Predictive Models for College Applications

Name: Anand **UB Mail:** anand6@buffalo.edu

Predict number of applications received using the variables in the college dataset using different methods.

Dataset: College

```
> dim(College)
[1] 777  18
> names(College)
 [1] "Private"     "Apps"        "Accept"
 [4] "Enroll"      "Top10perc"   "Top25perc"
 [7] "F.Undergrad" "P.Undergrad" "Outstate"
[10] "Room.Board"  "Books"       "Personal"
[13] "PhD"         "Terminal"    "S.F.Ratio"
[16] "perc.alumni" "Expend"      "Grad.Rate"
```

a. **Linear Regression:**

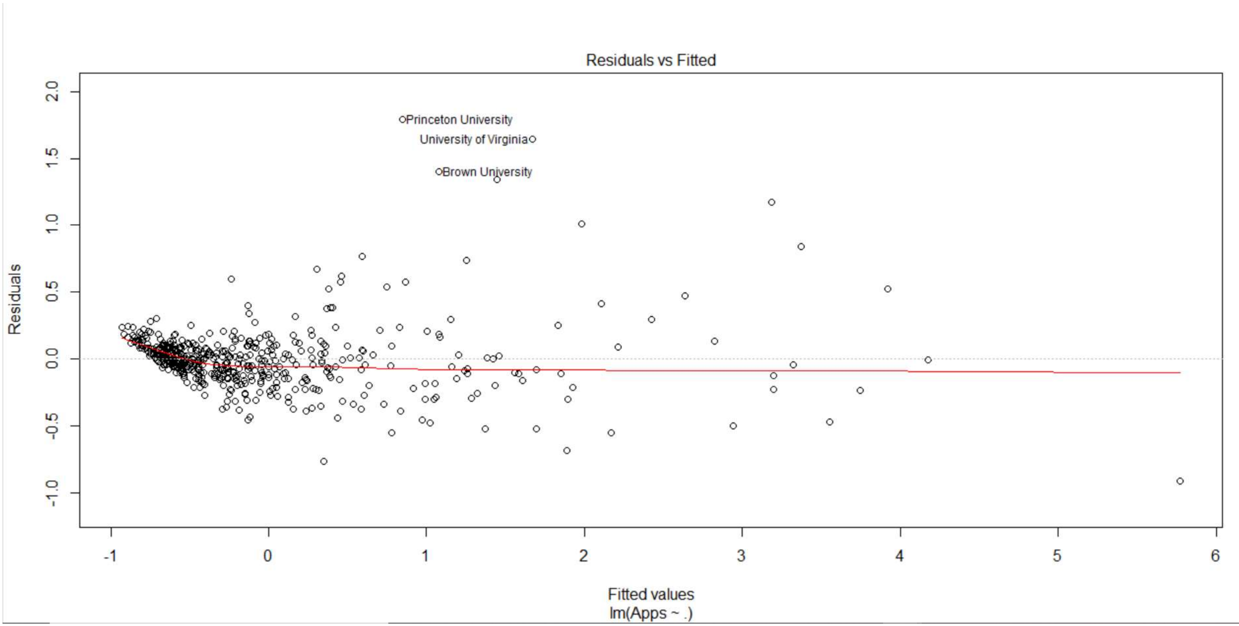
Y=b0+b1*x where b0 is a bias and b1 is bias for the predictor x.
It has very less bias and more variance as it considers all predictors in the model.

Dividing data into test and train:

```
> train_index <- sample(1:nrow(tCollege), (2/3) * nrow(tCollege))
> test_index <- setdiff(1:nrow(tCollege), train_index)
> traindat<-tCollege[train_index,]
> testdat<-tCollege[test_index,]
> dim(traindat)
[1] 518  18
> dim(testdat)
[1] 259  18
```

Linear model fit:

```
lm.fit<-lm(Apps~.,traindat)
```



It shows the significant predictors for the linear regression:

```
> summary(lm.fit)

Call:
lm(formula = Apps ~ ., data = traindat)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.91508 -0.10772 -0.01060  0.08257  1.79457 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.087796   0.032287   2.719  0.00677 ** 
Accept       0.842285   0.040158  20.974  < 2e-16 ***
Enroll      -0.129281   0.055116  -2.346  0.01939 *  
Top10perc    0.233973   0.028578   8.187 2.24e-15 ***
Top25perc   -0.081118   0.025944  -3.127  0.00187 ** 
F.Undergrad  0.138136   0.046387   2.978  0.00304 ** 
P.Undergrad  0.020166   0.013128   1.536  0.12514    
Outstate    -0.039432   0.023804  -1.656  0.09825 .  
Room.Board   0.041105   0.016039   2.563  0.01067 *  
Books        0.009508   0.011933   0.797  0.42594    
Personal     0.003839   0.013486   0.285  0.77600    
PhD         -0.028199   0.022267  -1.266  0.20596    
Terminal    -0.031971   0.022694  -1.409  0.15953    
S.F.Ratio    0.010631   0.016243   0.654  0.51309    
perc.alumni -0.009356   0.015535  -0.602  0.54729    
Expend       0.085882   0.019856   4.325 1.84e-05 ***
Grad.Rate    0.045612   0.015181   3.005  0.00279 ** 
Privbin     -0.132589   0.041763  -3.175  0.00159 ** 
```

```
> lmpred<-predict(lm.fit,testdat)
> lm.MSerror<-mean((lmpred-testdat$Apps)^2)
> lm.MSerror
[1] 0.1052585
```

b. Ridge Regression: method penalizes the mean squared loss there by reducing the beta coefficients by a constant – lambda which is learnt using cross validation from data. Ridge regression adds bias estimates reasonably which can approximate to the response variable there by making predictors significant which influence response variable where as least square fit can calculate bias estimate larger.

```
ridge.fit<-cv.glmnet(trainmat[,-1],trainmat[,"Apps"],nfolds=10,alpha=0)
```

alpha=0 for ridge

Applying 10-fold cross validation where dataset is divided into 10 subsets and leaving one out every iteration and finding the best possible lambda value for the ridge regression.

```
> ridge.pred<-predict(ridge.fit,testmat[,-1],s=cv.lambda)
> ridge.MSError<-mean((ridge.pred-testdat$Apps)^2)
> ridge.MSError
[1] 0.1903895
```

c. LASSO:

Shrinkage and selection of variables together by penalizing– LASSO.
Shrinkage to zero and eliminate the predictors and makes model simpler and easily interpretable.

Lambda =0 none of the variables eliminated

Lambda=infinity all variables eliminated

```
> lasso.fit<-cv.glmnet(trainmat[,-1],trainmat[,"Apps"],nfolds=10,alpha=
1)
```

alpha=1 for LASSO.

Applying 10-fold cross validation where dataset is divided into 10 subsets and leaving one out every iteration and finding the best possible lambda value for the LASSO.

```
> head(lasso.fit$lambda)
[1] 0.8307842 0.7569797 0.6897317 0.6284578 0.5726273 0.5217567
> cv.lambda<-lasso.fit$lambda.min
> cv.lambda
[1] 0.003127849
```

```
> lasso.pred<-predict(lasso.fit,testmat[,-1],s=cv.lambda)
> lasso.MSError<-mean((lasso.pred-testdat$Apps)^2)
> lasso.MSError
[1] 0.1154724
```

Coefficients of LASSO:

```
> lasso.fit<-glmnet(trainmat[,-1],trainmat[,"Apps"],alpha=1)
> lasso.pred.coef<-predict(lasso.fit,s=cv.lambda,type="coefficients")
> lasso.pred.coef
18 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  0.080731171
Accept       0.775955333
Enroll       .
Top10perc    0.189053815
Top25perc   -0.041851007
F.Undergrad  0.070784451
P.Undergrad  0.016772016
Outstate    -0.017666881
Room.Board   0.034041414
Books        0.008780688
Personal     0.001128648
PhD         -0.021329841
Terminal    -0.030786381
S.F.Ratio    0.004540648
perc.alumni -0.012917850
Expend       0.078070208
Grad.Rate    0.034401389
Privbin     -0.125174062
```

d. <u>PCR: Principal component regression</u>

This model uses principal component which explains the predictors which shows variations in the dataset and it assumes that the variations in the predictors is related to the response variable.

If all principal components are used to model the regression which will be equivalent to linear regression. So, we have to choose the number of principal components to be used for the model.

We will use cross validation method to verify the PCR k value.

```
> pcr.fit<-pcr(Apps~.,data=traindat,validation="CV")
> summary(pcr.fit)
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
CV          0.8895   0.8542   0.4401   0.4410   0.4356   0.3736
adjCV       0.8895   0.8542   0.4393   0.4405   0.4443   0.3727
       6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
CV      0.3206   0.3095   0.3093   0.3043    0.3047    0.3051
adjCV   0.3204   0.3074   0.3075   0.3038    0.3042    0.3043
       12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
CV       0.3075    0.3057    0.3077    0.3024    0.2736    0.2632
adjCV    0.3068    0.3049    0.3068    0.3016    0.2727    0.2622
```

```
TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X      33.779    57.42    64.71    70.69    76.30    81.28
Apps    8.338    76.27    76.28    76.42    83.43    87.73
      7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
X       85.17    88.83    92.19     94.49     96.28     97.54
Apps    88.62    88.65    88.99     89.06     89.17     89.17
      13 comps  14 comps  15 comps  16 comps  17 comps
X        98.45     99.04     99.49     99.84    100.00
Apps     89.28     89.28     89.68     91.58     92.32
```

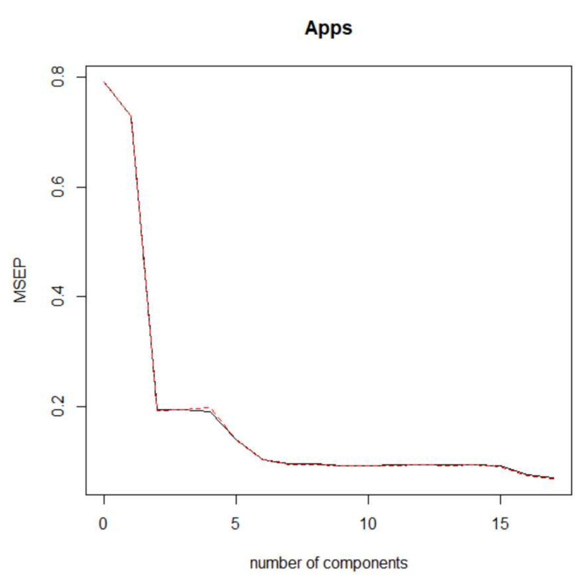
Value of mean square error for each principal component is shown which is calculated using the cross validation and also shows the variance in the predictors and also response variable for each principal component.

Coefficients for each principal component is given by :

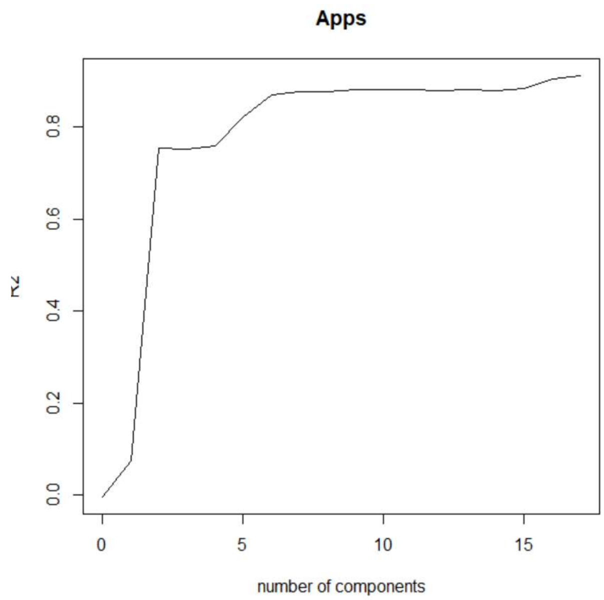
```
> pcr.fit$coefficients
, , 1 comps

                     Apps
Accept       7.245150e-03
Enroll       2.733430e-03
Top10perc    3.975426e-02
Top25perc    3.808601e-02
F.Undergrad -6.095137e-05
P.Undergrad -9.619995e-03
Outstate     4.115276e-02
```

Residuals for principal component 1: last index is the number of principal components

```
> pcr.fit$residuals[1:10,1,1]
       John Brown University            Houghton College
                  -0.4579014                  -0.5206781
   Southwest State University      Southwestern University
                  -0.1698321                  -0.6320632
            St. Paul's College          Gonzaga University
                  -0.1618197                  -0.3366759
Auburn University-Main Campus         Chestnut Hill College
                   1.2771293                  -0.7104173
        Campbellsville College        Castleton State College
                  -0.2583032                  -0.2974880
```



Mean square error gets constant from 8 components and reduces when it gets to include all principal components.



Principal components from 7 covers variations of 90% data and MSE is at lower constant. So optimally let's use principal components 7. Given 7 variables it covers 90 percent of data and MSE is minimal too.

```
> pcr.pred<-predict(pcr.fit,testdat,ncomp=7)
> pcr.MSError<-mean((pcr.pred-testdat$Apps)^2)
> pcr.MSError
[1] 0.2713831
```

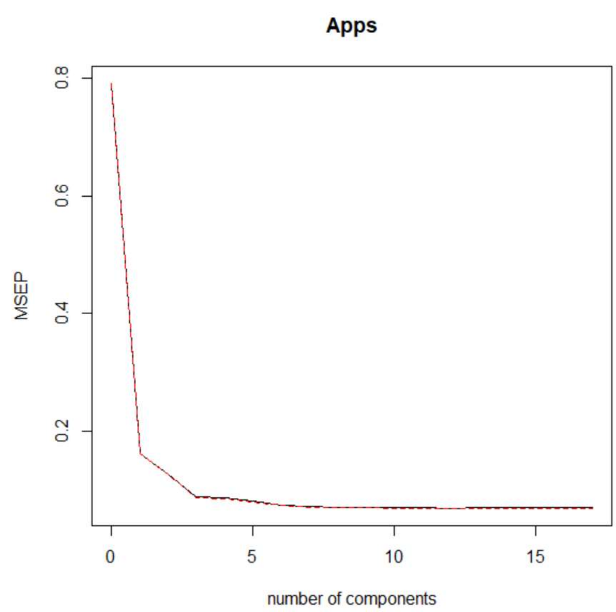
e. Partial least square:

The main problem of PCR is just focusing on the variation in predictors and assumes variations in predictors will reflect in the variation in the response variable. But this assumption may not hold good and variations in predictors will not be able to identify variations in response variable.

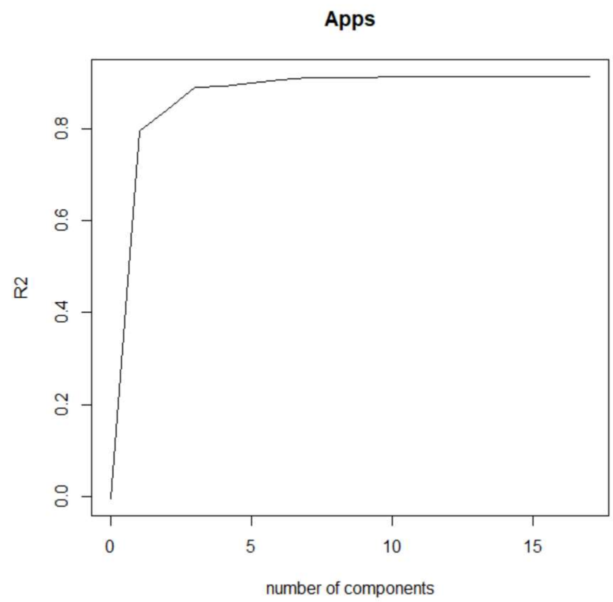
PCR just looks in the direction of predictors but PLS looks in both direction of predictors and also response variable to get the optimal solution.

It uses least square solution to find coefficients and apply weight if the variable is correlated with Y so it's called partial least square method.

```
> plsr.fit<-plsr(Apps~.,data=traindat,validation="CV")
> summary(plsr.fit)
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          0.8895   0.4026   0.3552   0.2954   0.2933   0.2830   0.2712
adjCV       0.8895   0.4013   0.3555   0.2948   0.2920   0.2798   0.2696
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV      0.2667   0.2651   0.2641    0.2630    0.2630    0.2625    0.2625
adjCV   0.2659   0.2642   0.2632    0.2622    0.2622    0.2616    0.2617
       14 comps  15 comps  16 comps  17 comps
CV       0.2626    0.2627    0.2627    0.2627
adjCV    0.2617    0.2618    0.2619    0.2618

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       25.52    53.41    62.68    66.21    68.61    72.68    77.41    81.03
Apps    79.94    84.59    89.57    90.43    91.60    91.98    92.10    92.17
      9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X       83.65     87.40     90.57     92.31     94.06     95.90     98.17
Apps    92.23     92.26     92.29     92.31     92.32     92.32     92.32
```



Mean square error has reduced early as PLS focus on the response variable along with predictor variations so it converges early for a smaller number of principal components.



Let’s consider k=6, as MSE gets constant from 6 principal components and also covers variations of 90% of data.

```
> plsr.pred<-predict(plsr.fit,testdat,ncomp=6)
> plsr.MSError<-mean((plsr.pred-testdat$Apps)^2)
> plsr.MSError
[1] 0.1215716
```

Prediction error reduced to half using PLS compared to PCR.

Conclusion:

Method	Parameter	MSE	Comments
Linear Regression	NA	0.1052585	all predictors
Ridge Regression	lambda=0.08307842	0.1903895	proportional shrinkage
LASSO	lambda=0.003127849	0.1154724	soft thresholding
PCR	k=7	0.2713831	variations in predictors
PLS	k=6	0.1215716	variations in predictors and response variable

We can predict number of applications received with 90-95% accuracy using any of these models but choosing simpler model which has correlation with response variable makes model interpretable easily. Using . LASSO and PLS has better accuracy other than OLS and their coefficients as mentioned below:

PLS:

```
> plsr.fit$coefficients[,,6]
      Accept       Enroll     Top10perc    Top25perc
 0.748756871  0.069308359  0.219317116 -0.074665437
 F.Undergrad  P.Undergrad     Outstate   Room.Board
 0.015697876  0.035065312 -0.028624605  0.046401321
       Books     Personal          PhD     Terminal
 0.019160631  0.006369821 -0.002508178 -0.069338814
   S.F.Ratio  perc.alumni       Expend    Grad.Rate
 0.033731672 -0.012208231  0.113626263  0.043005399
     Privbin
-0.104563407
```

LASSO:

```
(Intercept)  0.080731171
Accept       0.775955333
Enroll       .
Top10perc    0.189053815
Top25perc   -0.041851007
F.Undergrad  0.070784451
P.Undergrad  0.016772016
Outstate    -0.017666881
Room.Board   0.034041414
Books        0.008780688
Personal     0.001128648
PhD         -0.021329841
Terminal    -0.030786381
S.F.Ratio    0.004540648
perc.alumni -0.012917850
Expend       0.078070208
Grad.Rate    0.034401389
Privbin     -0.125174062
```