
Split train and test dataset and fit the PLS model.

```
    library(MASS)
    library(caret)
```

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

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

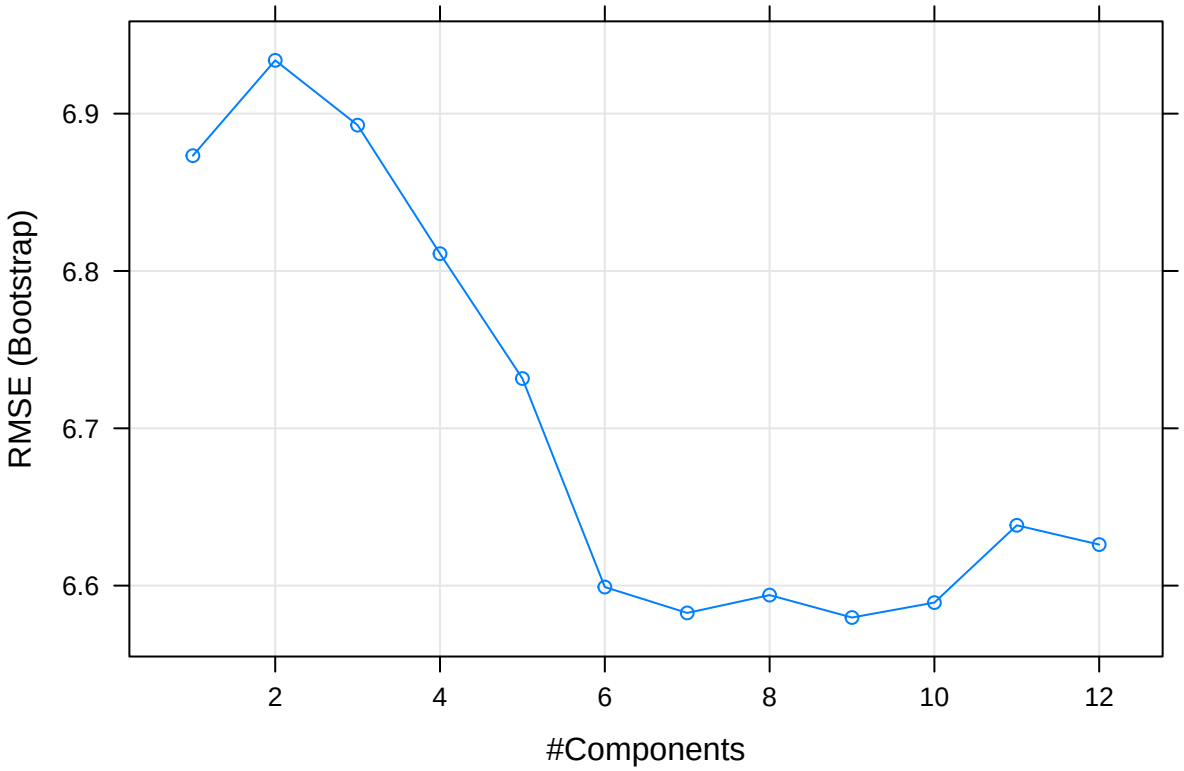
```
    head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

```
    set.seed(4)
    train = sample(1:nrow(Boston), 2/3*nrow(Boston))
    test = (-train)

    train_set = Boston[train,]
    test_set = Boston[test,]


    model = train(crim~.,data = Boston, method = "pls", metric = "RMSE", tuneLength = 20)
    plot(model)
```



From the plot, 9 components will be the best for model.Then use the model to predict test data.

```
    pred_crim = predict(model, newdata = test_set)
    mean((pred_crim-test_set$crim)^2)
```

```
## [1] 31.18738
```

The prediction MSE estimate is 31.18738.

The model which has minimum test error is LASSO, which is 22.8967, and the model has largest test error is updated to PLS, which is 31.18738.The reason might be there are some useless variables, and LASSO can help pick them out of the model.

Create a binary variable crim01 that contains a 1 if crim has a value equal to or above its median, and a 0 if crim has a value below its median.

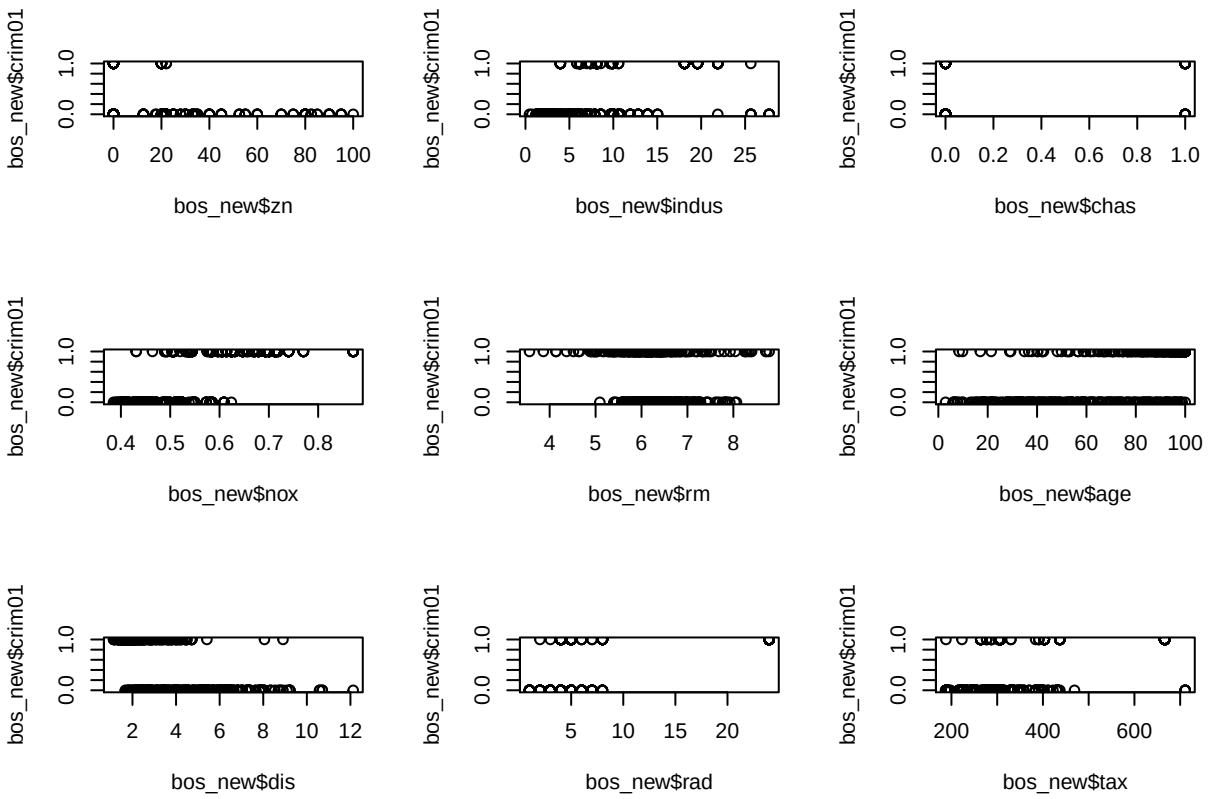
```
    Boston$crim01[Boston$crim >= median(Boston$crim)] = 1
    Boston$crim01[Boston$crim < median(Boston$crim)] = 0
    head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
```

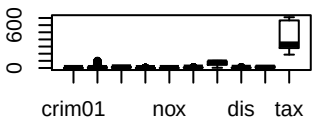
```
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv crim01
## 1 24.0      0
## 2 21.6      0
## 3 34.7      0
## 4 33.4      0
## 5 36.2      0
## 6 28.7      0
```

```
    bos_new = Boston[, c("crim01","zn", "indus","chas","nox","rm","age","dis","rad","tax")]

    par(mfrow = c(3,3))
    plot(bos_new$zn,bos_new$crim01)
    plot(bos_new$indus,bos_new$crim01)
    plot(bos_new$chas,bos_new$crim01)
    plot(bos_new$nox,bos_new$crim01)
    plot(bos_new$rm,bos_new$crim01)
    plot(bos_new$age,bos_new$crim01)
    plot(bos_new$dis,bos_new$crim01)
    plot(bos_new$rad,bos_new$crim01)
    plot(bos_new$tax,bos_new$crim01)
```



```
boxplot(bos_new)
```



From both scatter plot and box plot, the features indus, nox, rm, age, dis, and tax are most likely to be useful in predicting crim01. The variance of tax is the largest, and rest of variables have similar varibility.

Split the data into a training set and a test set with ratio 2:1.

```
set.seed(42)
train_new = sample(1: nrow(bos_new), 2/3*nrow(bos_new))
test_new = (-train)

train_bos = bos_new[train_new,]
test_bos = bos_new[test_new,]
```

The false positive rate is 16.28%, the false negative rate is 21.69%, and the accuracy of the model is 81.07%.

```
glm_model = glm(crim01 ~ indus+ nox + rm + age + dis + tax, data = train_bos, family = binomial)
pred_bos = predict(glm_model,test_bos, type = "response")

pred01 = rep(1,length(pred_bos))
pred01[pred_bos < 0.5] = 0

table(pred01,test_bos$crim01)
```

```
##
```

```
## pred01  0  1
##      0 72 18
##      1 14 65
```

```
    #false positive
    14/(72+14)
```

```
## [1] 0.1627907
```

```
    #false negative
    18/(18+65)
```

```
## [1] 0.2168675
```

```
    #accuracy of the model
    (72+65)/169
```

```
## [1] 0.8106509
```

Naive Bayes: The false positive rate is 19.77%, the false negetive is 31.33%, and the accuracy is 74.56%.

```
NB = naiveBayes(as.factor(crim01) ~ indus+ nox + rm + age + dis + tax,data = train_bos)
NB_pred = predict(NB, test_bos)
table(NB_pred, test_bos$crim01)
```

```
##
## NB_pred  0  1
##       0 69 26
##       1 17 57
```

```
#false positive
17/(69+17)
```

```
## [1] 0.1976744
```

```
#false negative
26/(26+57)
```

```
## [1] 0.313253
```

```
#accuracy
(69+57)/169
```

```
## [1] 0.7455621
```

---

compare the performance of logistic regression and Naive Bayes classifier on datasets with different joint distributions.

```r
comp <- function(alpha, p1,p2,p3,p4){
  rec = matrix(0,2,100)
  set.seed(123)
  for (ii in 1:100){
  miu1 = t(c(alpha, 0))
  sigma1 = matrix(c(p1,p2,p2,p1),2,2)
  miu2 = t(c(-alpha,0))
  sigma2 = matrix(c(p3,p4,p4,p3),2,2)

  X1 = mvrnorm(100,miu1,sigma1)
  X2 = mvrnorm(100,miu2,sigma2)

  Y = c(rep(1,100),rep(0,100))
  X = matrix(0,200,2)
  X[1:100,] = X1
  X[101:200,] = X2

  comp_data = data.frame(cbind(Y,X))

  train_comp = sample(1:nrow(comp_data),1/2*nrow(comp_data))
  test_comp = (-train)

  comp_train = comp_data[train_comp,]
  comp_test = comp_data[test_comp,]

  model_logit = glm(Y~.,data = comp_train,family = "binomial")
  pred_logit = predict(model_logit,comp_test)
  pred01 = rep(1,100)
  pred01[pred_logit<= 0.5] = 0

  rec[1,ii] = mean(pred01 == comp_test$Y)

  model_NB = naiveBayes(Y~., data = comp_train)
  pred_NB = predict(model_NB, comp_test,type = "raw")
  pred_NB01 = rep(1,100)
  pred_NB01[pred_NB[,2]<= 0.5] = 0

  rec[2,ii] = mean(pred_NB01 == comp_test$Y)
  }

  acc_logit = mean(rec[1,])
  acc_NB = mean(rec[2,])
  print(paste("The accuracy of logistic regression is ",acc_logit))
  print(paste("The accuracy of naive bayes is ",acc_NB))
}
```
