ISYE 6501 Homework 4

Due: June 17, 2021

1 Principal Component Analysis

1.1 Question 9.1

Using the same *uscrime* dataset from previous works, I aimed to create a new, and ideally, better regression model using principal component analysis (PCA). In a similar light to using linear regression and hand picking specific predictors based on a threshold of their p-values, I first completed PCA and extracted the first few principal components before building a refined regression model. First, I examined the PCA results after using *prcomp* in R on all of the data. Note that the data was scaled before PCA.

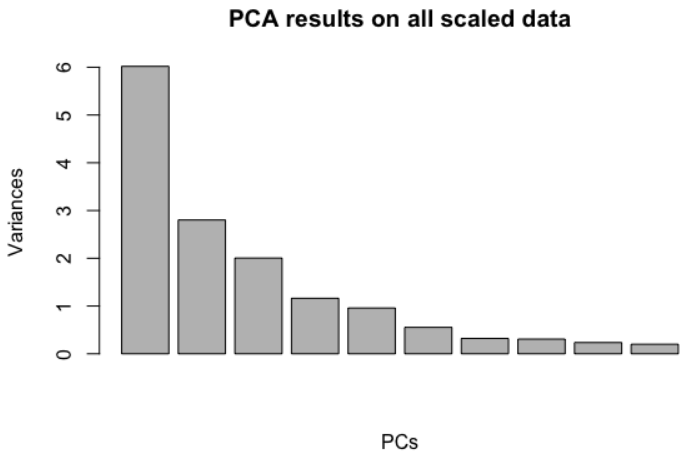

Figure 1: Visual inspection of the first few PCs and their variances.

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8
Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729 0.55444
Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145 0.02049
Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142 0.94191
```

Figure 2: A brief summary of the first few PCs

As you can see in Figures 1-2, the first few PCs account for most of the variance. Quantitatively, the first four PCs encompass $\approx 80\%$ of the variance as seen in the last row of Figure 2. I chose this as my threshold because reducing 15 dimensions to 4 is an appropriate when they explain this much variance. An additional reason I chose to build a model using the first four PCs is because my last prediction for the new test data point was using the predictors which had p ¡ 0.05. This happened to be four predictors and I am curious to see how the prediction varies from using four significant predictors to four PCs.

As mentioned, I then build a linear regression model using the first four PCs. Recall that the coefficients of this model are now in PC space and not in the frame of reference of the original 15 predictors. Also re-note that before PCA the data was scaled. Because of these two facts, in order to properly analyze our model and make accurate predictions using the test point in the original frame of reference, I needed to de-rotate and de-scale the new coefficients, effectively undoing the transformations to get them into the original space. This was done by matrix multiplying the rotation matrix (from PCA) and the scaled coefficients after linear regression of the PCs. Following this, the new coefficients were unscaled by dividing by the original standard deviations. Lastly, the intercept was unscaled by subtracting the sum of these divisions multiplied by the respective means of the original predictors and finally subtracting this value from the scaled intercept. See [1] and [2] for theory and Appendix A for code.

The model, with coefficients back in the original space, that I obtained using four PCs is shown in Equation 1. Plugging in the test point from the previous assignment yielding a crime rate = 1112.678. Interestingly, and expectedly, the prediction is different than that of linear regression using the most significant predictors. Although this has no rigor because we do not have the true target value, this crime rate appears to be closer to the mean of the other crime rates.

$$\begin{aligned} y = 1666.485 - 16.93076a_1 + 21.34368a_2 + 12.82972a_3 + 21.35216a_4 + 23.08832a_5 \\ -346.5657a_6 - 8.293097a_7 + 1.046216a_8 + 1.500994a_9 - 1509.935a_{10} \\ +1.688367a_{11} + 0.0400119a_{12} - 6.902022a_{13} - 144.9493a_{14} - 0.9330765a_{15} \end{aligned} \tag{1}$$

2 Regression Tress, Random Forests, and Logistic Regression

2.1 Question 10.1

Using the same *uscrime* dataset, I examined two tree-based models, a simple regression tree and a random forest. I used the *rpart* package in R and followed [3] for analysis and [4] for visualizations. All code can be found in Appendix B. First, I found a regression tree using default parameters to inspect its different to an optimized version. After cross-validation(cv), the default tree discovers three used variables NW, Po1, and Pop with a minimum cv error 0.8184. Figure 3, shows this tree and Figure 4 shows cv error varying with tree depth.

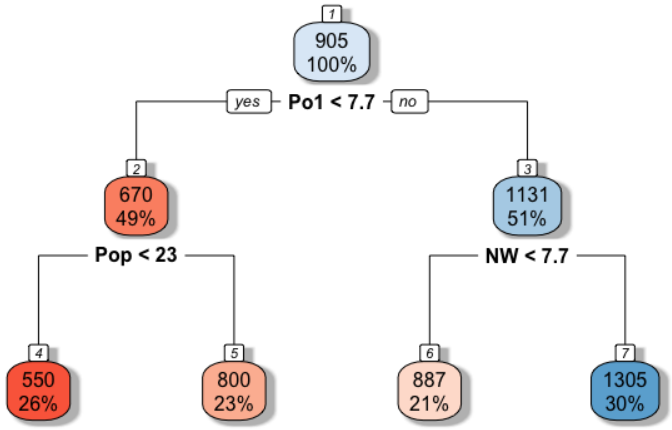

Figure 3: Regression tree using three predictors and having four leaves.

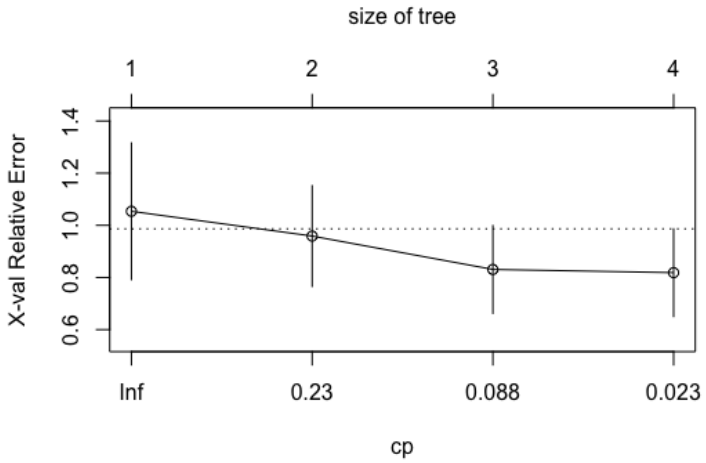

Figure 4: CV error with tree depth.

Figure 4 led me to believe a larger tree could perform better so I further optimized by allowing the tree to grow as large as each leaf containing ¿ 5% of the data. The results from this inspection are shown in Figure 5. Now, predictors Ed, M, NW, Po1, Pop, Prob, and Wealth are all used and the minimum cv error is 1.046041. This result has a few takeaways, although I initially thought the smaller tree was underfitting, it is clear after analysis that it was more appropriate than the deeper overfit tree.

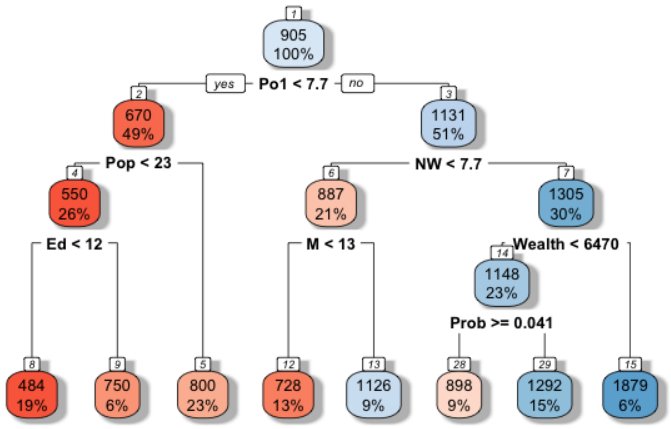

Figure 5: Regression tree using seven predictors and having eight leaves.

I followed up this analysis by building a random forest model and seeing how it predicts the crime rate given our data. Again default forests only use 500 trees and $m/5 = 3$ random predictors for each tree. The default tree only accounts for $\approx 39\%$ of the variance. After tuning parameters and optimizing, I found that increasing the number of trees to 1000 and the number of predictors to 4 accounted for the most variance ($\approx 43\%$) and had the lowest mean of squared residuals. As a quick and dirty visual analysis of this model I plotted the actual targets (*green*) and their respective predicted values (*red*) in Figure 6. As it can be seen the model does quite a good job likely because averaging over many trees using the same amount of predictors as the regression tree leads to a more accurate prediction.

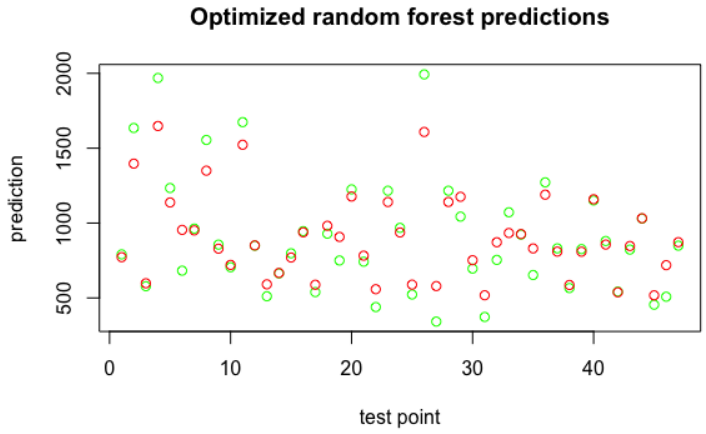

Figure 6: Actual targets (*green*) and their respective predicted values (*red*).

2.2 Question 10.2

A current event in which using a logistic regression model would be appropriate is predicting whether someone has Covid-19. The reason this model is appropriate is the model would output some probability that a person is infected and although a 50% could be used as a hard classifier, because of the cost function, one may wish to analyze the probability differently. For example one would much rather have the model give a false positive (quarantine a person it thought was sick but they were not) than a false negative (let a person it thought was not sick roam free when they are sick). In this way, the probability aspect becomes of huge importance. A few predictor the model could use are a person's: age, level/time of exposure, mask worn during exposure, and various medical history records.

2.3 Question 10.3

Additionally, I completed logistic regression on the *germancredit* dataset of 20 predictors and a binary output of either 1 (Good) or 2 (Bad). I used the *glm* function in R as seen in Appendix C. After running simple logistic regression on a training set comprised of 70% of the data, I found there to be eight significant coefficients for predictors (p ¡ 0.05). I then rerun the logistic regression model with just the eight significant predictors. Equation 2 show the mathematical model.

$$log\frac{p}{1-p} = -5.717 + 1.68a_1 + 0.0261a_2 + 1.711a_3 + 0.849a_4 + 0.0000566a_5 \\ +1.006a_6 + .1813a_7 + 0.9416a_8 \tag{2}$$

Similar to my discussion in the previous section, if a false positive is 5x worse than a false negative, one would want to create a cost function for analysis where the weight of the FP is a factor of 5 higher than that of the FN. One could then iteratively re-threshold the probability and calculate the expected cost by using model predictions until it is minimized. This would indeed be the optimal threshold for this logistic regression model.

References

[1] https://cdn-uploads.piazza.com/paste/kdyg5bmuj2n2v/ef0d98a3fd568f56233ec1c5147962cea31dfe2a4e79af22d7cbe62315

[2] https://cdn-uploads.piazza.com/paste/kdyg5bmuj2n2v/ec3b8702a968d3404740f62577a73a1e0b23f675addff6d74515b414c

[3] https://www.statmethods.net/advstats/cart.html

[4] https://blog.exploratory.io/visualizing-a-decision-tree-using-r-packages-in-explortory-b26d4cb5e71f

Appendix A PCA code

```
data <- read.table("uscrime.txt", header=TRUE)
crimes <- as.matrix(data[,1:15])

PCA <- prcomp(crimes, scale.=TRUE, retx=TRUE)
print(PCA)
summary(PCA)
screeplot(PCA, main='PCA results on all scaled data', xlab='PCs')
x4 <- PCA$x[,1:4]
rot <- PCA$rotation
#print(x4)
crimesPC4 <- as.data.frame(cbind(x4, data[,16]))
print(crimesPC4)
lr_PCA <- lm(V5~., data=crimesPC4)
summary(lr_PCA)
intercept <- lr_PCA$coefficients[1]
Bs <- lr_PCA$coefficients[2:5]
# rotation
As <- rot[,1:4] %*% Bs
t(As)
#de-scaling
descaled_A <- As / sapply(data[,1:15], sd)
descaled_intercept <- intercept - sum(As*sapply(data[,1:15], mean)/sapply(data[,1:15], sd))
t(descaled_A)
print(descaled_intercept)
#prediction of test point
test_point <- c(14.0, 0, 10.0, 12.0, 15.5, 0.640, 94.0, 150, 1.1,
                0.120, 3.6, 3200, 20.1, 0.04, 39.0)
print((t(as.matrix(descaled_A)) %*% as.matrix(test_point)) + descaled_intercept)
```

Appendix B Regression tree and Random forest code

```
library(rpart)
library(rpart.plot)
library(randomForest)
data_raw <- read.table("uscrime.txt", header=TRUE)
data <- as.data.frame(data_raw[,1:15])
response <- data_raw[,16]

reg_tree <- rpart(response~., data=data, method="anova",
                  control=rpart.control(minsplit=10))

printcp(reg_tree)
#plotcp(reg_tree)
summary(reg_tree)

#visualizations
rpart.plot(reg_tree)
rpart.plot(reg_tree, box.palette="RdBu", shadow.col="gray", nn=TRUE)
#plot(reg_tree, uniform=TRUE)# , main="Regression tree on crimes data")
```

```
#text(reg_tree, use.n=TRUE, all=TRUE)#, cex=.5)

preg_tree <- prune(reg_tree,
                   cp=reg_tree$cptable[which.min(reg_tree$cptable[,"xerror"]),"CP"])
rpart.plot(preg_tree, box.palette="RdBu", shadow.col="gray", nn=TRUE)

# random forest
rand_for <- randomForest(response~., data=data, ntree= 1000, mtry=4)
print(rand_for)
importance(rand_for)
pred <- predict(rand_for, data)
plot(response, col="green", main='Optimized random forest predictions',
     xlab='test point', ylab='prediction')
points(pred, col="red")
```

Appendix C Logistic Regression code

```
data_raw <- read.table("germancredit.txt", sep=' ')
#data <- data_raw[,1:20]
#response <- data_raw[,21] - 1

m <- nrow(data_raw)
train <- sample(1:m, size=round(m*0.7), replace = FALSE)
learn <- data_raw[train,]
val <- data_raw[-train,]

log_reg <- glm(V21 -1 ~., data=learn, family=binomial(link='logit'))
summary(log_reg)

pred <- predict(log_reg, val)

log_reg1 <- glm(V21 -1 ~ V1 + V2 +V3 +V4 +V5 +V6 +V8 +V10, data=learn,
                family=binomial(link='logit'))
summary(log_reg1)

pred <- predict(log_reg1, val, type="response")
```