WK6_HW6

Anon Skywalker

9/29/2020

Question 9.1 Part 1

Creating a PCA model and using it for linear regression

To begin with this question, we can utilize the prcomp() function in R to apply PCA to our crime data. I used scale = true to scale the data. This function provides us with the standard deviations (higher the better) for each of our PC components. This helps us decide which components to use with our regression model.

By plotting our pca components, we can get an idea of where the more important variables are. From this plot, we can see that the first 4-5 PCs are the most significant, so I will be using 5 PCs with my regression model.

The scale = True parameter of the function scaled the data into standard deviations from the mean. This is extremely important for PCA due to its sensitivity of mismatched units of measure. By scaling, we can maintain the data point positions and convert their units of measure into something standardized for PCA to work with.

After running our pca through lm() we can get our regression model using the 5 chosen pc variables. The summary of this presented an AIC of 662 (lower the better), indicating that my model from last week with a AIC of 640 was superior, but we will dive further into this to verify.

```
pca <- prcomp(crime.data[, -16], scale = TRUE, center = TRUE)

summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142
##                            PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     0.55444 0.48493 0.44708 0.41915 0.35804 0.26333 0.2418
## Proportion of Variance 0.02049 0.01568 0.01333 0.01171 0.00855 0.00462 0.0039
## Cumulative Proportion  0.94191 0.95759 0.97091 0.98263 0.99117 0.99579 0.9997
##                           PC15
## Standard deviation     0.06793
## Proportion of Variance 0.00031
## Cumulative Proportion  1.00000
```

```r
#combine the chosen PC Scores to the original crime.data
PCA.crime.data <- as.data.frame(cbind(pca$x[, 1:5], crime.data[,16]))


#Perform CV on
lm.pca <- lm(formula = V6~., data = PCA.crime.data)
summary(lm.pca)
```

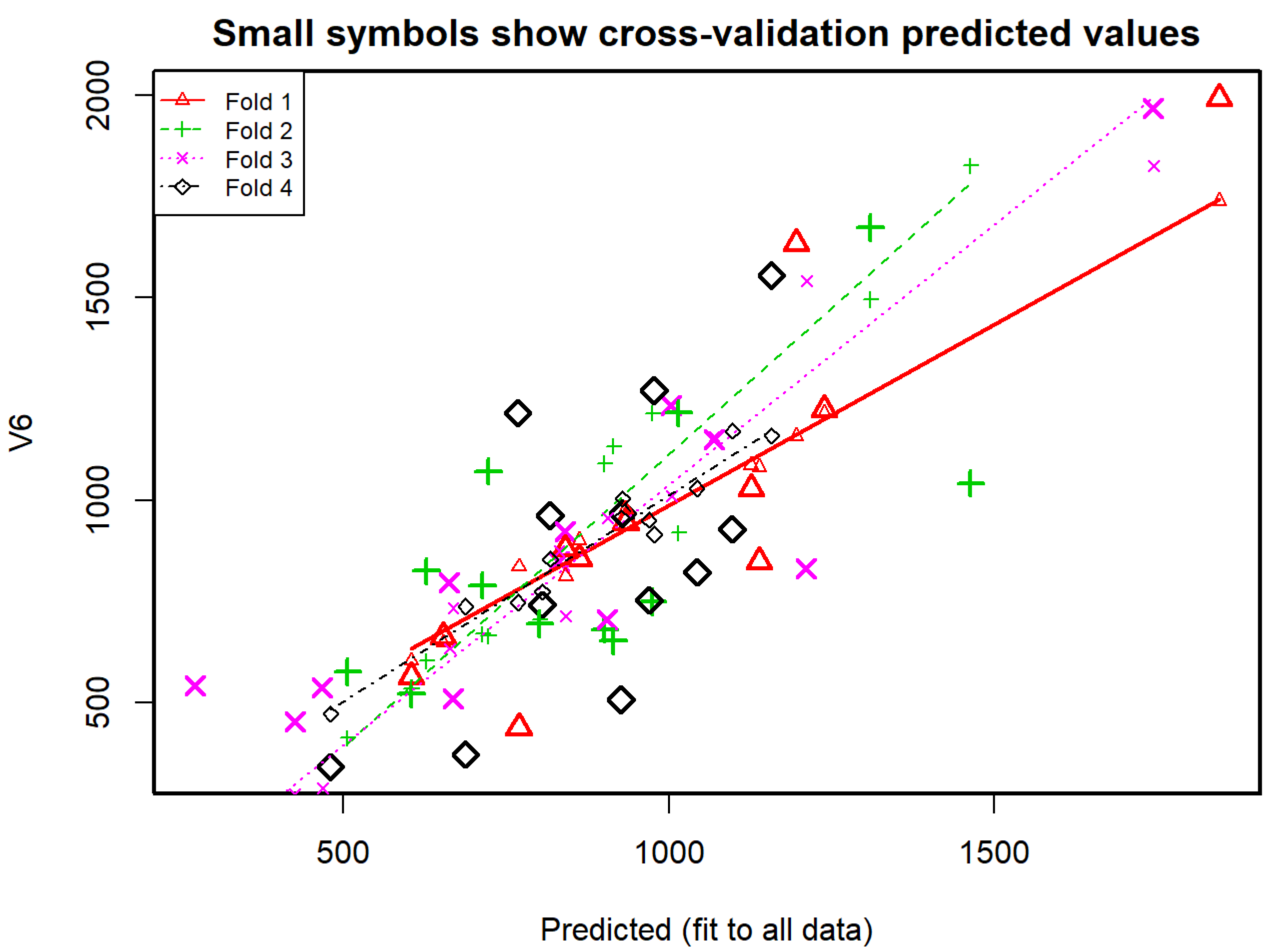
```
##
## Call:
## lm(formula = V6 ~ ., data = PCA.crime.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -420.79 -185.01   12.21  146.24  447.86
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      35.59  25.428  < 2e-16 ***
## PC1            65.22      14.67   4.447 6.51e-05 ***
## PC2           -70.08      21.49  -3.261  0.00224 **
## PC3            25.19      25.41   0.992  0.32725
## PC4            69.45      33.37   2.081  0.04374 *
## PC5          -229.04      36.75  -6.232 2.02e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 244 on 41 degrees of freedom
## Multiple R-squared:  0.6452, Adjusted R-squared:  0.6019
## F-statistic: 14.91 on 5 and 41 DF,  p-value: 2.446e-08
```

```r
#check model quality with AIC
AIC(lm.pca)
```

```
## [1] 657.703
```

```r
# cross validate using 4 folds
cv.pca <- cv.lm(lm.pca, data = PCA.crime.data, m=4)
```

```
## Analysis of Variance Table
##
## Response: V6
##           Df  Sum Sq Mean Sq F value  Pr(>F)    
## PC1        1 1177568 1177568   19.78 6.5e-05 ***
## PC2        1  633037  633037   10.63  0.0022 ** 
## PC3        1   58541   58541    0.98  0.3272    
## PC4        1  257832  257832    4.33  0.0437 *  
## PC5        1 2312556 2312556   38.84 2.0e-07 ***
## Residuals 41 2441394   59546                    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



```
##
## fold 1
## Observations in test set: 11
##                2     9    14    16      20   22   26    38    41     44   47
## Predicted   1196 862.7 653.8 933.8 1238.85  770 1846 604.3 841.5 1126.3 1139
## cvpred      1160 901.3 648.7 979.1 1215.03  837 1739 604.1 812.7 1087.8 1082
## V6          1635 856.0 664.0 946.0 1225.00  439 1993 566.0 880.0 1030.0  849
## CV residual  475 -45.3  15.3 -33.1    9.97 -398  254 -38.1  67.3  -57.8 -233
##
## Sum of squares = 516043    Mean square = 46913    n = 11
##
## fold 2
## Observations in test set: 12
##               1   3    6   11   19  25   28   29    30   33   35  39
## Predicted   714 506  901 1310  975 604 1015 1464 801.6  723  915 628
## cvpred      670 414 1091 1496 1216 535  920 1827 706.7  666 1134 604
## V6          791 578  682 1674  750 523 1216 1043 696.0 1072  653 826
## CV residual 121 164 -409  178 -466 -12  296 -784 -10.7  406 -481 222
##
## Sum of squares = 1604868    Mean square = 133739    n = 12
##
## fold 3
## Observations in test set: 12
##                4    5   10    12   13  15  17  34   37      40     42  45
## Predicted   1745 1004  906 831.7  669 663 468 842 1212 1069.89 272.25 425
## cvpred      1827 1010  957 875.1  734 634 289 714 1542 1157.89   7.74 275
## V6          1969 1234  705 849.0  511 798 539 923  831 1151.00 542.00 455
## CV residual  142  224 -252 -26.1 -223 164 250 209 -711   -6.89 534.26 180
##
## Sum of squares = 1140950    Mean square = 95079    n = 12
##
## fold 4
## Observations in test set: 12
##               7    8   18    21   23     24   27   31   32   36   43   46
## Predicted   818 1158 1098 805.8  768  929.0  480  688  970  978 1043  927
## cvpred      854 1159 1171 774.4  748 1003.8  473  738  951  916 1030  957
## V6          963 1555  929 742.0 1216  968.0  342  373  754 1272  823  508
## CV residual 109  396 -242 -32.4  468  -35.8 -131 -365 -197  356 -207 -449
##
## Sum of squares = 1009279    Mean square = 84107    n = 12
##
## Overall (Sum over all 12 folds)
##    ms
## 90875
```

```
#calculate for r-squared
1 - attr(cv.pca,"ms")*nrow(crime.data) / sum((crime.data$Crime - mean(crime.data$Crime))^2)
```

```
## [1] 0.379
```

Question 9.1 Part 2

[Get original variables for model Coefficients/model comparison]

For this section of the hw, the challange was to revert the model coefficients we received from the prcomp() function. To do this, I found the betas of the PCA model through its coefficients and multiplied this against its rotation (matrix of columns that contain the eigenvectors) in order to get the alpha values. However, these were NOT the original alphas, as they are still using scaled data. To resolve this problem, I just had to divide the alphas by sigma to get the original alpha values. Then it was just down to taking the beta0 (index of the pca coefficients) and subtracting that from the sum of the alphas multiplied by MU and then divided by sigma to get the initial beta value.

After this, it was time to put it all together and on to performing estimates with my renewed pca model! I used a new data frame called compare.df to put the original crime results from crime.data and the estimated crimes from the model against one another for better viewing.

Last part for this step was to take the r-squared/adjusted r-squared to verify model quality. This came out to be 0.645 with an adj r-squared of 0.602. This matched the PCA model from step one and verifies the calculations done here to unscale the data back into original variables and use them for the model.

In the code below, you can also find the model represented in terms of the orignal coefficients (initial.alpha).

```
#beta0
beta.0 <- lm.pca$coefficients[1]
#betas
betas <- lm.pca$coefficients[-1]

pca.rotate <- pca$rotation[,1:5]
#Convert the betas/PC Coefficients back to original variables
pca$rotation[,1:5]
```

```
##            PC1      PC2       PC3     PC4     PC5
## M      -0.3037  0.06280  0.172420 -0.0204 -0.3583
## So     -0.3309 -0.15837  0.015543  0.2925 -0.1206
## Ed      0.3396  0.21461  0.067740  0.0797 -0.0244
## Po1     0.3086 -0.26982  0.050646  0.3333 -0.2353
## Po2     0.3110 -0.26396  0.053065  0.3519 -0.2047
## LF      0.1762  0.31943  0.271530 -0.1433 -0.3941
## M.F     0.1164  0.39434 -0.203162  0.0105 -0.5788
## Pop     0.1131 -0.46723  0.077021 -0.0321 -0.0832
## NW     -0.2936 -0.22801  0.078816  0.2393 -0.3608
## U1      0.0405  0.00807 -0.659029 -0.1828 -0.1314
## U2      0.0181 -0.27971 -0.578501 -0.0689 -0.1350
## Wealth  0.3797 -0.07719  0.010065  0.1178  0.0117
## Ineq   -0.3658 -0.02752 -0.000294 -0.0807 -0.2167
## Prob   -0.2589  0.15832 -0.117673  0.4930  0.1656
## Time   -0.0206 -0.38015  0.223566 -0.5406 -0.1476
```

```
alphas <- pca$rotation[,1:5] %*% betas

initial.alpha <- alphas/sapply(crime.data[,-16],sd)#sigma
initial.beta <- beta.0 - sum(alphas*sapply(crime.data[,-16],mean) /
                                    sapply(crime.data[,-16],sd))

#view the initial coefficients for our pca model using unscaled data
t(initial.alpha)
```

```
##         M So   Ed  Po1  Po2   LF  M.F  Pop   NW  U1   U2 Wealth Ineq  Prob Time
## [1,] 48.4 79 17.8 39.5 39.9 1887 36.7 1.55 9.54 159 38.3 0.0372 5.54 -1524 3.84
```

```
#view the initial beta0 for equation
initial.beta
```

```
## (Intercept)
##       -5934
```

```
#perform estimates for the model
crime.est <- as.matrix(crime.data[,-16]) %*% initial.alpha + initial.beta

#A new data frame to check out how close the model estimated crime!
compare.df <- data.frame(crime.est, crime.data$Crime)
compare.df
```

```
##    crime.est crime.data.Crime
## 1        714              791
## 2       1196             1635
## 3        506              578
## 4       1745             1969
## 5       1004             1234
## 6        901              682
## 7        818              963
## 8       1158             1555
## 9        863              856
## 10       906              705
## 11      1310             1674
## 12       832              849
## 13       669              511
## 14       654              664
## 15       663              798
## 16       934              946
## 17       468              539
## 18      1098              929
## 19       975              750
## 20      1239             1225
## 21       806              742
## 22       770              439
## 23       768             1216
## 24       929              968
## 25       604              523
## 26      1846             1993
## 27       480              342
## 28      1015             1216
## 29      1464             1043
## 30       802              696
## 31       688              373
## 32       970              754
## 33       723             1072
## 34       842              923
## 35       915              653
## 36       978             1272
## 37      1212              831
## 38       604              566
## 39       628              826
## 40      1070             1151
## 41       841              880
## 42       272              542
## 43      1043              823
## 44      1126             1030
## 45       425              455
## 46       927              508
## 47      1139              849
```

```
#calc for r-squared

sse <- sum((crime.est - crime.data[,16])^2)
sst<- sum((crime.data$Crime - mean(crime.data$Crime))^2)
r.squared <- 1 - sse/sst
r.squared
```

```
## [1] 0.645
```

```
#adjusted r squared to compare with lm.pca model
r.adj <- r.squared - (1 - r.squared)*5/(nrow(crime.data)-5-1)
r.adj
```

```
## [1] 0.602
```

```
#its a match!



#view the initial coefficients for our pca model using unscaled data
orig.coefficients <- t(initial.alpha)
orig.coefficients
```

```
##         M So   Ed  Po1  Po2   LF  M.F  Pop   NW  U1   U2 Wealth Ineq  Prob Time
## [1,] 48.4 79 17.8 39.5 39.9 1887 36.7 1.55 9.54 159 38.3 0.0372 5.54 -1524 3.84
```

```
#view the initial beta0 for equation
initial.beta
```

```
## (Intercept)
##       -5934
```

Question 9.1 Part 3

[Predict using model, Compare with previous model/conclusion]

Last step involves using our test city to predict its crime. This step was pretty straightforward and only required that I create a new data frame using the pca values from our prcomp() function in order to apply the model for our prediction. The prediction for crime in the new city came out to be 1389, which is 85 higher than last weeks result.

In comparison with last weeks model, where I ended with a AIC of 640 and a r-squared value of 0.671 this weeks model seems to be less accurate. With a new r-squared of 0.645 and a AIC of 662, This indicates that the previous model was actually more accurate than this model despite the pca magic. After some research into why this may be the case, it turns out that the binary (0/1) data in column 2 of our uscrime data set is something that can throw off PCA to some extent. While there was some ways around this that I found online, it is beyond the scope of this assignment, but something to know for future situations. Another method could be to simply remove this column and work without it to ensure that it doesn’t negatively impact our models accuracy.

```
test.point <- data.frame(M = 14.0, So = 0, Ed = 10.0, Po1 = 12.0, Po2 = 15.5,
                         LF = 0.640, M.F = 94.0, Pop = 150, NW = 1.1, U1 = 0.120,
                         U2 = 3.6, Wealth = 3200, Ineq = 20.1, Prob = 0.040, Time = 39.0)

prediction.df <- data.frame(predict(pca, test.point))

#create new df of pca values in order to use pca model
crime.pred <- predict(lm.pca, prediction.df)
#crime prediction for new city
crime.pred
```

```
##    1
## 1389
```