Homework 7

Applied Predictive Modeling - Linear Regression and Its Cousins

Paul Perez

4/18/2021

Exercise 6.2

Developing a model to predict permeability (see Sect. 1.4) could save significant resources for a pharmaceutical company, while at the same time more rapidly identifying molecules that have a sufficient permeability to become a drug:

(a) Start R and use these commands to load the data:

The matrix `fingerprints` contains the 1,107 binary molecular predictors for the 165 compounds, while permeability contains permeability response.

(b) The fingerprint predictors indicate the presence or absence of substructures of a molecule and are often sparese meaning that relatively few of the molecules contain each substructure. Filter out the predictors that have low frequencies using `nearZeroVar` function from the caret package. How many predictors are left for modeling?

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
nzv <- nearZeroVar(fingerprints)
not.nzv <- fingerprints[, -nzv]
ncol(not.nzv)
```

```
## [1] 388
```

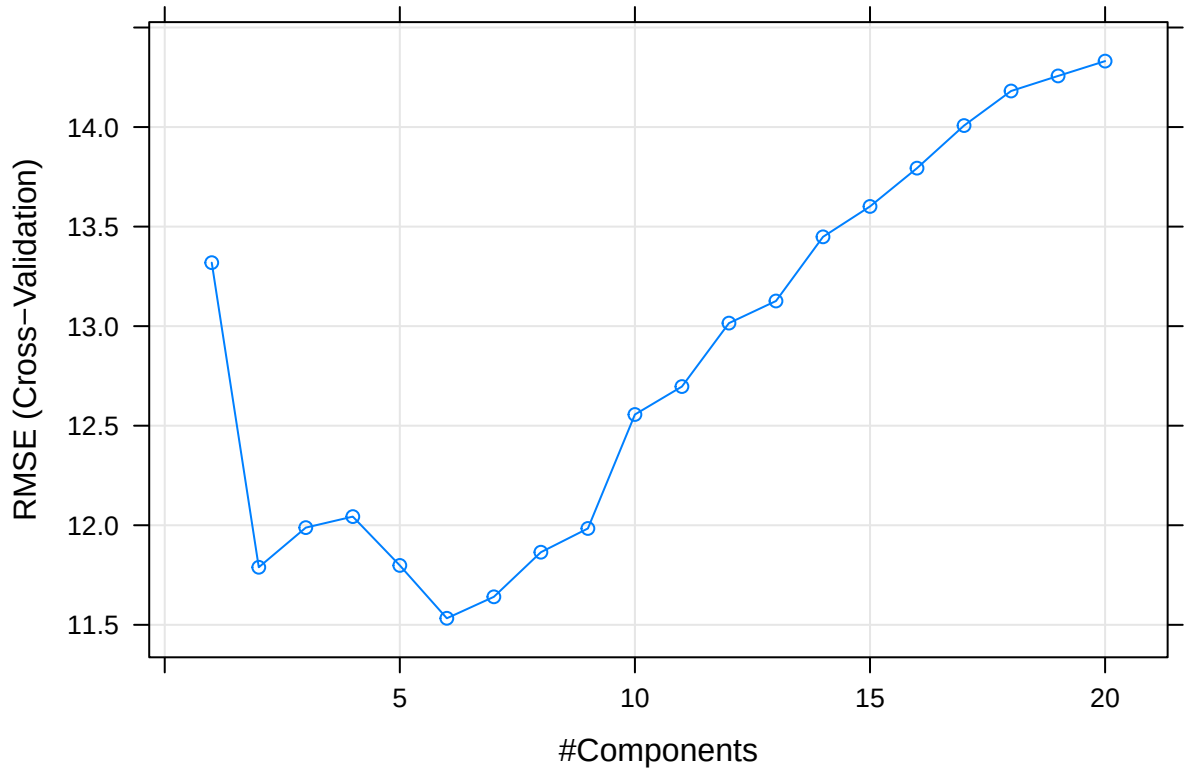
(c) Split the data into a training and test set, pre-process the data, and tune a PLS model. How many latent variables are optimal and what is the corresponding resampled estimate of R^2?

```r
set.seed(123)
split <- createDataPartition(permeability, p = 0.8, list = FALSE, times = 1)

Xtrain.data  <- not.nzv[split, ] #fingerprints train
xtest.data <- not.nzv[-split, ] #fingerprints test
Ytrain.data  <- permeability[split, ] #permability train
ytest.data <- permeability[-split, ] #permability test
```

```r
ctrl <- trainControl(method = "cv", number = 10)
pls.mod <- train(x = Xtrain.data,
                 y = Ytrain.data, method = "pls",
                 tuneLength = 20,
                 trControl = ctrl,
                 preProc = c("center", "scale"))
pls.mod
```

```
## Partial Least Squares
##
## 133 samples
## 388 predictors
##
## Pre-processing: centered (388), scaled (388)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 121, 121, 118, 119, 119, 119, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1     13.31894  0.3442124  10.254018
##    2     11.78898  0.4830504   8.534741
##    3     11.98818  0.4792649   9.219285
##    4     12.04349  0.4923322   9.448926
##    5     11.79823  0.5193195   9.049121
##    6     11.53275  0.5335956   8.658301
##    7     11.64053  0.5229621   8.878265
##    8     11.86459  0.5144801   9.265252
##    9     11.98385  0.5188205   9.218594
##   10     12.55634  0.4808614   9.610747
##   11     12.69674  0.4758068   9.702325
##   12     13.01534  0.4538906   9.956623
##   13     13.12637  0.4367362   9.878017
##   14     13.44865  0.4140715  10.065088
##   15     13.60135  0.4034269  10.188150
##   16     13.79361  0.3943904  10.247160
##   17     14.00756  0.3845119  10.412776
##   18     14.18113  0.3711378  10.587027
##   19     14.25674  0.3703610  10.575726
##   20     14.33121  0.3723176  10.679764
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 6.
```

```r
plot(pls.mod)
```




```r
pls.mod$bestTune
```

```
##   ncomp
## 6     6
```

```r
summary(pls.mod$finalModel)
```

```
## Data:    X dimension: 133 388
##  Y dimension: 133 1
## Fit method: oscorespls
## Number of components considered: 6
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X           22.98    34.61    40.51    46.13    53.69    58.12
## .outcome    33.73    55.03    61.84    67.77    71.65    75.73
```

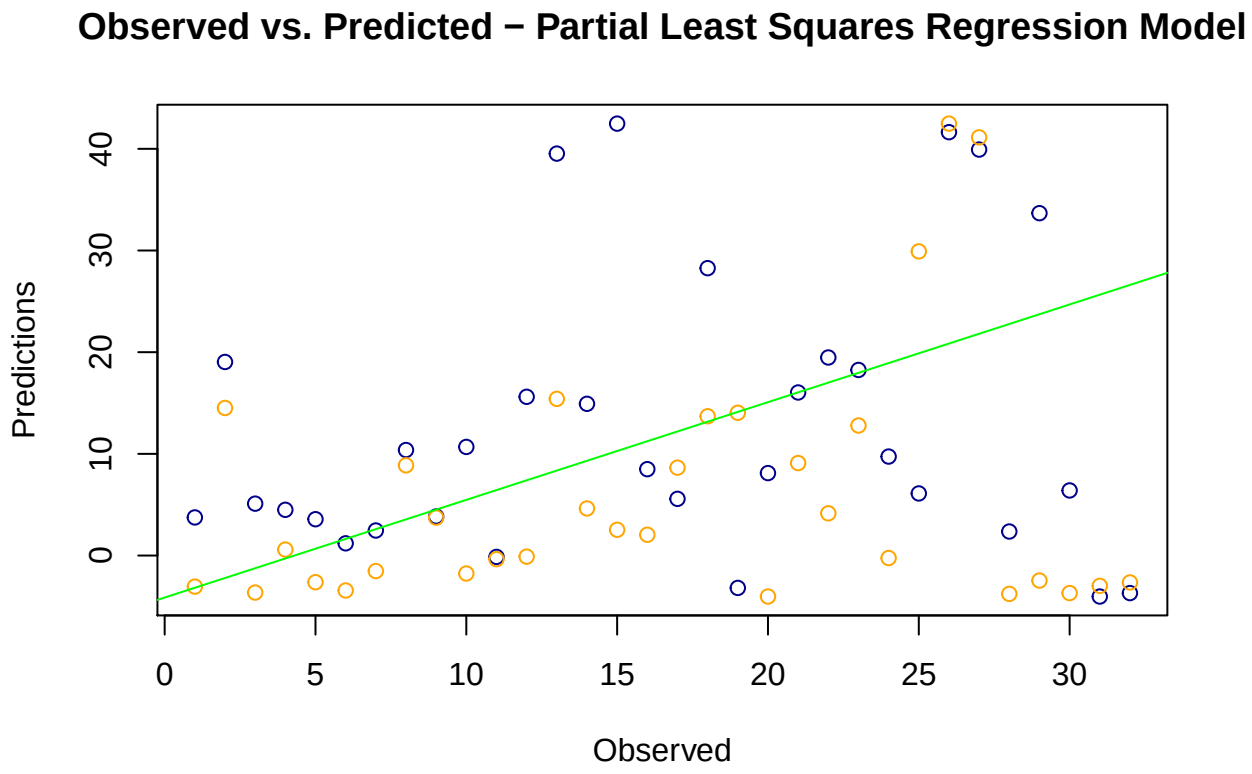
The best tuning parameter is 7 which minimizes the cross validation error, that is, the best estimate for the test error of model.

(d) Predict the response for the test set. What is the test set estimate of R^2?

```
predictions <- predict(pls.mod, xtest.data)
cbind(RMSE = RMSE(predictions, ytest.data),  R_squared = caret::R2(predictions, ytest.data))
```

```
##          RMSE R_squared
## [1,] 12.34869 0.3244542
```

```
plot(predictions,
     col = "darkblue",
     main = "Observed vs. Predicted - Partial Least Squares Regression Model",
     xlab = "",
     ylab = "Predictions")

par(new = TRUE)

plot(ytest.data,
     col = "orange",
     axes=F,
     ylab = "",
     xlab="Observed")

abline(0, 1, col='green')
```



(e) Try building other models discussed in this chapter. Do any have better predictive performance?

We can perform a similar predition with both Ridge and Lasso regression models.

Ridge Model

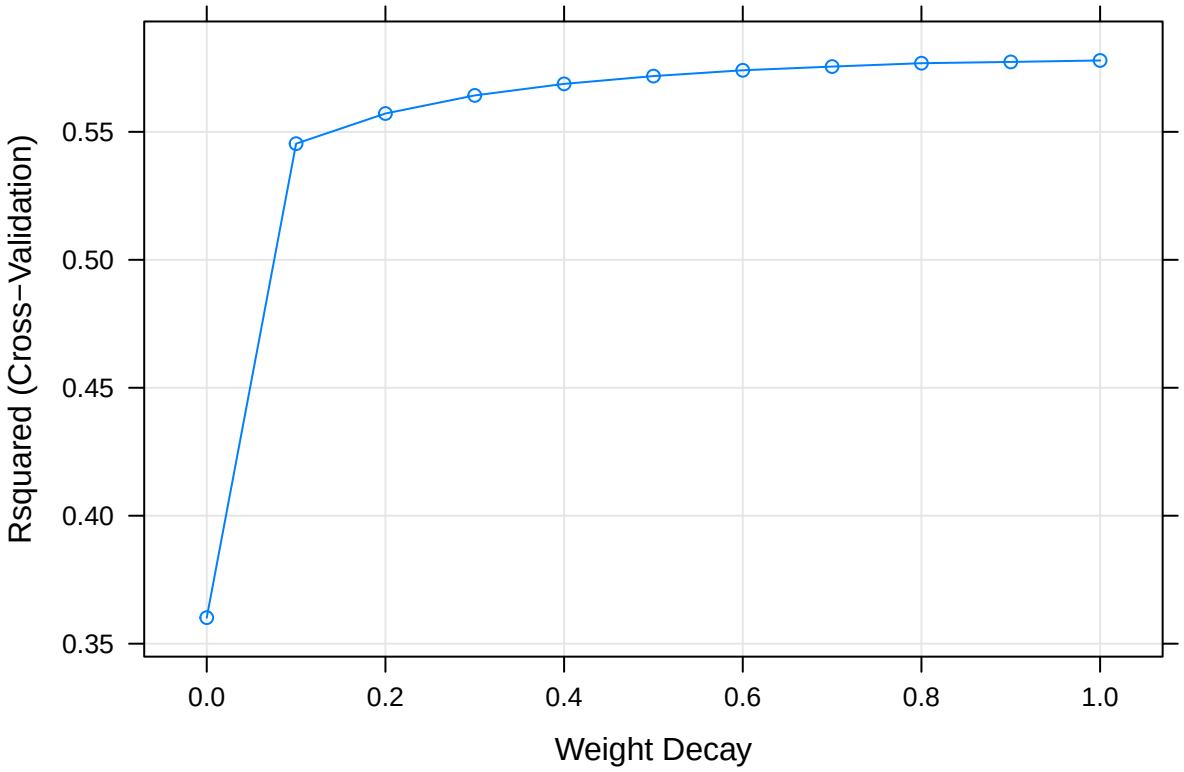
```
set.seed(123)
ridge.mod <- train(x=Xtrain.data,
                   y=Ytrain.data,
                   method='ridge',
                   metric='Rsquared',
                   tuneGrid=data.frame(.lambda = seq(0, 1, by=0.1)),
                   trControl=trainControl(method='cv'),
                   preProcess=c('center','scale')
                   )
```

```
## Warning: model fit failed for Fold02: lambda=0.0 Error in if (zmin < gamhat) { : missing value where

## Warning: model fit failed for Fold09: lambda=0.0 Error in if (zmin < gamhat) { : missing value where

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

```
ridge.mod
```

```
## Ridge Regression
##
## 133 samples
## 388 predictors
##
## Pre-processing: centered (388), scaled (388)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 119, 118, 120, 121, 119, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE      Rsquared   MAE
##   0.0     16.20635  0.3601753  11.220501
##   0.1     12.06264  0.5453869   9.100052
##   0.2     12.10944  0.5571793   9.168267
##   0.3     12.37240  0.5642431   9.388516
##   0.4     12.76763  0.5687514   9.760973
##   0.5     13.26055  0.5717972  10.174632
##   0.6     13.82796  0.5740629  10.616896
##   0.7     14.45460  0.5755140  11.137567
##   0.8     15.12637  0.5768246  11.742376
##   0.9     15.83746  0.5773256  12.371193
##   1.0     16.57763  0.5778968  13.011360
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was lambda = 1.
```

```r
plot(ridge.mod)
```




```r
ridge.mod$bestTune
```

```
##    lambda
## 11      1
```

```r
summary(ridge.mod$finalModel)
```

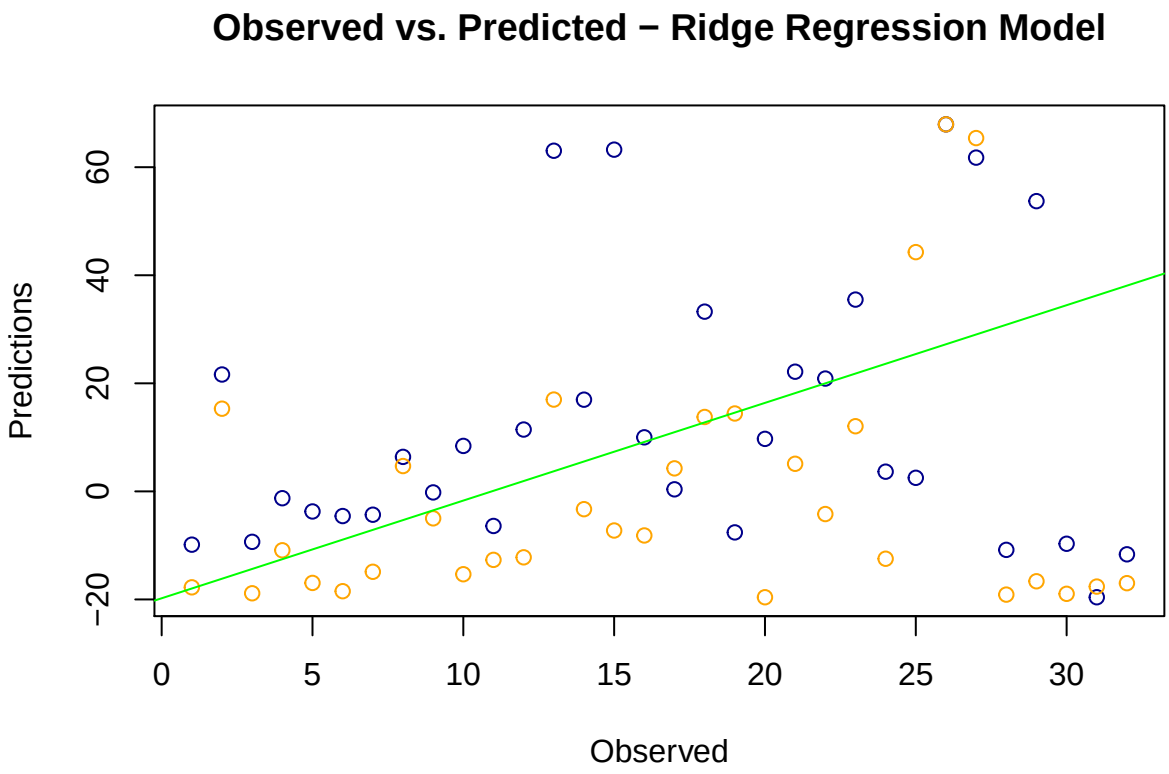
```
##             Length Class      Mode
## call             4 -none-     call
## actions        153 -none-     list
## allset         388 -none-     numeric
## beta.pure    59364 -none-     numeric
## vn             388 -none-     character
## mu               1 -none-     numeric
## normx          388 -none-     numeric
## meanx          388 -none-     numeric
## lambda           1 -none-     numeric
## L1norm         153 -none-     numeric
## penalty        153 -none-     numeric
## df             153 -none-     numeric
## Cp             153 -none-     numeric
## sigma2           1 -none-     numeric
```

```
## xNames        388  -none-     character
## problemType     1  -none-     character
## tuneValue       1  data.frame list
## obsLevels       1  -none-     logical
## param           0  -none-     list
```

```
predictions <- predict(ridge.mod, xtest.data)
cbind(RMSE = RMSE(predictions, ytest.data),  R.squared = caret::R2(predictions, ytest.data))
```

```
##         RMSE R.squared
## [1,] 19.9078 0.3695863
```

```
plot(predictions,
     col = "darkblue",
     main = "Observed vs. Predicted - Ridge Regression Model",
     xlab = "",
     ylab = "Predictions")

par(new = TRUE)

plot(ytest.data,
     col = "orange",
     axes=F,
     ylab = "",
     xlab="Observed")

abline(0, 1, col='green')
```


Observed vs. Predicted − Ridge Regression Model
Predictions
−20
0
20
40
60
0
5
10
15
20
25
30
Observed

Lasso Model

```
set.seed(123)
lasso.mod <- train(x=Xtrain.data,
                   y=Ytrain.data,
                   method='lasso',
                   metric='Rsquared',
                   tuneGrid=data.frame(.fraction = seq(0, 0.5, by=0.05)),
                   trControl=trainControl(method='cv'),
                   preProcess=c('center','scale')
                   )
```

```
## Warning: model fit failed for Fold02: fraction=0.5 Error in if (zmin < gamhat) { : missing value whe
```

```
## Warning: model fit failed for Fold09: fraction=0.5 Error in if (zmin < gamhat) { : missing value whe
```

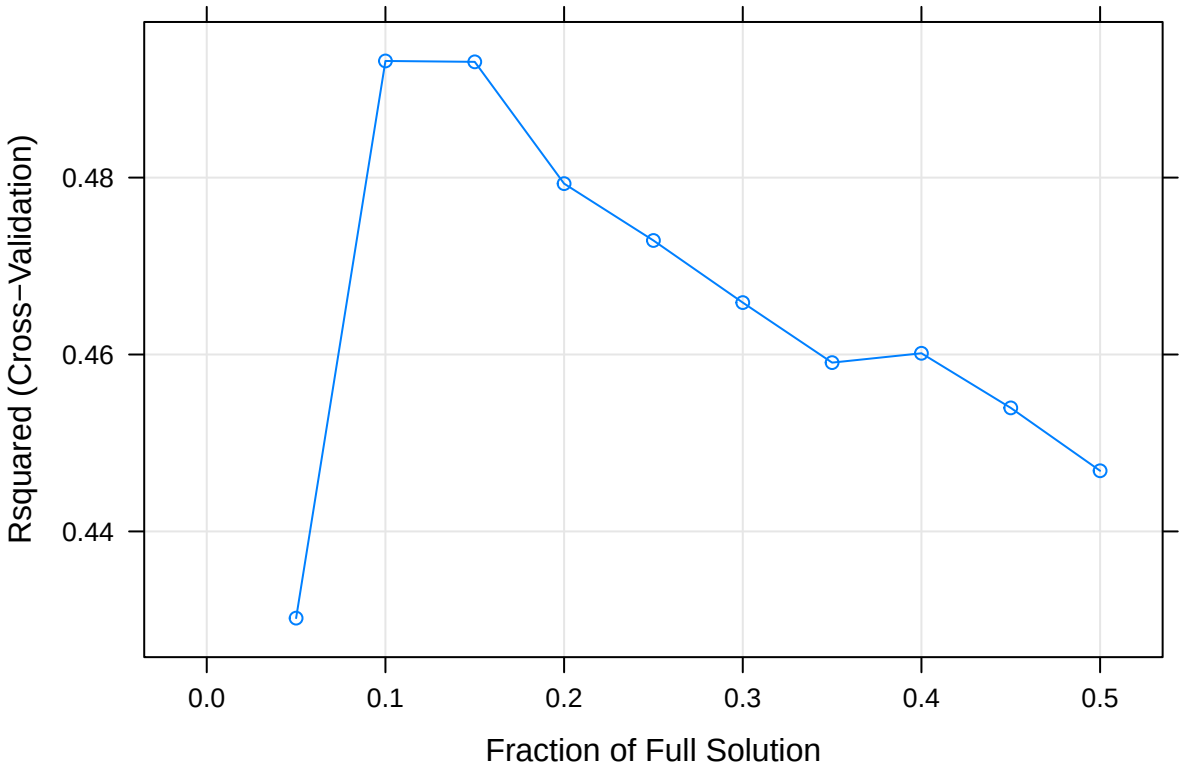
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
## Warning in train.default(x = Xtrain.data, y = Ytrain.data, method = "lasso", :
## missing values found in aggregated results
```

```
lasso.mod
```

```
## The lasso
##
## 133 samples
## 388 predictors
##
## Pre-processing: centered (388), scaled (388)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 120, 119, 118, 120, 121, 119, ...
## Resampling results across tuning parameters:
##
##   fraction  RMSE      Rsquared   MAE
##   0.00      16.00688        NaN  12.917824
##   0.05      12.97787  0.4301959   9.877733
##   0.10      11.80785  0.4931852   9.107986
##   0.15      11.66393  0.4930871   8.897928
##   0.20      11.86820  0.4793179   9.030145
##   0.25      12.10947  0.4728912   9.253825
##   0.30      12.34485  0.4658690   9.393513
##   0.35      12.61952  0.4590835   9.612879
##   0.40      12.75663  0.4601371   9.686790
##   0.45      12.96171  0.4539654   9.774599
##   0.50      13.22572  0.4468568   9.900789
##
## Rsquared was used to select the optimal model using the largest value.
## The final value used for the model was fraction = 0.1.
```

```
plot(lasso.mod)
```



Model had some issue when fitting in some validation folds.

```
lasso.mod$bestTune
```

```
##   fraction
## 3      0.1
```

```
summary(lasso.mod$finalModel)
```

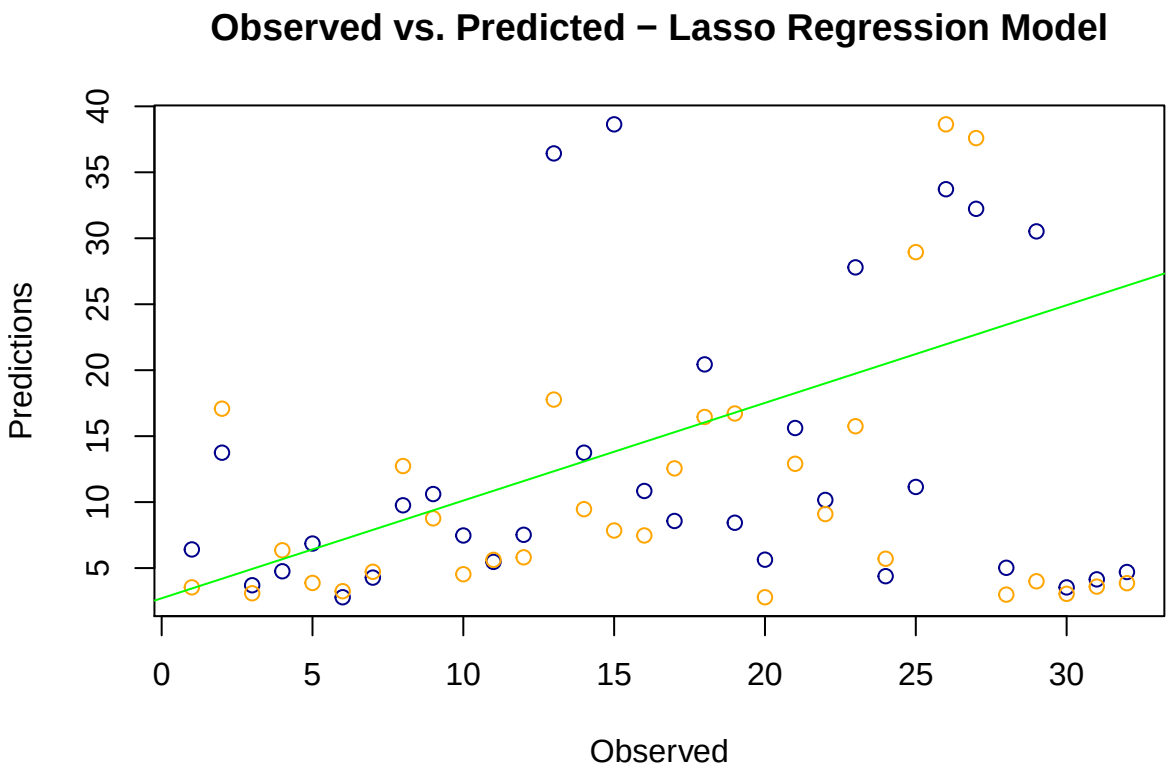
```
##             Length Class      Mode
## call           4   -none-     call
## actions      193   -none-     list
## allset       162   -none-     numeric
## beta.pure  31266   -none-     numeric
## vn           388   -none-     character
## mu             1   -none-     numeric
## normx        162   -none-     numeric
## meanx        162   -none-     numeric
## lambda         1   -none-     numeric
## L1norm       193   -none-     numeric
## penalty      193   -none-     numeric
## df           193   -none-     numeric
```

```
## Cp            193  -none-     numeric
## sigma2          1  -none-     numeric
## xNames        388  -none-     character
## problemType     1  -none-     character
## tuneValue       1  data.frame list
## obsLevels       1  -none-     logical
## param           0  -none-     list
```

```
predictions <- predict(lasso.mod, xtest.data)
cbind(RMSE = RMSE(predictions, ytest.data), R.squared = caret::R2(predictions, ytest.data))
```

```
##          RMSE R.squared
## [1,] 10.68287 0.3661212
```

```
plot(predictions,
     col = "darkblue",
     main = "Observed vs. Predicted - Lasso Regression Model",
     xlab = "",
     ylab = "Predictions")

par(new = TRUE)

plot(ytest.data,
     col = "orange",
     axes=F,
     ylab = "",
     xlab="Observed")

abline(0, 1, col='green')
```



(f) Would you recommend any of your models to replace the permeability laboratory experiment?

Exercise 6.3

A chemical manufacturing process for a pharmaceutical product was discussed in Sect. 1.4. In this problem, the objective is to understand the relationship between biological measurements of the raw materials (predictors), measurements of the manufacturing process (predictors), and the response of product yield. Biological predictors cannot be changed but can be used to assess the quality of the raw material before processing. On the other hand, manufacturing process predictors can be changed in the manufacturing process. Improving product yield by 1% will boost revenue by approximately one hundred thousand dollars per batch:

(a) Start R and use these commands to load the data:

```
library(AppliedPredictiveModeling)
data("ChemicalManufacturingProcess")
```

The matrix processPredictors contains the 57 predictors (12 describing the input biological material and 45 describing the process predictors) for the 176 manufacturing runs. `yield` contains the percent yield for each run.

(b) A small percentage of cells in the predictor set contain missing values. Use an imputation function to fill in the missing values (e.g., See Sect. 3.8)

```
#preprocess data excluding the yeild column
library(RANN)
```

```
## Warning: package 'RANN' was built under R version 3.6.3
```

```
preprocessing <- preProcess(ChemicalManufacturingProcess[,-1], method = c("center", "scale", "knnImpute
Xpreprocess <- predict(preprocessing, ChemicalManufacturingProcess[,-1])
#missmap(Xpreprocess, col = c("yellow", "navy"))
```

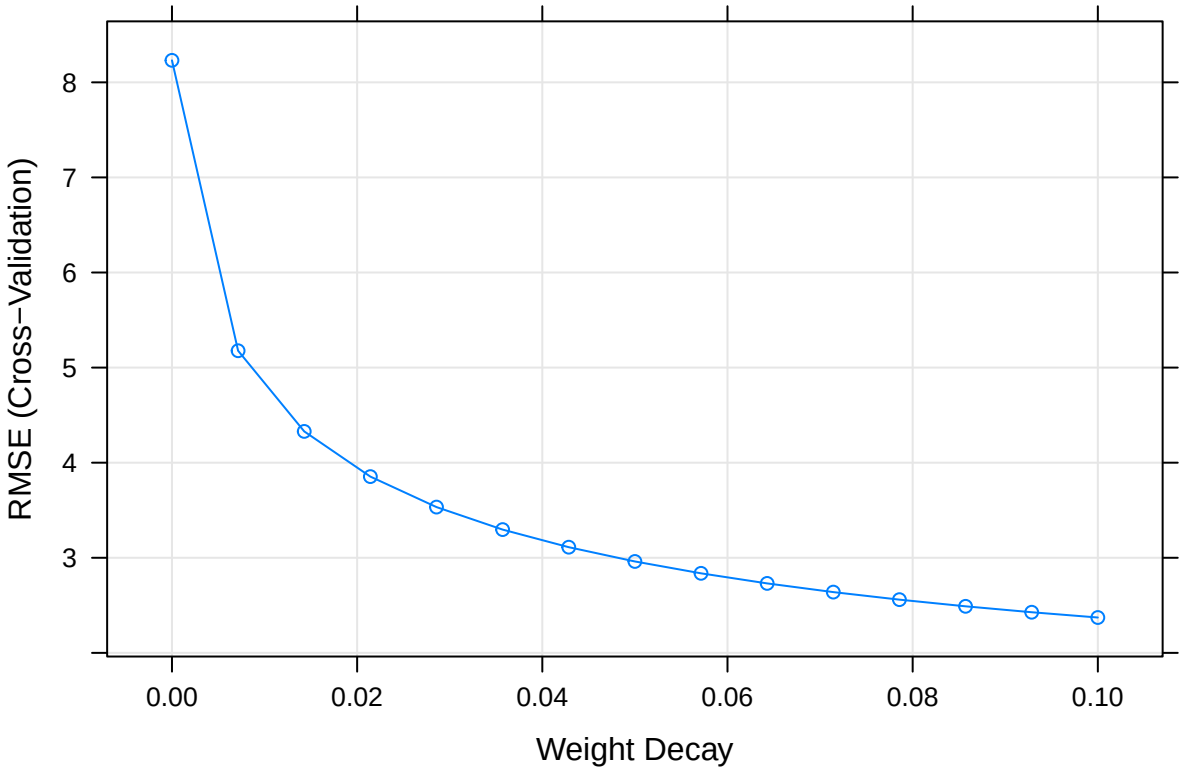
(c) Split the data into a training and a test set, pre-process the data, and tune am odel of your choice from this chapter. What is the optimal value of the performance metric?

```
yield <- as.matrix(ChemicalManufacturingProcess$Yield)
set.seed(123)
split2 <- createDataPartition(yield, p = 0.8, list = FALSE, times = 1)

Xtrain.data2  <- Xpreprocess[split2, ]
xtest.data2 <- Xpreprocess[-split2, ]
Ytrain.data2  <- yield[split2, ]
ytest.data2 <- yield[-split2, ]
```

```
ridgeGrid <- data.frame(.lambda = seq(0, .1, length = 15))
set.seed(123)
ridge.mod2 <- train(x=Xtrain.data2,
                    y=Ytrain.data2,
                    method="ridge",
                    tuneGrid=ridgeGrid,
                    trControl= ctrl)
ridge.mod2
```

```
## Ridge Regression
##
## 144 samples
##  56 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 128, 129, 129, 130, 128, 131, ...
## Resampling results across tuning parameters:
##
##   lambda       RMSE      Rsquared   MAE
##   0.000000000  8.231140  0.5001829  3.365686
##   0.007142857  5.177328  0.4945675  2.430293
##   0.014285714  4.329076  0.5069033  2.138368
##   0.021428571  3.853764  0.5158292  1.977340
##   0.028571429  3.533281  0.5225199  1.872177
##   0.035714286  3.296381  0.5278183  1.792915
##   0.042857143  3.111303  0.5321913  1.729989
##   0.050000000  2.961284  0.5359104  1.678308
##   0.057142857  2.836451  0.5391442  1.634722
##   0.064285714  2.730522  0.5420033  1.597395
##   0.071428571  2.639265  0.5445641  1.565126
##   0.078571429  2.559696  0.5468811  1.536722
##   0.085714286  2.489636  0.5489948  1.511494
##   0.092857143  2.427447  0.5509361  1.488916
##   0.100000000  2.371869  0.5527288  1.468582
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was lambda = 0.1.
```

```
plot(ridge.mod2)
```



Optimal value:

The lowest point in the curve indicates the optimal lambda: the log value of lambda that best minimised the error in cross-validation. We can extract this values as:

```
ridge.mod2$bestTune
```

```
##    lambda
## 15    0.1
```

(d) Predict the response for the test set. What is the value of the performance metric and how does this compare with the resampled performance metric of the training set?

```
predictions <- predict(ridge.mod2, xtest.data2)
cbind(RMSE = RMSE(predictions, ytest.data2),  R.squared = caret::R2(predictions, ytest.data2))
```

```
##          RMSE R.squared
## [1,] 1.38534 0.4873811
```

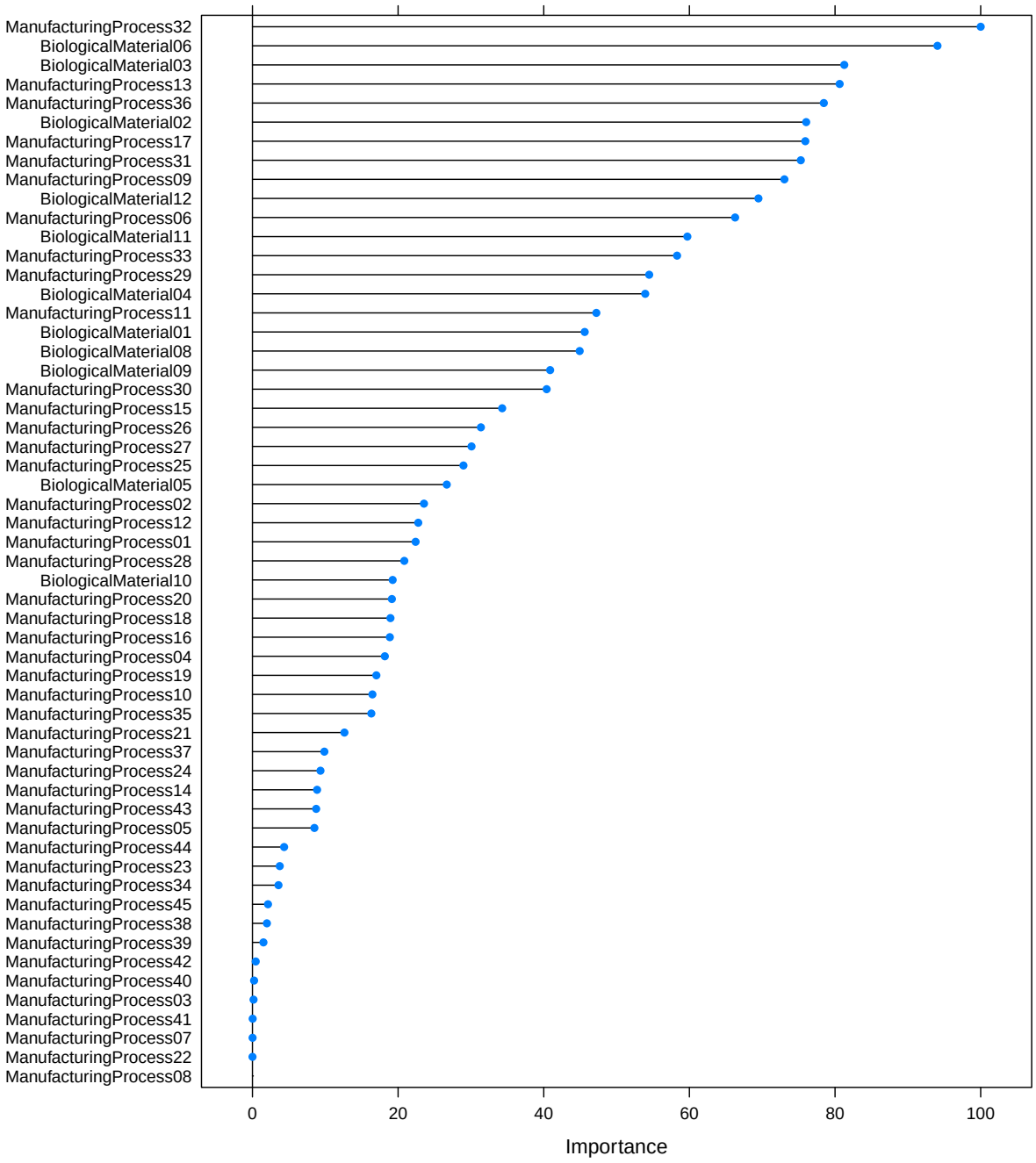
Better than the resampled metrics.

(e) Which predictors are most important in the model you have trained? Do either the biological or process predictors dominate the list?

```
varImp(ridge.mod2)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 56)
##
##                         Overall
## ManufacturingProcess32  100.00
## BiologicalMaterial06     94.06
## BiologicalMaterial03     81.27
## ManufacturingProcess13   80.63
## ManufacturingProcess36   78.46
## BiologicalMaterial02     76.04
## ManufacturingProcess17   75.92
## ManufacturingProcess31   75.30
## ManufacturingProcess09   73.04
## BiologicalMaterial12     69.48
## ManufacturingProcess06   66.28
## BiologicalMaterial11     59.72
## ManufacturingProcess33   58.31
## ManufacturingProcess29   54.48
## BiologicalMaterial04     53.93
## ManufacturingProcess11   47.22
## BiologicalMaterial01     45.62
## BiologicalMaterial08     44.93
## BiologicalMaterial09     40.88
## ManufacturingProcess30   40.40
```

```
plot(varImp(ridge.mod2))
```



(f) Explore the relationships between each of the top predictors and the response. How could this information be helpful in improving yield in the future runs of the manufacturing process?

```
cor(yield, ChemicalManufacturingProcess$ManufacturingProcess13)
```

```
##            [,1]
```

```
## [1,] -0.5036797
```

```
cor(yield, ChemicalManufacturingProcess$BiologicalMaterial06)
```

```
##           [,1]
## [1,] 0.4781634
```