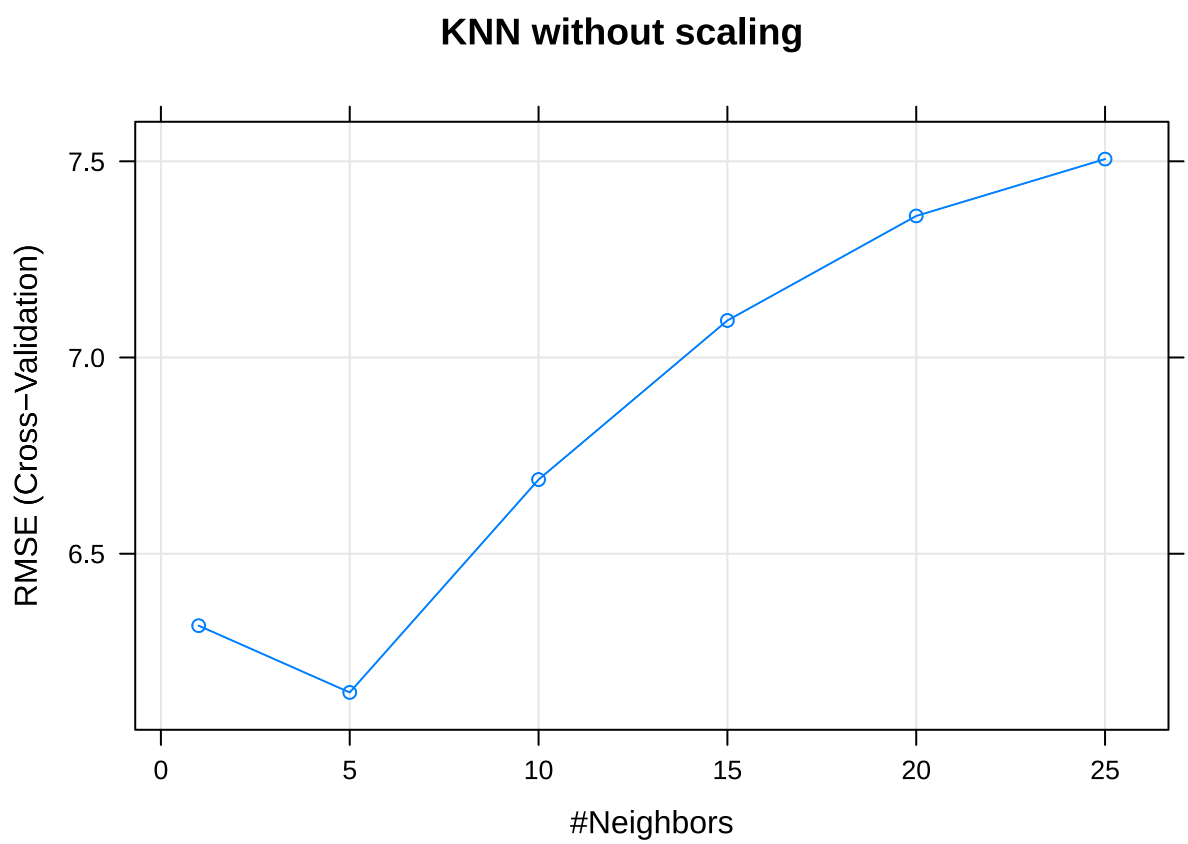
```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed(1337)
m5 = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "gbm",
  tuneGrid = gbm_grid,
  verbose = FALSE
  )
```

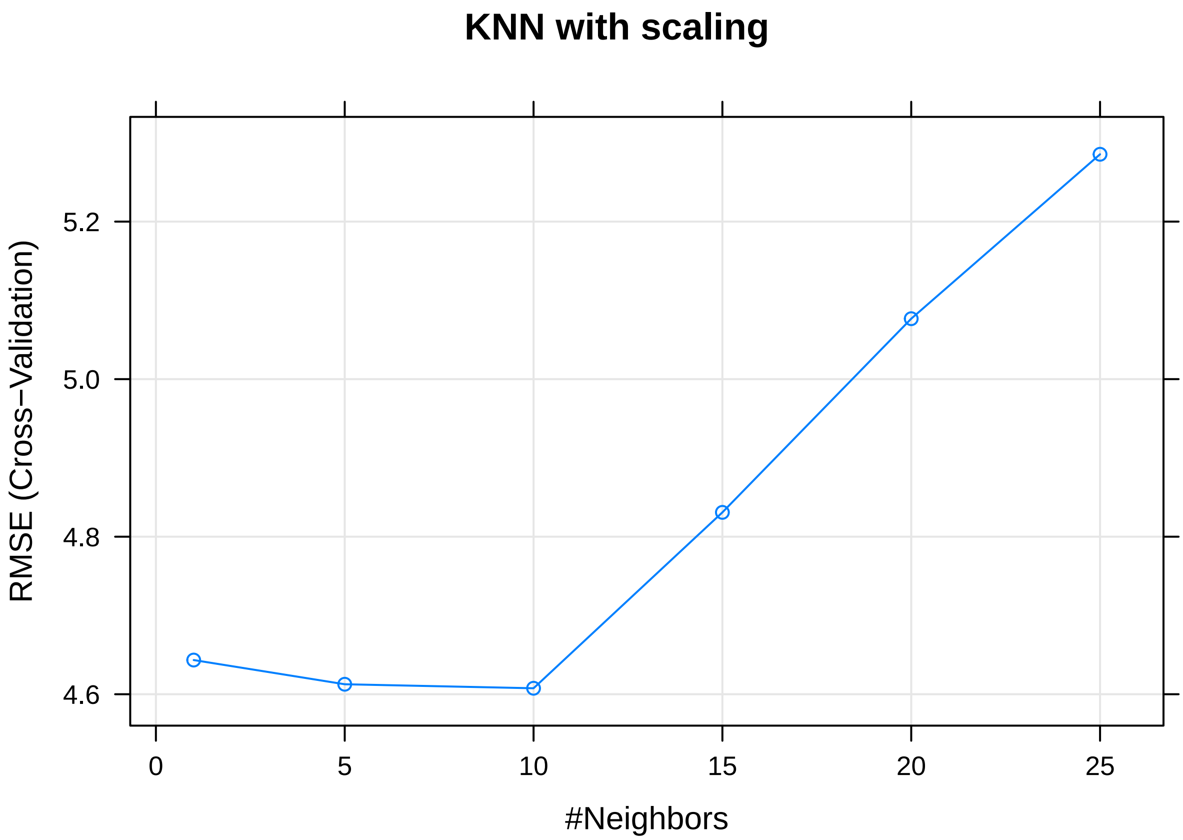
```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

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.

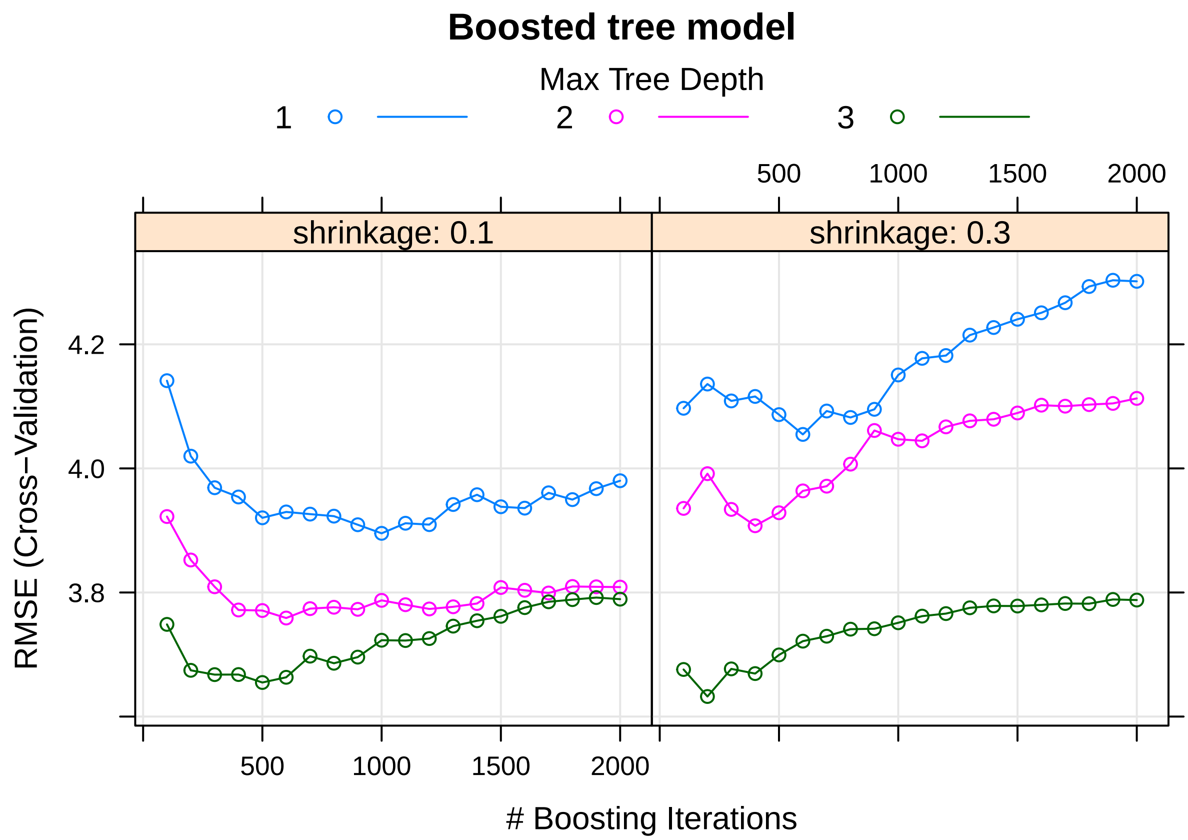
```r
plot(m2, main = 'KNN without scaling')
```




```r
plot(m3, main = 'KNN with scaling')
```




```r
plot(m5, main = 'Boosted tree model')
```




```r
calc_rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

```r
get_best_result = function(caret_fit) {
  best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
  best_result = caret_fit$results[best,]
  rownames(best_result) = NULL
  best_result
}
```

```r
cv1 = m1$results$RMSE
cv2 = get_best_result(m2)$RMSE
cv3 = get_best_result(m3)$RMSE
cv4 = get_best_result(m4)$RMSE
cv5 = get_best_result(m5)$RMSE
cross_validated_RMSE = c(cv1, cv2, cv3, cv4, cv5)
```

```r
r1 = calc_rmse(actual = bstn_tst$medv,
               predicted = predict(m1, bstn_tst))
r2 = calc_rmse(actual = bstn_tst$medv,
               predicted = predict(m2, bstn_tst))
r3 = calc_rmse(actual = bstn_tst$medv,
               predicted = predict(m3, bstn_tst))
r4 = calc_rmse(actual = bstn_tst$medv,
               predicted = predict(m4, bstn_tst))
r5 = calc_rmse(actual = bstn_tst$medv,
               predicted = predict(m5, bstn_tst))
```

```r
rmse = c(r1, r2, r3, r4, r5)
```

```r
models = c(
  'additive linear',
  'knn without scaling',
  'knn with scaling',
  'random forest',
  'boosted tree'
  )
  df = data.frame(models, cross_validated_RMSE, rmse)
knitr::kable(df)
```

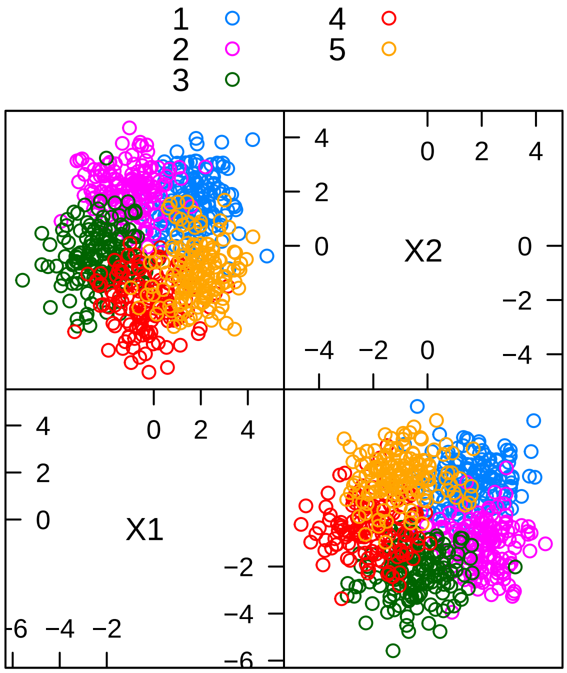
models	cross_validated_RMSE	rmse
additive linear	4.835444	4.989490
knn without scaling	6.146077	6.491325
knn with scaling	4.607600	5.460893
random forest	3.277205	3.033097
boosted tree	3.632349	3.666415

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.

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

```r
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

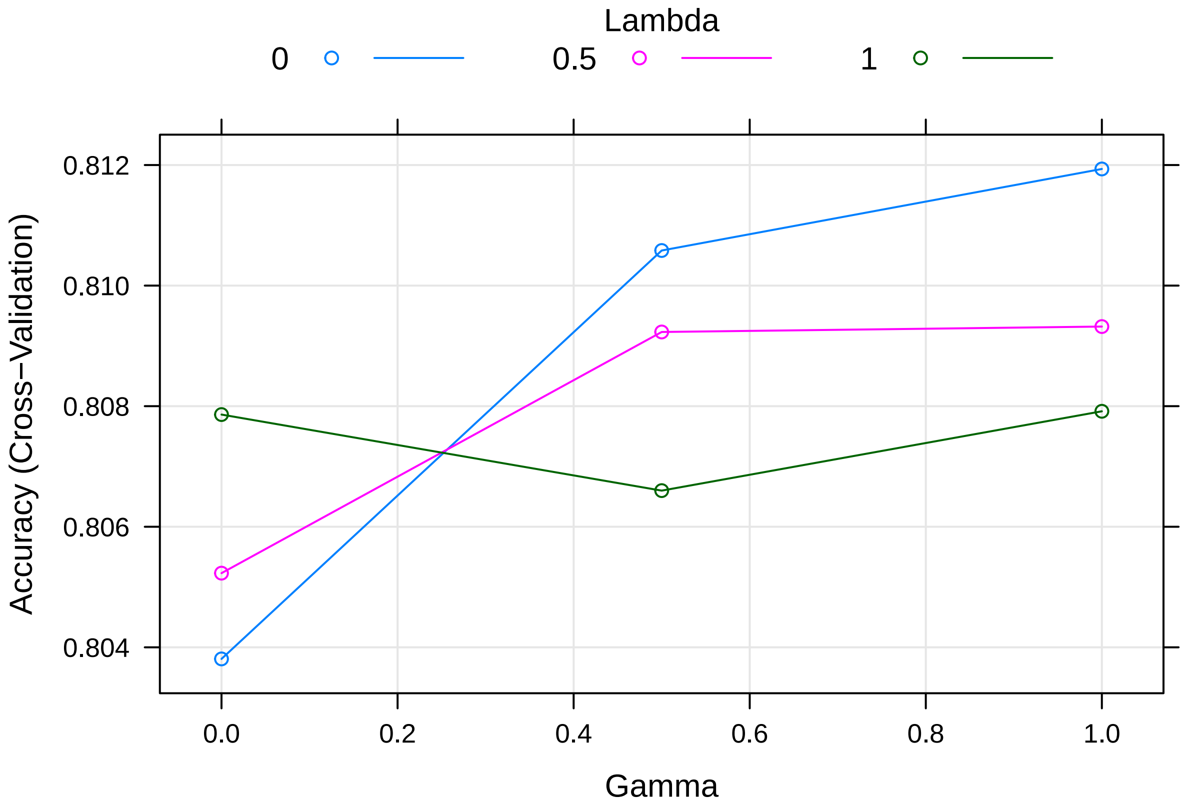
Provide a plot of acuracy versus tuning parameters for the RDA model. Also provide a table that summarizes the cross-validated accuracy and their standard deviations for each of the four (tuned) models.

```
library(klaR)
```

```
## Loading required package: MASS
```

```
set.seed(1337)
lda = train(classes ~., data = sim_trn, method = 'lda',
            trControl = trainControl(method = 'cv', number = 10))
set.seed(1337)
qda = train(classes ~., data = sim_trn, method = 'qda',
            trControl = trainControl(method = 'cv', number = 10))
set.seed(1337)
nb = train(classes ~., data = sim_trn, method = 'nb',
            trControl = trainControl(method = 'cv', number = 10))
set.seed(1337)
rda = train(classes ~., data = sim_trn, method = 'rda',
            trControl = trainControl(method = 'cv', number = 10))
```

```r
plot(rda)
```




```r
df2 = data.frame(
  models = c('LDA', 'QDA', 'Naive Bayes', 'RDA'),
  accuracy = c(
  lda$results$Accuracy,
  qda$results$Accuracy,
  get_best_result(nb)$Accuracy,
  get_best_result(rda)$Accuracy
  ),
  standard_deviation = c(
  lda$results$AccuracySD,
  qda$results$AccuracySD,
  get_best_result(nb)$AccuracySD,
  get_best_result(rda)$AccuracySD
  )
  )
  knitr::kable(df2)
```

models	accuracy	standard_deviation
LDA	0.8078610	0.0356863
QDA	0.8038074	0.0385323
Naive Bayes	0.8118103	0.0366514
RDA	0.8119350	0.0379171

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?

```r
get_best_result(m2)
```

```
##   k     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1 5 6.146078 0.5642096 4.230584 0.6478563 0.09510225 0.3854976
```

k = 5

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

```r
get_best_result(m3)
```

```
##    k   RMSE Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1 10 4.6076 0.766277 3.094923 0.7864472 0.05886789 0.4109533
```

K = 10

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

```r
m5$bestTune
```

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

shrinkage = 0.3, interaction depth = 3, n minobsinnode = 20, ntrees = 200

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

random forest

(e) Which method achieves the lowest test error?

random forest

Classification

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

```r
get_best_result(rda)
```

```
##   gamma lambda Accuracy     Kappa AccuracySD    KappaSD
## 1     1      0 0.811935 0.7644837 0.03791713 0.04758528
```

gamma = 1 and lambda = 0

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

LDA since the distribution of four classes are similar, the variances of each class does not seems to be varied too much, and they do not seems to be correlated.

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

Naive Bayes The four classes seems to be independent, so Naive Bayes would be better

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

```r
df0 = data.frame(
  lda$results$Accuracy,
  qda$results$Accuracy,
  get_best_result(nb)$Accuracy[1],
  get_best_result(rda)$Accuracy
  )
  colnames(df0) = c('lda', 'qda', 'navie bayes', 'rda')
  knitr::kable(df0)
```

lda	qda	navie bayes	rda
0.807861	0.8038074	0.8118103	0.811935

RDA

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

Yes, Since the accuracy of this model is the best, and the rda method intermediate the lda and qda method, which is a good choice based on the scatter plot.