PRINCIPAL COMPONENT REGRESSION & PARTIAL LEAST SQUARE ON GRADUATE ADMISSION DATA

Introduction

This dataset is created for prediction of Graduate Admissions from an Indian perspective. The dataset was built with the purpose of helping students in shortlisting universities with their profiles. The predicted output gives them a fair idea about their chances for a particular university.

About the Data

The dataset contains several parameters which are considered important during the application for Masters Programs. It contains 400 observations and 8 variables.

The Variables included are :

- GRE Scores (out of 340)
- TOEFL Scores (out of 120)
- University Rating (out of)
- Statement of Purpose and Letter of Recommendation Strength (out of 5)
- Undergraduate GPA (out of 10)
- Research Experience (either 0 or 1)
- Chance of Admit (ranging from 0 to 1)

CITATION

Mohan S Acharya, Asfia Armaan, Aneeta S Antony : A Comparison of Regression Models for Prediction of Graduate Admissions, IEEE International Conference on Computational Intelligence in Data Science 2019

Principal component regression (PCR) is a linear regression technique that involves principal component analysis. It is especially used to overcome the problem with multicollinearity in linear regression by combining the explanatory variables to a smaller set of uncorrelated variables (principal components associated with the smallest eigen values. It is problematic to use multiple linear regression if there is a strong correlation between explanoratory variables. One way to deal with the problem of collinearity is to delete one of the explanatory variables. However, we might remove important information. Also, if we like to predict how much one variable affects the response variable when we control other variables, this means we need to include all predictor variables in the model. Instead of deleting one variable, we could combine the explanatory variables using principal component. Another advantage of principal component regression is its ability to reduce overfitting leading to better performance of the model.

Partial least squares is a supervised alternative to principal component analysis. Like PCR, PLS is a dimension reduction method which first identifies a new set of features Z1,.., Zm that are linear combinations of the original features, and then fit a linear modelvia least squares using the M new features.

```
#libraries
library(psych)
library(pls)
library(Metrics)
library(corrplot)
library(tidyverse)

# Import data
df <- read.csv("Admission_Predict.csv")
df <- df[-1]
attach(df)
```

Since my target variable is a proportion or probability that are bounded between 0 and 1, and I want to use it for modeling purposes, I perform a logit transformation on it.The logit transformation maps the original values to a continuous scale from negative infinity to positive infinity.

```
# Logit transformation
y <- Chance.of.Admit
Y_trans <- log(y / (1 - y))

new_df <- df %>% mutate(Y_trans)
new_df <- new_df %>% select(Y_trans, GRE.Score,TOEFL.Score,
University.Rating, SOP, LOR, CGPA, Research)
colnames(new_df)[colnames(new_df) == "Y_trans"] <- "Chance.Of.Admit"
head(new_df)

##   Chance.Of.Admit GRE.Score TOEFL.Score University.Rating SOP LOR CGPA
Research
## 1       2.4423470       337         118                4 4.5 4.5 9.65
1
## 2       1.1526795       324         107                4 4.0 4.5 8.87
1
## 3       0.9444616       316         104                3 3.0 3.5 8.00
1
## 4       1.3862944       322         110                3 3.5 2.5 8.67
1
## 5       0.6190392       314         103                2 2.0 3.0 8.21
0
## 6       2.1972246       330         115                5 4.5 3.0 9.34
1

# Checking data for missing values
missing_values <- sum(is.na(new_df))
missing_values
```

```
## [1] 0
```

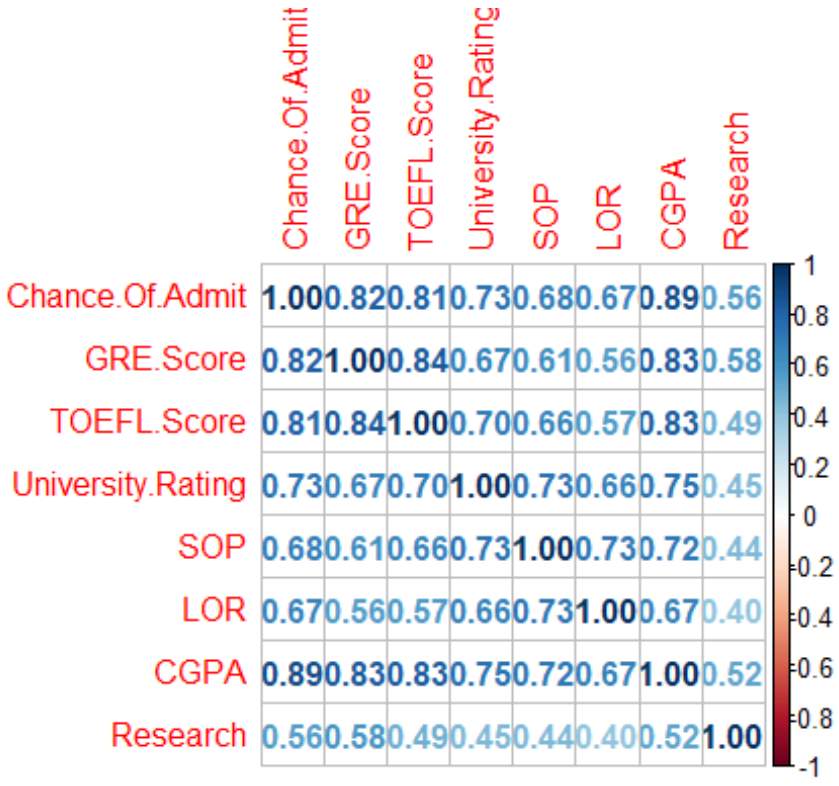
- There are no missing values in the data

```
# Summary statistics of the data
summary(new_df)

##  Chance.Of.Admit      GRE.Score       TOEFL.Score    University.Rating
##  Min.   :-0.6633   Min.   :290.0   Min.   : 92.0   Min.   :1.000
##  1st Qu.: 0.5754   1st Qu.:308.0   1st Qu.:103.0   1st Qu.:2.000
##  Median : 0.9946   Median :317.0   Median :107.0   Median :3.000
##  Mean   : 1.1218   Mean   :316.8   Mean   :107.4   Mean   :3.087
##  3rd Qu.: 1.5856   3rd Qu.:325.0   3rd Qu.:112.0   3rd Qu.:4.000
##  Max.   : 3.4761   Max.   :340.0   Max.   :120.0   Max.   :5.000
##       SOP             LOR             CGPA          Research
##  Min.   :1.0   Min.   :1.000   Min.   :6.800   Min.   :0.0000
##  1st Qu.:2.5   1st Qu.:3.000   1st Qu.:8.170   1st Qu.:0.0000
##  Median :3.5   Median :3.500   Median :8.610   Median :1.0000
##  Mean   :3.4   Mean   :3.453   Mean   :8.599   Mean   :0.5475
##  3rd Qu.:4.0   3rd Qu.:4.000   3rd Qu.:9.062   3rd Qu.:1.0000
##  Max.   :5.0   Max.   :5.000   Max.   :9.920   Max.   :1.0000

# Checking the correlation between pairs of variables
# cor(new_df)

corrplot(cor(new_df), method="number")
```



Above is the pairwise correlation coefficients between all pairs of variables in the data frame. A correlation of 1 indicates a perfect positive correlation, -1 indicates a perfect negative

correlation, and 0 indicates no correlation. From the correlation plot, we see that all variables are positively influence the chances of admission with values over 0.5. The correlation coefficients among the exploratory variables are generally moderate to high, but not extremely close to 1 or -1, which suggests that there may be some correlation among the variables, but not to a severe extent. We can use a scatter plot to visualize it better.

```
# pairplot
# pairs(new_df[,1:6])
# pairs(new_df[,5:7])
```

Using matrix approach to detect outliers. I computed the hat matrix and printed the values of the diagonal of the hat matrix which are greater than the condition 2p/n (p = predictors, n = total observations) and considered them to be high leverage points.

```
n <- length(GRE.Score)
X <- cbind(rep(1,n), GRE.Score, TOEFL.Score, University.Rating, SOP, LOR,
CGPA, Research)
XTX.inv <- solve(t(X) %*% X)

# Hat matrix
H <- X %*% XTX.inv %*% t(X)

# Diagonal values of the hat matrix
hii <- diag(H)
```

- Using matrix approach to detect outliers. I computed the hat matrix and printed the values of the diagonal of the hat matrix which are greater than the condition 2p/n (p = predictors, n = total observations) and considered them to be high leverage points.

```
condition <- (2 * 7) / 400
which(hii > condition)

##  [1]  22  39  40  51  53  54  56  57  58  59  76  79  91  92  96 110 111
118 120
## [20] 123 126 132 139 140 142 169 252 258 259 274 294 298 346
```

I went further to compute the cook's distance which measures the influence of the ith observation (yi) on all fitted values. The ith observation is influential if the cook's distance is greater than F(0.95, p, n-p).

```
model <- lm(Chance.of.Admit ~ ., data = df)
MSE <- sum(resid(model)^2)/(n - 7)
r.i <- resid(model)/(sqrt(MSE * (1-hii)))
c.i <- ((r.i^2)/7) * (hii/(1-hii))

# Determine the critical value for Cook's distance based on the F-
distribution using 0.05 level of significance.
F <- qf(0.5, 7, 393)

# Compare Cook's distance for each observation to the critical value.
Observations with Cook's distance greater than the critical value are
considered potential outliers
```

```
for (i in c.i)
{
  if (i > F) {
    print(i)
  }
}
```

There are no influential ith observations.

SPLITTING OF DATA INTO TRAIN AND TEST

```
# Split the data into train and test
train <- new_df[1:300, c("Chance.Of.Admit", "GRE.Score", "TOEFL.Score",
"University.Rating", "SOP", "LOR", "CGPA", "Research")]
y_test <- new_df[301:nrow(new_df), c("Chance.Of.Admit")]
test <- new_df[301:nrow(new_df), c("GRE.Score", "TOEFL.Score",
"University.Rating", "SOP", "LOR", "CGPA", "Research")]
```

BASELINE MODEL (LEAST SQUARES REGRESSION MODEL)

```
# fitting a simple linear regression model
simple_lm <- lm(Chance.Of.Admit ~., data = train)
summary(simple_lm)
```

```
##
## Call:
## lm(formula = Chance.Of.Admit ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30307 -0.20955  0.03346  0.24590  0.88918
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -11.208770   0.818589 -13.693  < 2e-16 ***
## GRE.Score           0.010641   0.003924   2.712 0.007093 **
## TOEFL.Score         0.020120   0.006963   2.890 0.004144 **
## University.Rating   0.076859   0.030118   2.552 0.011223 *
## SOP                -0.036449   0.034850  -1.046 0.296486
## LOR                 0.136212   0.036504   3.731 0.000229 ***
## CGPA                0.711221   0.079559   8.940  < 2e-16 ***
## Research            0.140777   0.051463   2.735 0.006610 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3703 on 292 degrees of freedom
## Multiple R-squared:  0.826,  Adjusted R-squared:  0.8219
## F-statistic: 198.1 on 7 and 292 DF,  p-value: < 2.2e-16
```

We are not just going to remove features based on whether they are significant or not.

```
lm_pred <- predict(simple_lm, test)
rmse(actual = y_test, predicted = as.numeric(lm_pred))
```

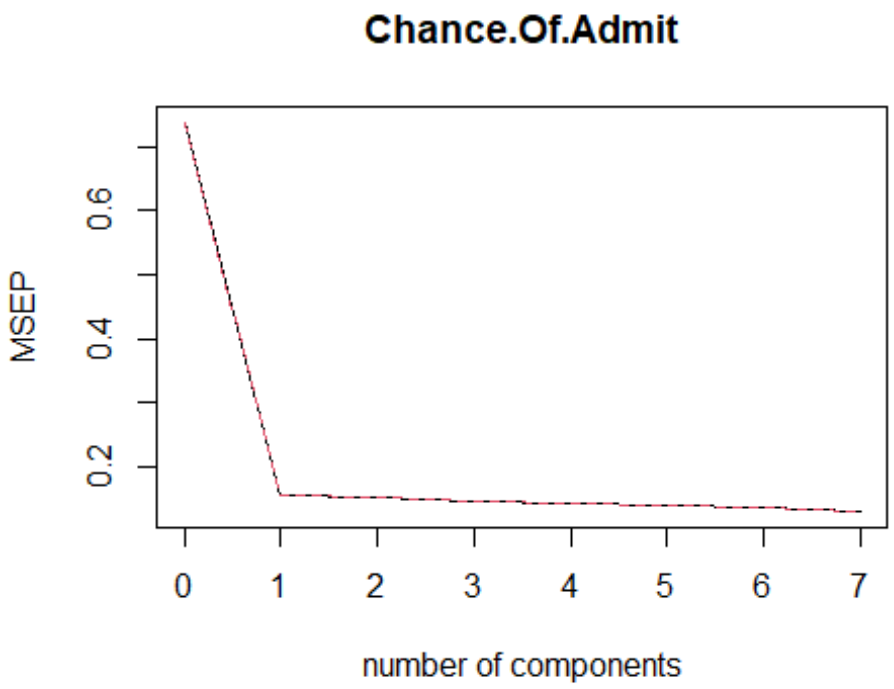
```
## [1] 0.3338271
```

```
set.seed(2)
pcr.fit <- pcr(Chance.Of.Admit~., data=new_df, scale=TRUE, validation = "CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 400 7
##  Y dimension: 400 1
## Fit method: svdpc
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8588   0.3934   0.3912   0.3796   0.3762   0.3742   0.3681
## adjCV       0.8588   0.3934   0.3911   0.3795   0.3760   0.3741   0.3684
##        7 comps
## CV      0.3606
## adjCV   0.3603
##
## TRAINING: % variance explained
##                  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7
comps
## X                  69.61    79.97    87.78    92.21    95.71    97.98
100.00
## Chance.Of.Admit    79.00    79.32    80.60    81.02    81.31    81.94
82.78
```

The setting scale=TRUE has the effect of standardizing each predictor prior to generating the principal components, so that the sacle on which each variable is measured will not have an effect. The setting validation="CV" causes the pcr() to compute the ten-fold cross validation error for each possible value of M.

```
# visualize cross-validation plots
validationplot(pcr.fit, val.type = "MSEP")
```




```
set.seed(2)
pcr_model <- pcr(Chance.Of.Admit~., data = train, scale = TRUE, validation =
"CV")
summary(pcr_model)
```

```
## Data:    X dimension: 300 7
##  Y dimension: 300 1
## Fit method: svdpc
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8789   0.4062   0.4038   0.3950   0.3907   0.3879   0.3822
## adjCV       0.8789   0.4060   0.4035   0.3947   0.3904   0.3874   0.3824
##        7 comps
## CV      0.3762
## adjCV   0.3756
##
## TRAINING: % variance explained
##                  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7
comps
## X                  66.73    78.44    86.63    91.28    95.38    97.85
100.0
## Chance.Of.Admit    78.92    79.25    80.27    80.68    81.33    81.76
82.6
```

- 1. VALIDATION: RMSEP

This table tells us the test RMSE calculated by the k-fold cross validation. We can see the following:

If we only use the intercept term in the model, the test RMSE is 0.8789 If we add in the first principal component, the test RMSE drops to 0.4062 If we add in the second principal component, the test RMSE drops to 0.4038 If we add in the third principal component, the test RMSE drops to 0.3950 If we add in the fourth principal component, the test RMSE drops to 0.3907

We can see that adding additional principal components after the 4 components does not actually decrease the RMSE by much. Thus, it appears that it would be optimal to only use four principal components in the final model.

- TRAINING: % variance explained

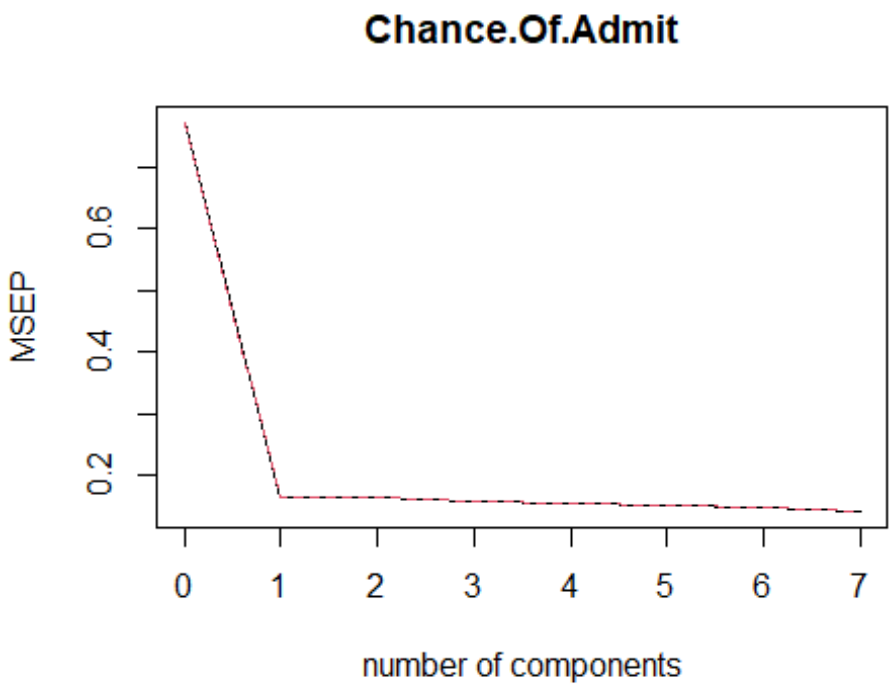
This table tells us the percentage of the variance in the response variable explained by the principal components. We can see the following:

By using just the first principal component, we can explain 66.73% of the variation in the response variable. By adding in the second principal component, we can explain 78.44% of the variation in the response variable. By adding in the third principal component, we can explain 86.63% of the variation in the response variable. By adding in the fourth principal component, we can explain 91.28% of the variation in the response variable.

Note that we'll always be able to explain more variance by using more principal components, but we can see that adding in more than four principal components doesn't actually increase the percentage of explained variance by much.

Visualizing the test RMSE (along with the test MSE and R-squared) based on the number of principal components by using the validationplot() function.

```
# visualize cross-validation plots
validationplot(pcr_model, val.type = "MSEP")
```




```
#Use the Final Model to Make Predictions
pcr.pred <- predict(pcr_model, test, ncomp = 7)
pcr_pred <- predict(pcr_model, test, ncomp = 4)

#calculate RMSE
rmse(actual = y_test, predicted = as.numeric(pcr.pred))

## [1] 0.3338271

rmse(actual = y_test, predicted = as.numeric(pcr_pred))

## [1] 0.3439025
```

Comparing the root mean square error using using all 7 components and 4 components, we see that using all M components is similar to performing least square regression with all p-features. Comparing the root mean square error when using all 7 components and using 4 components, we realize that the root mean square error using 4 components is just slightly higher compared to using all 7 components.

```
# Fitting the pcr on the full data set using M=4
pcrfit <- pcr(Chance.Of.Admit~., data = new_df, scale=TRUE, ncomp=4)
summary(pcrfit)

## Data:    X dimension: 400 7
##  Y dimension: 400 1
## Fit method: svdpc
## Number of components considered: 4
## TRAINING: % variance explained
```

```
##                  1 comps  2 comps  3 comps  4 comps
## X                  69.61    79.97    87.78    92.21
## Chance.Of.Admit    79.00    79.32    80.60    81.02
```

PARTIAL LEAST SQUARES

In a dataset with labeled data, it is expected that the principal components align with the target variable, meaning that the directions of maximum variation in the input features are asociated with the target variable. when this is not the case, principal components may not accurately capture the underlying relationship between the input features and the target variable. Partial least squareds is a supervised alternative to PCR. Partial least squares makes use of the response Y in order to identify new features that not only approximate the old features well, but also that are related to the response.

```
set.seed(2)
pls.fit <- plsr(Chance.Of.Admit~., data=train, scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 300 7
##  Y dimension: 300 1
## Fit method: kernelpls
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8789   0.4013   0.3819   0.3777   0.3770   0.3762   0.3762
## adjCV       0.8789   0.4011   0.3815   0.3772   0.3764   0.3756   0.3757
##        7 comps
## CV      0.3762
## adjCV   0.3756
##
## TRAINING: % variance explained
##                  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7
comps
## X                  66.71    75.00    81.99    89.19    93.24    97.08
100.0
## Chance.Of.Admit    79.48    81.79    82.42    82.57    82.60    82.60
82.6
```

We can see that adding additional principal components does not actually decrease the cross validation error by much after 3 principal components. Thus, it appears that it would be optimal to only use three principal components in the final model.

```
# Evaluate the corresponding test set RMSE
pls.pred <- predict(pls.fit, test, ncomp = 2)
rmse(actual = y_test, predicted = as.numeric(pls.pred))
```

```
## [1] 0.3344521
```

The test RMSE is comparable to, and slightly lower than the test RMSE obtained using PCR. We then perform partial least squares using the full data set, using M=3, the number of components identified by cross-validation.

```
pls_fit <- plsr(Chance.Of.Admit~., data=new_df, scale=TRUE, ncomp=2)
summary(pls_fit)
```

```
## Data:    X dimension: 400 7
##  Y dimension: 400 1
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##                   1 comps  2 comps
## X                   69.59    77.32
## Chance.Of.Admit     79.53    81.94
```

- Notice that the percentage of variance in Chance of Admission that the two-components PLS fit explains , 81.94%, is almost as much as that explained using the final 4 component model PCR fit, 81.02%. This is because PCR only attempts to maximize the amount of variance explained in the predictors, while PLS searches for directions that explain variance in both the predictors and response.
- Two-component PLS model is slightly more effective in explaining the variance in "Chance of Admission" compared to the four-component PCR model, as it explains a higher percentage of the variability in the target variable.