DATA 624 Homework 9

Bin Lin

2018-5-11

8.4. Use a single predictor in the solubility data, such as the molecular weight or the number of carbon atoms and fit several models:

a. A simple regression tree

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(AppliedPredictiveModeling)
data("solubility")

sol_Train_X <- subset(solTrainXtrans, select = "NumCarbon")
sol_Train_Y <- solTrainY

sol_Test_X <- subset(solTestX, select = "NumCarbon")
sol_Test_Y <- solTestY
```

```
set.seed(888)
rpartTune  <- train(sol_Train_X, sol_Train_Y, method = "rpart2")
rpartTune
```

```
## CART
##
## 951 samples
##   1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 951, 951, 951, 951, 951, 951, ...
## Resampling results across tuning parameters:
##
##   maxdepth  RMSE      Rsquared   MAE
##   1         1.622600  0.3655507  1.261489
##   2         1.558405  0.4148207  1.212956
##   3         1.497246  0.4599550  1.171599
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was maxdepth = 3.
```

b. A random forest model

```
set.seed(888)
rfTune  <- train(sol_Train_X, sol_Train_Y, method = "rf", tuneLength = 1)
rfTune
```

```
## Random Forest
##
## 951 samples
##   1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 951, 951, 951, 951, 951, 951, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   1.499159  0.4599305  1.171506
##
## Tuning parameter 'mtry' was held constant at a value of 1
```

c. Different Cubist models with a single rule or multiple committees (each with and without using neighbor adjustments)

RMSE was used to select the optimal model using the smallest value. The final values used for the model were committees = 1 and neighbors = 0.

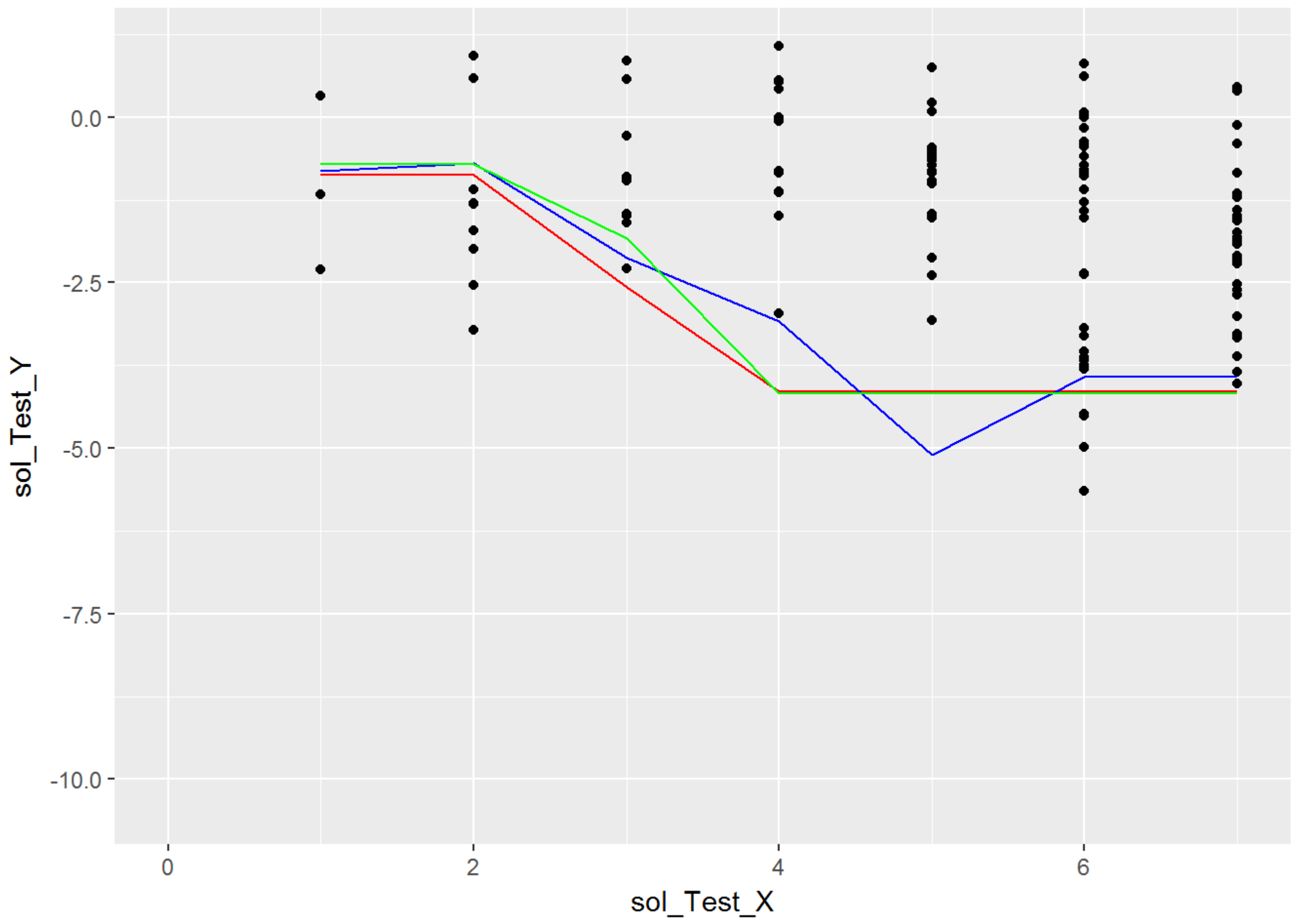
```
set.seed(888)
grid <- expand.grid(committees = c(1, 10, 50, 100), neighbors = c(0, 1, 5, 9))

cubistTune <- train(sol_Train_X, sol_Train_Y, method = "cubist",  tuneGrid = grid)
cubistTune
```

```
## Cubist
##
## 951 samples
##   1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 951, 951, 951, 951, 951, 951, ...
## Resampling results across tuning parameters:
##
##   committees  neighbors  RMSE      Rsquared   MAE
##     1         0          1.513696  0.4598572  1.159236
##     1         1          1.591658  0.4297735  1.235227
##     1         5          1.590905  0.4301010  1.234767
##     1         9          1.588607  0.4310926  1.233317
##    10         0          1.523531  0.4536249  1.164063
##    10         1          1.592666  0.4292480  1.235701
##    10         5          1.591908  0.4295842  1.235233
##    10         9          1.589577  0.4306198  1.233813
##    50         0          1.519819  0.4558860  1.162366
##    50         1          1.592385  0.4293843  1.235568
##    50         5          1.591627  0.4297223  1.235095
##    50         9          1.589272  0.4307587  1.233646
##   100         0          1.518559  0.4565576  1.162328
##   100         1          1.592505  0.4293628  1.235732
##   100         5          1.591751  0.4296976  1.235265
##   100         9          1.589388  0.4307384  1.233822
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were committees = 1 and neighbors = 0.
```

Plot the predictor data versus the solubility results for the test set. Overlay the model predictions for the test set. How do the model differ? Does changing the tuning parameter(s) significantly affect the model fit?

```r
library(ggplot2)

rpartPred <- predict(rpartTune, newdata = sol_Test_X)
rfPred <- predict(rfTune, newdata = sol_Test_X)
cubistPred <- predict(cubistTune, newdata = sol_Test_X)

test <- cbind(sol_Test_X, sol_Test_Y, simple_regression = rpartPred, random_forest = rfPred, cub
ist = cubistPred)

ggplot(data = test, aes(x = sol_Test_X, y = sol_Test_Y)) +
  geom_point() +
  xlim(0, 7) +
  geom_line(aes(y = rpartPred), color = "red") +
  geom_line(aes(y = rfPred), color = "blue") +
  geom_line(aes(y = cubistPred), color = "green")
```

```
## Warning: Removed 195 rows containing missing values (geom_point).
## Warning: Removed 195 rows containing missing values (geom_path).

## Warning: Removed 195 rows containing missing values (geom_path).

## Warning: Removed 195 rows containing missing values (geom_path).
```



8.5. Fit different tree- and rule-based models for the Tecator data discussed in Exercise 6.1. How do they compare to linear models? Do the between predictor correlations seem to affect your models? If so, how would you transform or re-encode the predictor data to mitigate this issue?

```r
library(caret)
data(tecator)
set.seed(888)

endpoints1 <- as.data.frame(endpoints)
absorp1 <- as.data.frame(absorp)


trainingRows <- createDataPartition(endpoints1[, 2], p = 0.8, list= FALSE)
endpoints_train <- endpoints1[trainingRows, 2]
endpoints_test <- endpoints1[-trainingRows, 2]

absorp_train <- absorp1[trainingRows, ]
absorp_test <- absorp1[-trainingRows, ]
```

```
set.seed(888)

ctrl <- trainControl(method = "cv", preProc = c("center", "scale"))

plsTune2  <- train(x = absorp_train, y = endpoints_train, method = "pls", trControl = ctrl)
rpartTune2  <- train(x = absorp_train, y = endpoints_train, method = "rpart2", trControl = ctrl)
rfTune2  <- train(x = absorp_train, y = endpoints_train, method = "rf", trControl = ctrl)

gbmGrid <- expand.grid(interaction.depth = seq(1, 7, by = 2), n.trees = seq(100, 1000, by = 50),
shrinkage = c(0.01, 0.1), n.minobsinnode = 10)
gbmTune2 <- train(x = absorp_train, y = endpoints_train, method = "gbm", tuneGrid = gbmGrid, ver
bose = FALSE)

cubistTune2 <- train(x = absorp_train, y = endpoints_train, method = "cubist")
```

```
pls_RMSE2 <- min(plsTune2$results$RMSE)
rpart_RMSE2 <- min(rpartTune2$results$RMSE)
rf_RMSE2 <- min(rfTune2$results$RMSE)
gbm_RMSE2 <- min(gbmTune2$results$RMSE)
cubist_RMSE2 <- min(cubistTune2$results$RMSE)

result2 <- data.frame(Model = c("pls", "rpart", "rf", "gbm", "cubist"), RMSE = c(pls_RMSE2, rpar
t_RMSE2, rf_RMSE2, gbm_RMSE2, cubist_RMSE2))

result2
```

```
##    Model      RMSE
## 1    pls  5.213722
## 2  rpart 10.551262
## 3     rf  7.298284
## 4    gbm  7.421699
## 5 cubist  2.914658
```

Interpretation: Based on RMSE that is generated from each model, we can tell that the Cubist model has the best performance. Among all the models, ramdon forest model is one of the worst along with single regression trees model. From some previous exercise, we know the predictor correlation will affect the models. The predictors that are highly correlated to each other will receive same level of importance, while decreasing other predictors' importance down. Therefore, the best practice is to eliminate highly correlated predictors.

8.6. Return to the permeability problem described in Exercises 6.2 and 7.4. Train several tree-based models and evaluate the resampling and test set performance:

a. Which tree-based model gives the optimal resampling and test set performance?

Ramdon forest model provides the optimal resampling and test set performance, because it has the lowest RMSE value.

```
data(permeability)
set.seed(888)

fingerprints1 <- as.data.frame(fingerprints)
permeability1 <- as.data.frame(permeability)

fingerprints1 <- fingerprints1[, -nearZeroVar(fingerprints1)]

trainingRows <- createDataPartition(permeability1$permeability, p = 0.8, list= FALSE)
fingerprints_train <- fingerprints1[trainingRows,]
fingerprints_test <- fingerprints1[-trainingRows,]

permeability_train <- permeability1[trainingRows, ]
permeability_test <- permeability1[-trainingRows, ]
```

```
set.seed(888)

ctrl <- trainControl(method = "cv", preProc = c("center", "scale"))

plsTune3  <- train(x = fingerprints_train, y = permeability_train, method = "pls", trControl = c
trl)
rpartTune3  <- train(x = fingerprints_train, y = permeability_train, method = "rpart2", trContro
l = ctrl)
rfTune3  <- train(x = fingerprints_train, y = permeability_train, method = "rf", trControl = ctr
l)

gbmGrid <- expand.grid(interaction.depth = seq(1, 7, by = 2), n.trees = seq(100, 1000, by = 50),
shrinkage = c(0.01, 0.1), n.minobsinnode = 10)
gbmTune3 <- train(x = fingerprints_train, y = permeability_train, method = "gbm", tuneGrid = gbm
Grid, verbose = FALSE)

cubistTune3 <- train(x = fingerprints_train, y = permeability_train, method = "cubist")
```

```
pls_RMSE3 <- min(plsTune3$results$RMSE)
rpart_RMSE3 <- min(rpartTune3$results$RMSE)
rf_RMSE3 <- min(rfTune3$results$RMSE)
gbm_RMSE3 <- min(gbmTune3$results$RMSE)
cubist_RMSE3 <- min(cubistTune3$results$RMSE)

result3 <- data.frame(Model = c("pls", "rpart", "rf", "gbm", "cubist"), RMSE = c(pls_RMSE3, rpar
t_RMSE3, rf_RMSE3, gbm_RMSE3, cubist_RMSE3))

result3
```

```
##    Model     RMSE
## 1    pls 11.51701
## 2  rpart 10.82113
## 3     rf 11.01250
## 4    gbm 11.00841
## 5 cubist 11.94154
```

b. Do any of these models outperform the covariance or non-covariance based regression models you have previously developed for these data? What criteria did you use to compare models' performance?

None of these models outperform the covariance or non-covariance based regression models than SVM radial model. I am using RMSE to compare modesl' performance.

```
set.seed(888)
ctrl <- trainControl(method = "cv", preProc = c("center", "scale", "knnImpute"))

knnTune3 <- train(x = fingerprints_train, y = permeability_train, method = "knn", tuneLength = 1
0)

marsGrid <- expand.grid(degree = 1:2, nprune = 1:20)
marsTune3 <- train(x = fingerprints_train, y = permeability_train, method = "earth", tuneGrid =
 marsGrid, trControl = ctrl)
```

```
## Loading required package: earth
```

```
## Loading required package: plotmo
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
```

```
svmRTune3 <- train(x = fingerprints_train, y = permeability_train, method = "svmRadial", tuneLen
gth = 20, trControl = ctrl)
```

```
knn_RMSE3 <- min(knnTune3$results$RMSE)
mars_RMSE3 <- min(marsTune3$results$RMSE)
svmR_RMSE3 <- min(svmRTune3$results$RMSE)

result4 <- data.frame(Model = c("knn", "mars", "svmR"), RMSE = c(knn_RMSE3, mars_RMSE3, svmR_RMS
E3))

result4
```

```
##   Model     RMSE
## 1   knn 12.75464
## 2  mars 10.80341
## 3  svmR  9.85278
```

c. Of all the models you have developed thus far, which, if any, would you recommend to replace the permeability laboratory experiment?

I would recommend SVM model with svmRadial method, because it generate lowest RMSEvalue.