Project Five for Data Mining

Raymond Anthony Ford
raford2@miners.utep.edu

Due: 17 April 2018

Contents

1 Data preparation

We begin this project by first bringing the data into R.

```
dat <- read.csv('~/Dropbox/Spring2018/STAT5474/crime.csv',
                header=TRUE, sep=',')
```

1.1 Preparing the data

We next remove the first five columns from the data, as they are not predictive.

```
dat <- dat[, -c(1:5)]
```

Next we inspect the variables to identify which ones are missing more than 60% of their values.

```r
vnames <- colnames(dat)
n <- nrow(dat)
out <- NULL
for (j in 1:ncol(dat)) {
    vname <- colnames(dat)[j]
    x <- as.vector(dat[,j])
    n1 <- sum(is.na(x), na.rm=TRUE)
    n2 <- sum(x=="NA", na.rm=TRUE)
    n3 <- sum(x=="", na.rm=TRUE)
    nmiss <- n1 + n2 + n3
    miss.perc <- nmiss/n
    if (miss.perc >= 0.60) {
        out <- rbind(out, c(vname, miss.perc))
    }
}
out <- as.data.frame(out)
row.names(out) <- NULL
colnames(out) <- c("Variable", "Missing Proportion")
out
```

```
##                 Variable Missing Proportion
## 1           LemasSwornFT  0.840020060180542
## 2        LemasSwFTPerPop  0.840020060180542
## 3      LemasSwFTFieldOps  0.840020060180542
## 4   LemasSwFTFieldPerPop  0.840020060180542
## 5          LemasTotalReq  0.840020060180542
## 6      LemasTotReqPerPop  0.840020060180542
## 7        PolicReqPerOffic  0.840020060180542
## 8             PolicPerPop  0.840020060180542
## 9     RacialMatchCommPol  0.840020060180542
## 10         PctPolicWhite  0.840020060180542
## 11         PctPolicBlack  0.840020060180542
## 12          PctPolicHisp  0.840020060180542
## 13         PctPolicAsian  0.840020060180542
## 14         PctPolicMinor  0.840020060180542
## 15   OfficAssgnDrugUnits  0.840020060180542
## 16     NumKindsDrugsSeiz  0.840020060180542
## 17      PolicAveOTWorked  0.840020060180542
## 18             PolicCars  0.840020060180542
## 19         PolicOperBudg  0.840020060180542
## 20   LemasPctPolicOnPatr  0.840020060180542
## 21   LemasGangUnitDeploy  0.840020060180542
## 22        PolicBudgPerPop  0.840020060180542
```

Now we need to find the column numbers of these variables to make their deletion easier. We accomplish this task with the following code.

```
delete.index <- NULL
for (k in 1:length(colnames(dat))) {
    if (colnames(dat)[k] %in% out$Variable) {
        delete.index <- c(delete.index, k)
    }
}
delete.index
```

```
##  [1]  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113
## [18] 117 118 119 120 122
```

Next, we delete those columns from the data set.

```
dat <- dat[,-c(delete.index)]
```

Before we impute any missing data, we need to make sure that our target variable `ViolentCrimesPerPop` does not have any missing values.

```
sum(is.na(dat$ViolentCrimesPerPop))
```

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

We see that `ViolentCrimesPerPop` does not have any missing values, so we proceed with the imputation.

```
library(mice, quietly=TRUE)
fit.mice <- mice(dat, m=1, maxit=50, method='pmm', seed=500, printFlag=FALSE)
dat.imp <- complete(fit.mice, 1)
```

1.2 Exploratory data analysis

We next perform some minor exploratory data analysis. We plot the response variable `ViolentCrimesPerPop` in the following histogram.

```
hist(dat.imp$ViolentCrimesPerPop, main="ViolentCrimesPerPop", xlab="Value")
box()
```

We see in Figure 1 that the response appears to be approximately exponentially distributed, but we do not perform a transformation on it.

1.3 Partitioning data

Now we partition the data set into a training and testing data set.

```
set.seed(124)
training.data.index <- sample(1:nrow(dat.imp), 0.667*nrow(dat.imp))
```

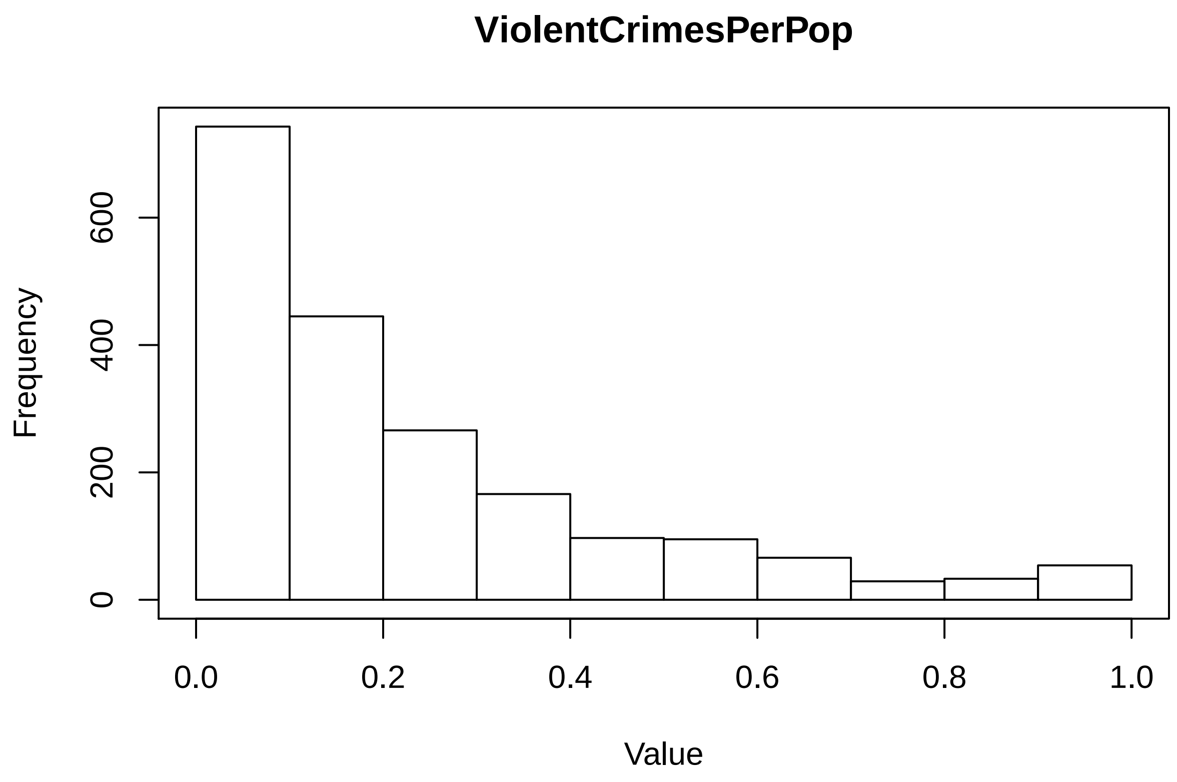

Figure 1: Histogram of our target variable ViolentCrimesPerPop.

```
training <- dat.imp[training.data.index, ]
test <- dat.imp[-training.data.index, ]
```

2 Predictive modeling

For this project we will fit five models and compare their performance with mean square error of prediction (MSEP).

2.1 Ridge regression

The first model that we fit is ridge regression. We need to select a tuning parameter, and this parameter is chosen with cross validation. The code for this model is below.

```
suppressMessages(library(glmnet, quietly=TRUE))
lambda <- seq(0, 80.0, 0.01)
X <- as.matrix(training[, -101]); y <- training[, 101]
cv.RR <- cv.glmnet(x=X, y=y, alpha = 0, lambda = lambda, nfolds=10)  # 10-FOLD CV BY DE
lmbd0 <- cv.RR$lambda.min # MINIMUM CV ERROR
fit.RR <- cv.RR$glmnet.fit
```

```
y.pred <- predict(fit.RR, s=lmbd0, newx = as.matrix(test[, -101]))
yobs <- test[,101]
MSEP.RR <- mean((yobs -y.pred)^2); MSEP.RR
```

```
## [1] 0.01948436
```

2.2 Principal components regression (PCR)

The second model that we fit is principal components regression. We need to determine the number of components that will yield the best model, and this is accomplished with cross validation. The code for this model is below.

```
suppressMessages(library(pls, quietly=TRUE))
fit.PCR.best <- pcr(training[,101] ~ ., ncomp=16, data=training, method = pls.options()$
CV <- fit.PCR.best$validation
ncomp.best <- which.min(CV$PRESS)
# PREDICTION
yhat.PCR <- predict(fit.PCR.best, newdata=test, comps=1:ncomp.best)
yobs <- test[,101]
# MEAN SQUARE ERROR FOR PREDICTION
MSEP.PCR <- mean((yobs-yhat.PCR)^2); MSEP.PCR
```

```
## [1] 0.01767888
```

2.3 Partial least squares regression (PLSR)

The third model that we fit is partial least squares regression. We need to determine the number of components that will yield the best model, and this is accomplished with cross validation. The code for this model is below.

```
suppressMessages(library(pls, quietly=TRUE))
fit.PLS <- plsr(ViolentCrimesPerPop ~ ., ncomp=100, data=training, method = "simpls",
    validation = "CV", segments = 10,  segment.type ="random", scale=TRUE)
CV <- fit.PLS$validation
ncomp.best <- which.min(CV$PRESS)
# MAKE PREDICTION
yhat.PLSR <- predict(fit.PLS, newdata=test, comps=1:ncomp.best)
yobs <- test[,101]
# MEAN SQUARE ERROR FOR PREDICTION
MSEP.PLSR <- mean((yobs-yhat.PLSR)^2); MSEP.PLSR
```

```
## [1] 0.01986281
```

2.4 Total least squares regression (TLSR)

The fourth model that we fit is total least squares regression. The code for this model is below.

```
suppressMessages(library(pracma, quietly=TRUE))
X <- as.matrix(training[, -101]); y <- training[, 101]
fit.TLS <- odregress(x=X, y=y)
beta <- fit.TLS$coeff
Xnew <- as.matrix(test[, -101])
yhat.TLS <- cbind(Xnew, 1) %*% beta
# MEAN SQUARE ERROR FOR PREDICTION
MSEP.TLS <- mean((yobs-yhat.TLS)^2); MSEP.TLS
```

```
## [1] 425.4965
```

2.5 Least angle regression

The fifth, and final, model that we fit is least angle regression. The code for this model is below.

```
suppressMessages(library(lars, quietly=TRUE))
X <- as.matrix(training[, -101]); y <- training[, 101]
fit.lar <- lars(X,y,type="lar", trace = FALSE, normalize = TRUE, intercept = TRUE)
# TO FIND OUT THE L1-NORM
BETA <- as.matrix(fit.lar$beta)
L1.norm <- function(x) sum(abs(x))
norm.L1 <- apply(BETA, 1, L1.norm)
norm.L1 <- norm.L1/max(norm.L1)
df <- as.vector(unlist(fit.lar$df))
Cp <- as.vector(fit.lar$Cp)
RSS <- as.vector(fit.lar$RSS)
lambda <- c(as.vector(fit.lar$lambda), 0)
b.lambda <- lambda[Cp==min(Cp)]
b.L1norm <- norm.L1[Cp==min(Cp)]
b.max.steps <- 5
yhat.lar <- predict(fit.lar,  s=b.max.steps, newx=as.matrix(test[, -101]))$fit
MSEP.lar <- mean((yobs-yhat.lar)^2); MSEP.lar
```

```
## [1] 0.02342362
```

3 Results

Finally, we can view the performance of the models as measured with mean square error of prediction (MSEP). These results are below.

```
results <- matrix(c("Method", "MSEP", "Ridge Regression", MSEP.RR, "PCR",
                    MSEP.PCR, "PLS", MSEP.PLSR, "TLS", MSEP.TLS, "LAR", MSEP.lar),
                  byrow=TRUE, ncol=2)
results
```

```
##      [,1]               [,2]
## [1,] "Method"           "MSEP"
## [2,] "Ridge Regression" "0.0194843605154931"
## [3,] "PCR"              "0.0176788810065814"
## [4,] "PLS"              "0.019862812238809"
## [5,] "TLS"              "425.496522596517"
## [6,] "LAR"              "0.0234236221277717"
```

We see that the performance of all of the methods are fairly similar except for total least squares regression that has a MSEP of 425. This is likely due to a bug in the code.[1]

4 Discussion

One interesting finding with this data set deals with the demographics of law enforcement officers. Some of the variables removed from the data set prior to fitting the model were the ethnicity of the police. It would be interesting to see if such data has been recorded in recent years and what impact, if any, it has had on the violent crime rate because it has been suggested in some political circles that a police force ethnically representative of the area it serves can decrease overall crime.

[1]This is something that I need to investigate further.