College Application Prediction

2023-07-18

PROJECT OVERVIEW

In this exercise, we will predict the number of applications received using the other variables in the College data set.

First we will split the data set into a training set and a test set.

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.2
```

```
data(College)
head(College)
```

```
##                              Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University     Yes 1660   1232    721        23        52
## Adelphi University               Yes 2186   1924    512        16        29
## Adrian College                   Yes 1428   1097    336        22        50
## Agnes Scott College              Yes  417    349    137        60        89
## Alaska Pacific University        Yes  193    146     55        16        44
## Albertson College                Yes  587    479    158        38        62
##                              F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University        2885         537     7440       3300   450
## Adelphi University                  2683        1227    12280       6450   750
## Adrian College                      1036          99    11250       3750   400
## Agnes Scott College                  510          63    12960       5450   450
## Alaska Pacific University            249         869     7560       4120   800
## Albertson College                    678          41    13500       3335   500
##                              Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University     2200  70       78      18.1          12   7041
## Adelphi University               1500  29       30      12.2          16  10527
## Adrian College                   1165  53       66      12.9          30   8735
## Agnes Scott College               875  92       97       7.7          37  19016
## Alaska Pacific University        1500  76       72      11.9           2  10922
## Albertson College                 675  67       73       9.4          11   9727
##                              Grad.Rate
## Abilene Christian University        60
## Adelphi University                  56
## Adrian College                      54
## Agnes Scott College                 59
## Alaska Pacific University           15
## Albertson College                   55
```

```
set.seed(123)
indis <- sample(1:nrow(College), round(2/3*nrow(College)), replace = FALSE)
college_train <- College[indis, ]
college_test <- College[-indis, ]
```

Now I will fit a linear model using least squares on the training set, and report the test error obtained.

```r
lm.fit <- lm(Apps~., data = college_train)
lm_pred <- predict(lm.fit, college_test )
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = college_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3098.1  -435.7   -32.6   326.9  6524.3
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -320.63000  483.82540  -0.663 0.507830
## PrivateYes  -631.06608  166.38884  -3.793 0.000167 ***
## Accept         1.22765    0.05907  20.782  < 2e-16 ***
## Enroll         0.07342    0.22242   0.330 0.741483
## Top10perc     45.28449    6.30692   7.180 2.54e-12 ***
## Top25perc    -12.88783    5.12008  -2.517 0.012144 *
## F.Undergrad    0.02496    0.04024   0.620 0.535386
## P.Undergrad    0.03394    0.03505   0.968 0.333304
## Outstate      -0.06350    0.02155  -2.947 0.003361 **
## Room.Board     0.20100    0.05392   3.728 0.000215 ***
## Books          0.16346    0.27890   0.586 0.558084
## Personal      -0.03987    0.07418  -0.537 0.591204
## PhD           -6.76818    5.36695  -1.261 0.207866
## Terminal      -5.29390    5.82889  -0.908 0.364201
## S.F.Ratio     -0.13458   14.77294  -0.009 0.992735
## perc.alumni   -7.16431    4.68079  -1.531 0.126506
## Expend         0.08032    0.01338   6.005 3.69e-09 ***
## Grad.Rate      9.82319    3.37117   2.914 0.003730 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 980.1 on 500 degrees of freedom
## Multiple R-squared:  0.918,  Adjusted R-squared:  0.9153
## F-statistic: 329.5 on 17 and 500 DF,  p-value: < 2.2e-16
```

```r
Test_error_linear <- mean((college_test$Apps - lm_pred)^2)
Test_error_linear
```

```
## [1] 1684049
```

#The test error is 1684049

Now, we will fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.3.3
```

```
## Loaded glmnet 4.1-8
```

```
set.seed(123)
X_train = model.matrix(Apps~., data = college_train)
X_test = model.matrix(Apps~., data = college_test)
#Choosing lambda using cross-validation
cv.out = cv.glmnet(X_train, college_train$Apps, alpha=0)
sel = cv.out$lambda.min
sel
```

```
## [1] 311.779
```

```
#fitting ridge model
ridge_mod = glmnet(X_train, college_train$Apps, alpha = 0, lambda=sel)
#Make predictions
ridge_pred = predict(ridge_mod, s=sel, newx = X_test, type = "response")
#Calculate test error
summary(ridge_pred)
```

```
##        s1
##  Min.   : -361.0
##  1st Qu.:  861.8
##  Median : 1760.8
##  Mean   : 3167.3
##  3rd Qu.: 3642.1
##  Max.   :27483.2
```

```
Test_error_ridge <- mean((ridge_pred - college_test$Apps)^2)
Test_error_ridge
```

```
## [1] 2791017
```

#The best lambda by cross validation is 311.779 and the test error is 2791017

Now, we will fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates

```
#first choosing best lambda
set.seed(123)
cv.out_2 = cv.glmnet(X_train, college_train$Apps, alpha=1)
sel2 = cv.out_2$lambda.min
sel2
```

```
## [1] 6.120348
```

```
#Fitting lasso model
lasso_mod = glmnet(X_train, college_train$Apps, alpha=1, lambda=sel2)
#Make predictions
lasso_pred = predict(lasso_mod, s=sel2, newx=X_test)
Test_error_lasso <- mean((lasso_pred -college_test$Apps)^2)
Test_error_lasso
```

```
## [1] 1692748
```

```
coefficient <- predict(lasso_mod, s = sel2, type = "coefficients")

coefficient[coefficient!=0]
```

```
##  [1] -409.19194117 -613.21678718    1.22135193    0.09502764   41.68875513
##  [6]   -9.93420805    0.02373219    0.02791910   -0.05716962    0.18949382
## [11]    0.11856415   -0.02360973   -6.01752912   -5.11448728   -6.85847164
## [16]    0.07907405    8.88004892
```

```
which(coefficient!=0)
```

```
##  [1]  1  3  4  5  6  7  8  9 10 11 12 13 14 15 17 18 19
```

```
numberofnonzero <- sum(coef(lasso_mod, s = sel2) != 0)
numberofnonzero
```

```
## [1] 17
```

#The best lambda by cross validation is 6.120348, the test error is 1692748 and the non-zero coefficient estimates are also listed accordingly

Now we will fit a PCR model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

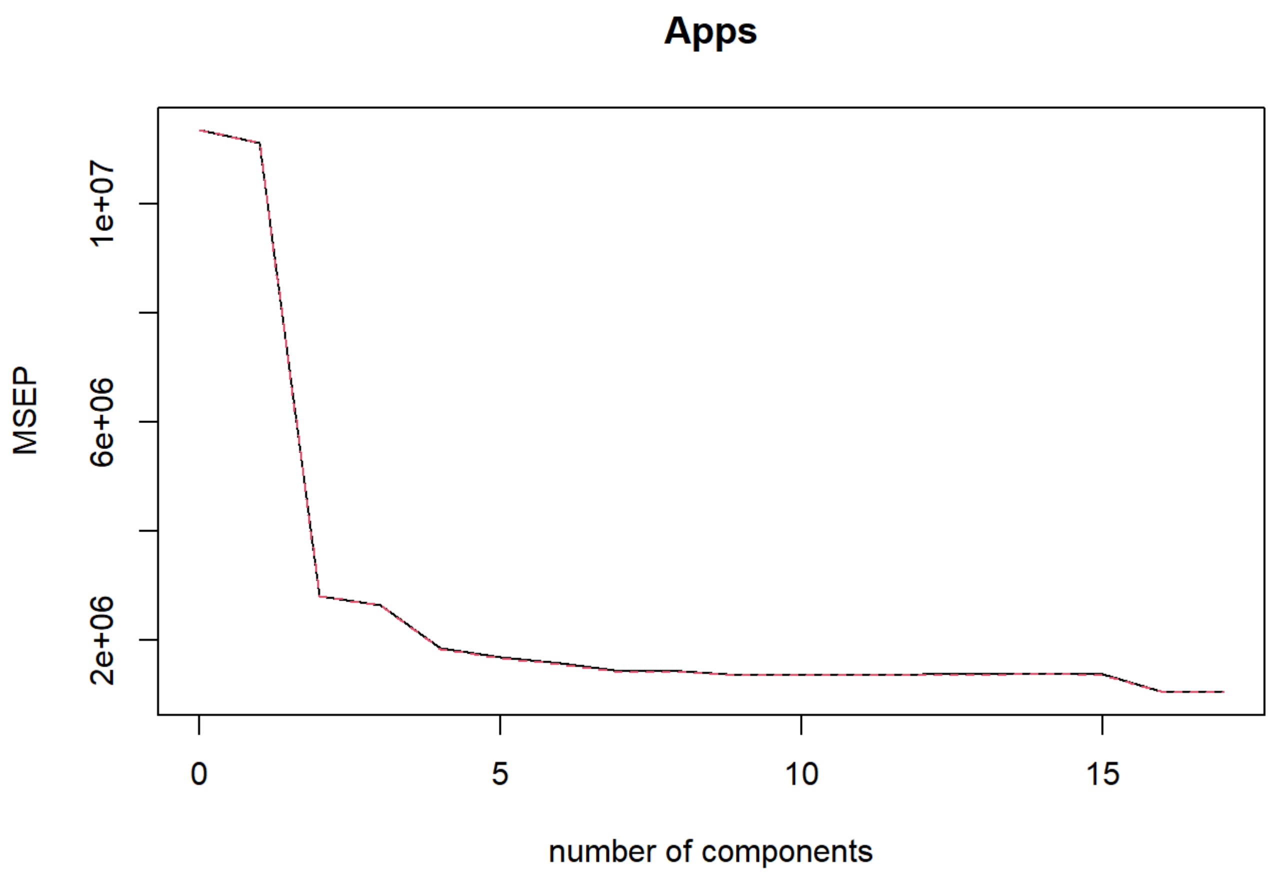
```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(123)

pcrfit <- pcr(Apps~., data=college_train, scale=TRUE, validation="CV")

validationplot(pcrfit, val.type = "MSEP")
```




```
summary(pcrfit)
```

```
## Data:    X dimension: 518 17
##  Y dimension: 518 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3370     3336     1680     1631     1363     1303     1257
## adjCV         3370     3336     1678     1630     1357     1299     1253
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1202     1201     1169      1169      1168      1174      1176
## adjCV     1195     1196     1166      1167      1165      1171      1173
##        14 comps  15 comps  16 comps  17 comps
## CV         1176      1176      1029      1029
## adjCV      1173      1173      1025      1025
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      31.765    57.84    64.68    70.19    75.49    80.39    84.01    87.40
## Apps    3.386    75.80    77.45    84.75    86.02    86.91    88.03    88.22
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.57     93.02     95.07     96.93     98.02     98.88     99.40
## Apps    88.84     88.89     88.94     88.98     89.03     89.03     89.23
##       16 comps  17 comps
## X        99.82     100.0
## Apps     91.74      91.8
```

#The lowest MSEP occurs at around M = 17 which can be confirmed from the summary as well ignoring the rest of the components which we neglect for now as they do not solve the purpose.

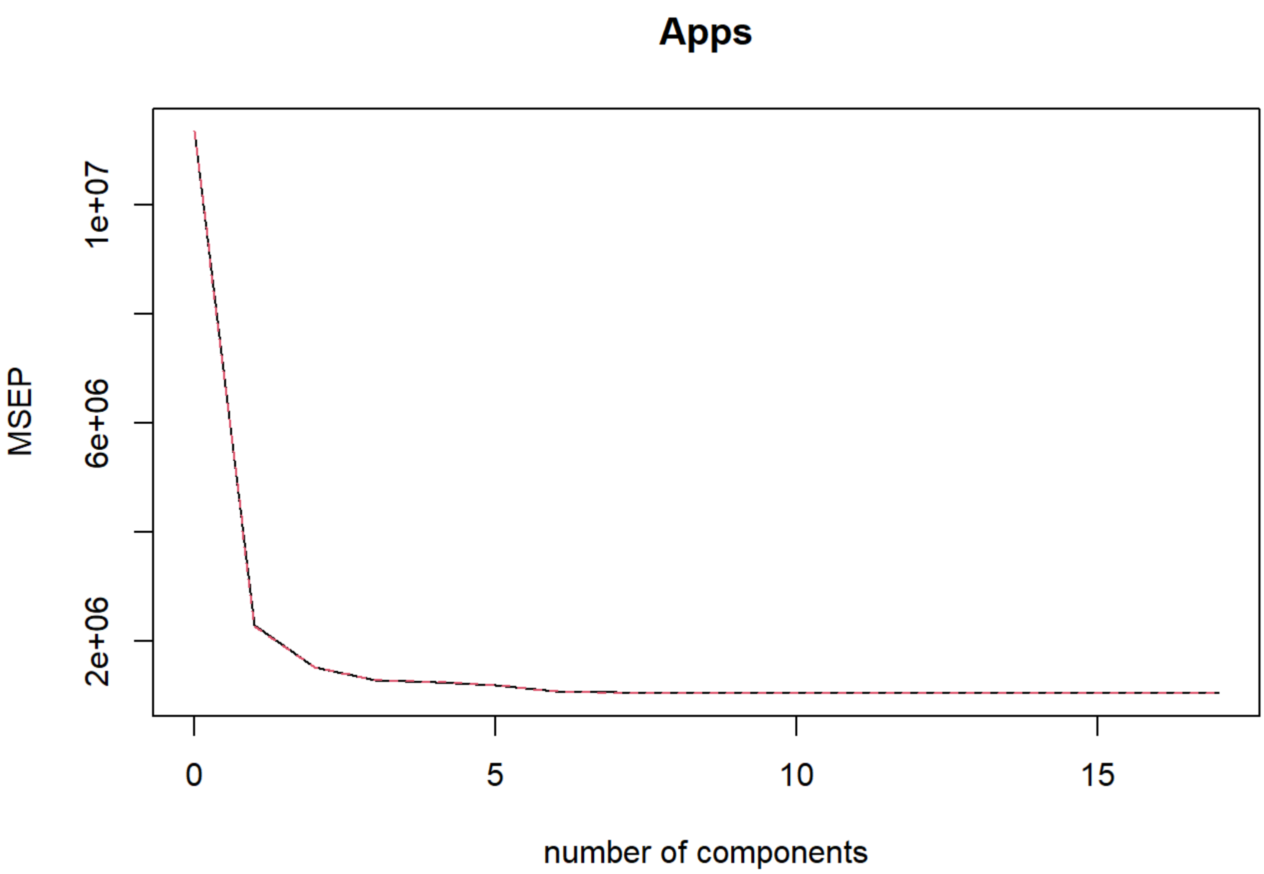
```
#predicting using M = 17 found by cross validation
pcrfit1 <- pcr(Apps~., data=college_train, scale=TRUE, ncomp=17)
prediction <- predict(pcrfit1, college_test, ncomp=17)

#test error
Test_error_pcr <- mean((prediction-college_test$Apps)^2)
Test_error_pcr
```

```
## [1] 1684049
```

#also confirmed the M value by changing the value of ncomp value from 8 to 17 and got the minimum value of test error at 17. Hence, considered M = 17 in the final answer. #The test error is 1684049

Now we will fit a PLS model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

```
set.seed(123)
#Fit and determine M based on CV results
plsfit = plsr(Apps~., data=college_train, scale=TRUE, validation="CV")
validationplot(plsfit, val.type = "MSEP")
```




```
summary(plsfit)
```

```
## Data: 	X dimension: 518 17
## 	Y dimension: 518 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3370     1513     1233     1138     1121     1099     1045
## adjCV         3370     1511     1236     1136     1117     1092     1040
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1031     1028     1029      1030      1027      1028      1028
## adjCV     1027     1025     1026      1026      1024      1025      1025
##        14 comps  15 comps  16 comps  17 comps
## CV         1029      1029      1029      1029
## adjCV      1025      1025      1025      1025
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       26.30    42.01    63.26    67.75    71.41    74.08    77.53    80.83
## Apps    80.53    86.92    89.34    90.16    91.05    91.71    91.77    91.79
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.35     86.14     89.53     91.21     93.22     94.67     97.06
## Apps    91.79     91.80     91.80     91.80     91.80     91.80     91.80
##       16 comps  17 comps
## X        99.11     100.0
## Apps     91.80      91.8
```

#From the summary and the plot, the lowest MSEP occur at M = 17.

```
#making prediction with M = 17
plsfit1 = plsr(Apps~., data=college_train, scale=TRUE, ncomp=17)
prediction = predict(plsfit1, college_test, ncomp = 17)
#test error
Test_error_pls <- mean((prediction - college_test$Apps)^2)
Test_error_pls
```

```
## [1] 1684049
```

#also confirmed the M value by changing the value of ncomp value from 8 to 17 and got the minimum value of test error at 17. Hence, considered M = 17 in the final answer. #the test error is 1684049

Finally explaining the results obtained.We will check how accurately can we predict the number of college applications received. Also, is there much difference among the test errors resulting from these five approaches?

```
Test_error_linear
```

```
## [1] 1684049
```

```
Test_error_ridge
```

```
## [1] 2791017
```

```
Test_error_lasso
```

```
## [1] 1692748
```

```
Test_error_pcr
```

```
## [1] 1684049
```

```
Test_error_pls
```

```
## [1] 1684049
```

We see that the test errors for inear regression, PCR and PLS are relatively close, the test errors of ridge regression is a little higher. Lasso also has a little more value as compared to linear, pcr and pls. There are not much differences in the test errors except for that in ridge regression. We can predict the number of college application received with reasonable accuracy.

Data Preparation

I began by splitting the dataset into a training set and a test set. This was done to ensure that the models could be evaluated on unseen data, helping to prevent overfitting and giving a better estimate of their performance in real-world scenarios.

Linear Regression

The first model I used was linear regression, which served as a baseline for understanding the relationship between the number of applications and the other variables in the dataset. By fitting this model to the training data, I could predict the number of applications in the test set and calculate the associated test error. This provided a straightforward way to gauge how well the other variables explained the variation in application numbers.

Ridge Regression

Next, I explored Ridge Regression, a technique designed to address issues of multicollinearity by regularizing the coefficients of the model. This helps to prevent overfitting, especially when dealing with datasets that have highly correlated variables. I used cross-validation to select the optimal value for the regularization parameter, λ. Ridge Regression slightly increased the test error compared to linear regression, indicating that while it may help with multicollinearity, it didn't drastically improve predictive accuracy for this particular dataset.

Lasso Regression

I then applied Lasso Regression, which not only regularizes the coefficients like Ridge Regression but also performs variable selection by shrinking some coefficients to zero. This makes Lasso particularly useful when dealing with high-dimensional data, as it can simplify the model by eliminating less important variables. After selecting the optimal λ via cross-validation, I found that the test error was similar to that of linear regression, but with fewer variables contributing to the prediction. This indicated which predictors were most influential, offering insights into the key factors driving college applications.

Principal Component Regression (PCR)

Principal Component Regression (PCR) was the next technique I used. PCR reduces the dimensionality of the data by transforming the original variables into a set of uncorrelated components before fitting the regression model. I used cross-validation to determine the optimal number of components (M) to include. The results showed that using 17 components provided the lowest test error, which was comparable to the linear regression model. This indicated that while PCR effectively reduced the dimensionality, it did not significantly improve the predictive accuracy.

Partial Least Squares (PLS)

Finally, I applied Partial Least Squares (PLS), which, like PCR, is a dimensionality reduction technique. However, PLS differs by considering the relationship between the predictors and the response variable when forming the components. Cross-validation also suggested using 17 components, resulting in a test error similar to that of PCR and linear regression. This reinforced the idea that while dimensionality reduction can be useful, it didn’t offer a substantial performance boost for this dataset.

Results and Conclusions

After comparing the test errors from all the models, I found that the differences were relatively small. Linear regression, PCR, and PLS produced similar test errors, while Ridge Regression had a slightly higher error. Lasso Regression performed similarly to linear regression but with fewer variables, providing a more interpretable model.

These results suggest that for this dataset, simpler models like linear regression are sufficient to achieve reasonable predictive accuracy. More complex models like Ridge and Lasso Regression or dimensionality reduction techniques like PCR and PLS did not significantly outperform linear regression. This indicates that the relationships in the data were relatively straightforward, and more advanced methods were not necessary for accurate predictions in this case.