HW7

Zixin Ouyang

10/30/2017

Exercise 1

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

```
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, ]
```

Exercise 1

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

```
set.seed(1337)
knn_mod = train(
  medv ~ .,
  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)
knn_mod2 = train(
  medv ~ .,
  data = bstn_trn,
  method = "knn",
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale"),
  tuneGrid = expand.grid(k = c(1,5,10,15,20,25))
)
```

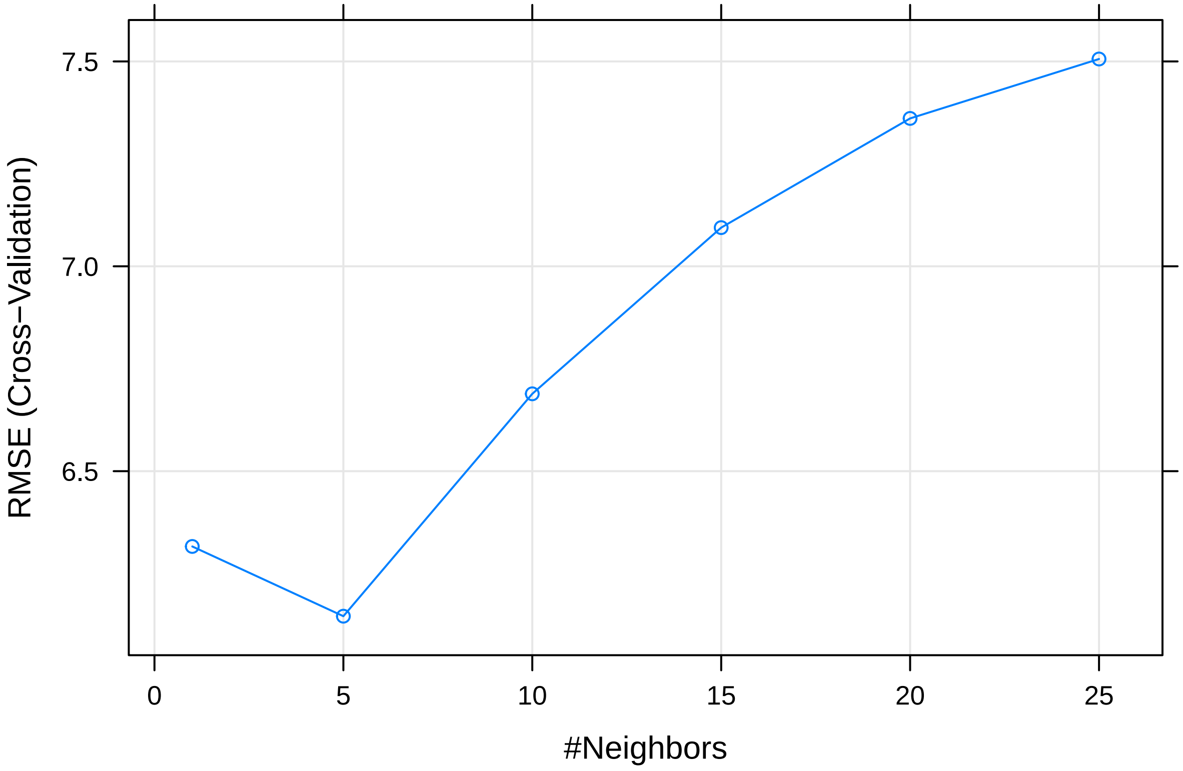
```
library(randomForest)
```

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

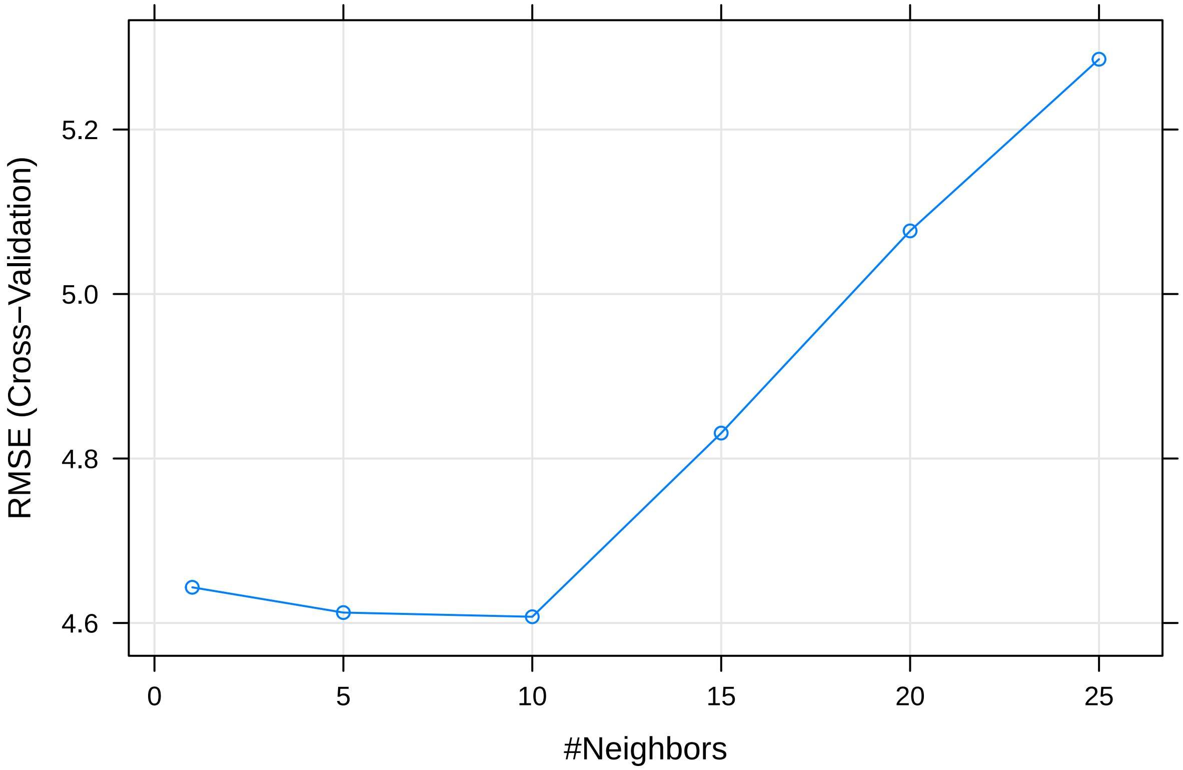
```
library(survival)
```

```
set.seed(1337)
gbm_grid = expand.grid(interaction.depth = c(1, 2, 3),
                       n.trees = (1:20) * 100,
                       shrinkage = c(0.1, 0.3),
                       n.minobsinnode = 20)
gbm_mod = train(
  medv ~ .,
  data = bstn_trn,
  trControl = trainControl(method = "cv", number = 5),
  method = "gbm",
  tuneGrid = gbm_grid,
  verbose = FALSE
)
```

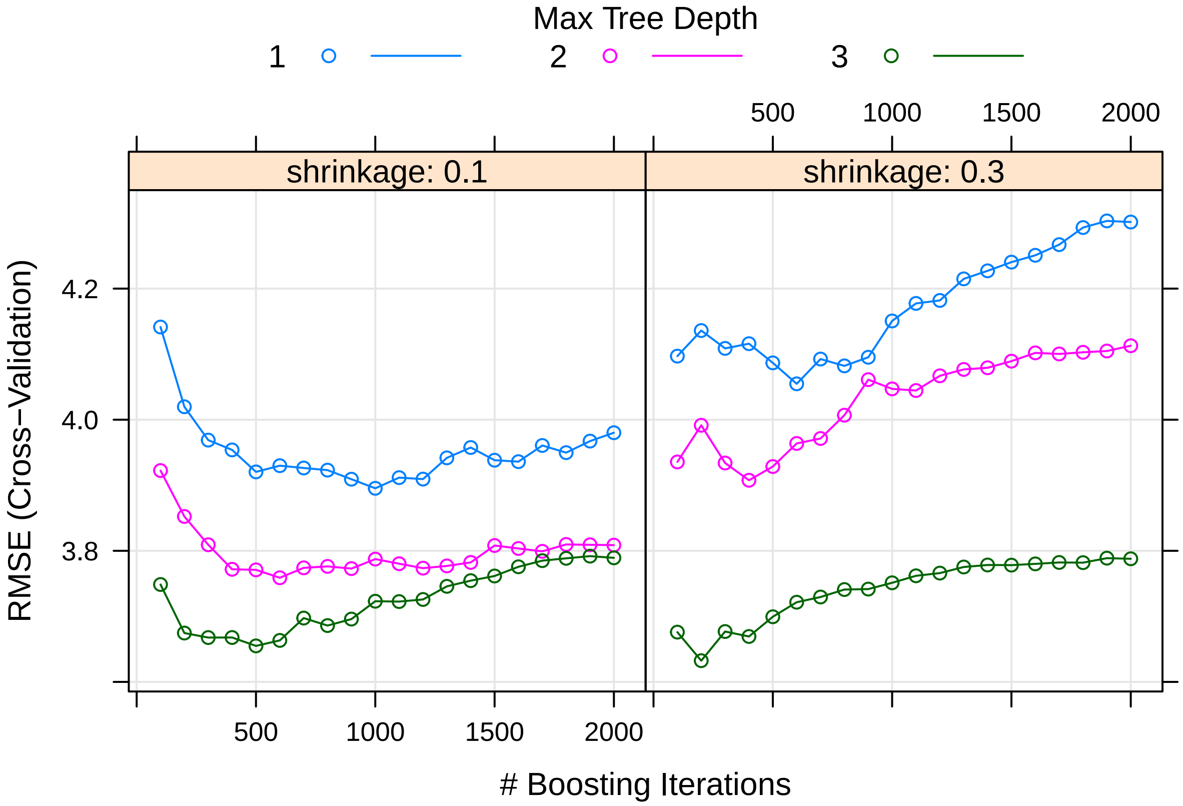
```
plot(knn_mod)
```



```
plot(knn_mod2)
```



```r
plot(gbm_mod)
```



```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
models = list(linear_mod, knn_mod, knn_mod2, rf_mod, gbm_mod)
```

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

```r
get_rmse = function(model, data, response) {
  rmse(actual = data[, response],
       predicted = predict(model, data))
}
```

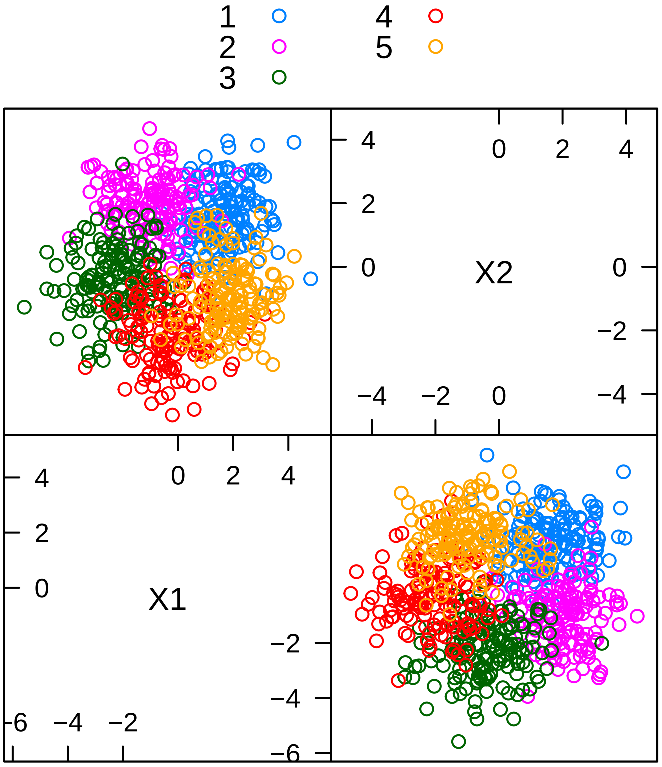
```r
test_rmse = sapply(models, get_rmse, data = bstn_tst, response = "medv")
```

Model	Cross-Validated RMSE	Test RMSE
`linear_mod`	4.8354435	4.98949
`knn_mod`	6.1460775	6.4913254
`knn_mod2`	4.6076	5.4608927
`rf_mod`	3.2772055	3.0330971
`gbm_mod`	3.6323487	3.6664155

Exercise 2

```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

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

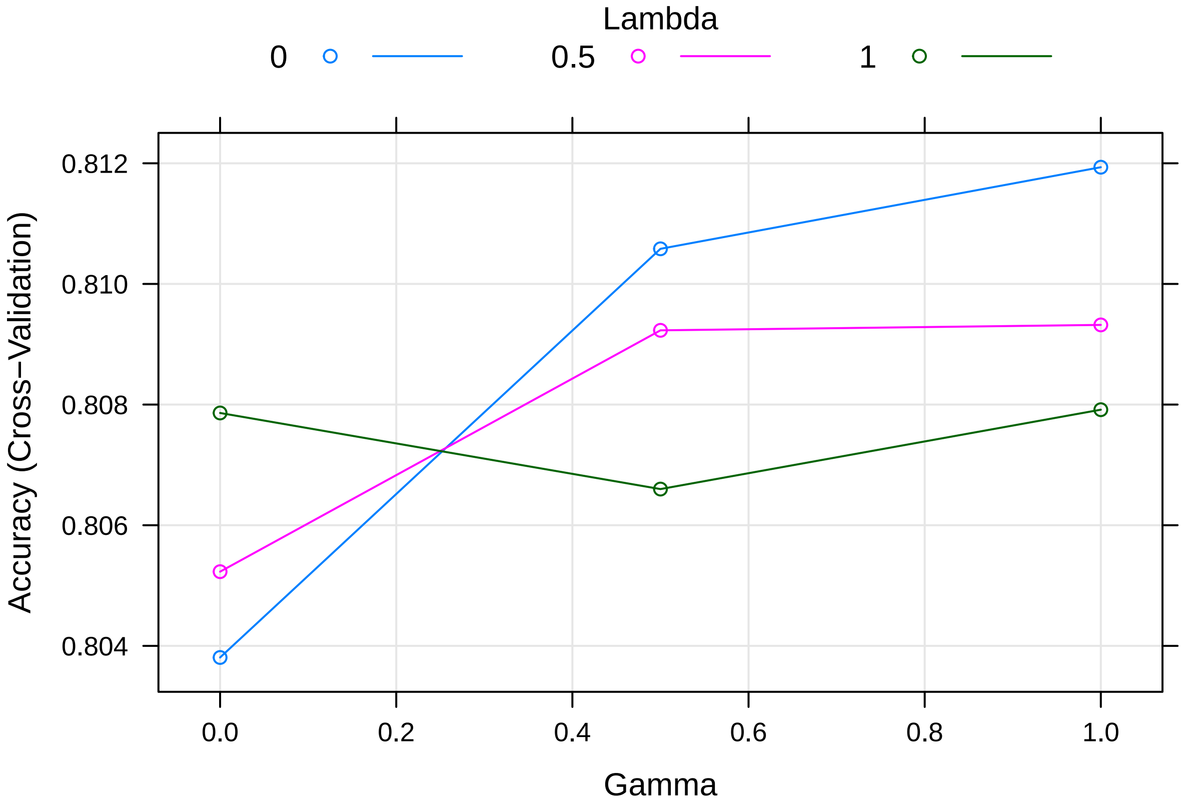
```
set.seed(1337)
QDA_mod = train(
  classes ~ .,
  data = sim_trn,
  method = "qda",
  trControl = trainControl(method = "cv", number = 10)
)
```

```
library(klaR)
```

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

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

```
plot(RDA_mod)
```



Model	Cross-Validated Accuracy	Standard Deviations
LDA_mod	0.807861	0.0356863
QDA_mod	0.8038074	0.0385323
nb_mod	0.8118103	0.0366514
RDA_mod	0.811935	0.0379171

Exercise 3

```
test_pred1 = predict(LDA_mod, newdata = sim_trn)
test_pred2 = predict(QDA_mod, newdata = sim_trn)
test_pred3 = predict(nb_mod, newdata = sim_trn)
test_pred4 = predict(RDA_mod, newdata = sim_trn)
```

```
calc_acc = function(actual, predicted) {
  mean(actual == predicted)
}
```

```
predicted_list = list(test_pred1, test_pred2, test_pred3, test_pred4)
train_acc = sapply(predicted_list, calc_acc, actual=sim_trn$classes)
```

```
train_acc
```

```
## [1] 0.8093333 0.8093333 0.8226667 0.8093333
```

Question	Answer
(a)	5
(b)	10
(c)	shrinkage: 0.3, interaction.depth: 3, n.minobsinnode: 20, n.trees: 200
(d)	The random forest method achieves the lowest cross-validated error
(e)	The random forest method achieves the lowest test error
(f)	gamma:1, lambda:0
(g)	LDA is better as the covariance appear to be very similar for different classes.
(h)	Naive Bayes is more appropriate because there is no clearly correlation between x1 and x2 in all classes.
(i)	The RDA model achieves the best cross-validated accuracy
(j)	No. The results of all the other models are within one SE. We should pick a less complex model, perhpas LDA.