STAT 4360 Mini Project 5

Name: Jaemin Lee

Section 1: Answers to the specific question asked

Question 1

a) It is necessary to standardize the data so that all variables are given a mean of zero and a standard deviation of one. Some variables might dominate other variables. For example, if we measured salary in Euros or if we measured height in meters, then we'd get different classification results from what we get if those two variables are measured in dollars and feet. By standardizing, all variables will be on a comparable scale as all the variances will be standardized to 1.

b) Given in the code.

c) The confusion matrix for logistic regression is given below.

```
##         test.Y
## lr.pred  No Yes
##     No  897  44
##     Yes  44  15
```

Sensitivity = TP/(TP+FN) = 15/(15+44) = 0.25, specificity = TN/(TN+FP) = 897/(897+44) = 0.95, and overall misclassification rate = (44+44)/1000 = 0.088 = 8.8%. This means that the accuracy for logistic regression is 100-8.8 = 91.2%.

d) The confusion matrix for ridge is given below.

```
## [1] 1000

##       true
## pred   No Yes
##   No  924  54
##   Yes  17   5
```

Sensitivity = TP/(TP+FN) = 6/(6+53) = 0.10, specificity = TN/(TN+FP) = 923/(923+18) = 0.98, and overall misclassification rate = (53+18)/1000 = 0.071 = 7.1%. This means that the accuracy for ridge is 100-7.1 = 92.9%. Compared to the coefficients of logistic regression, one can definitely see that the coefficients of ridge regression has shrunk towards 0. This makes sense as ridge regression tries to regularize (or shrink) the coefficients so that the model doesn't overfit.

e) The confusion matrix for lasso is shown below.

```
##       true
## pred   No Yes
##   No  936  54
##   Yes   5   5
```

Sensitivity = TP/(TP+FN) = 5/(5+54) = 0.08, specificity = TN/(TN+FP) = 936/(936+5) = 0.99, and overall misclassification rate = (54+5)/1000 = 0.059 = 5.9%. This means that the accuracy for ridge is 100-5.9 = 94.1%. Just as what we observed in the ridge regression, the coefficients of estimates has shrunken compared to those of logistic regression. Lasso also tries to regularize coefficients towards 0, so that model doesn't overfit.

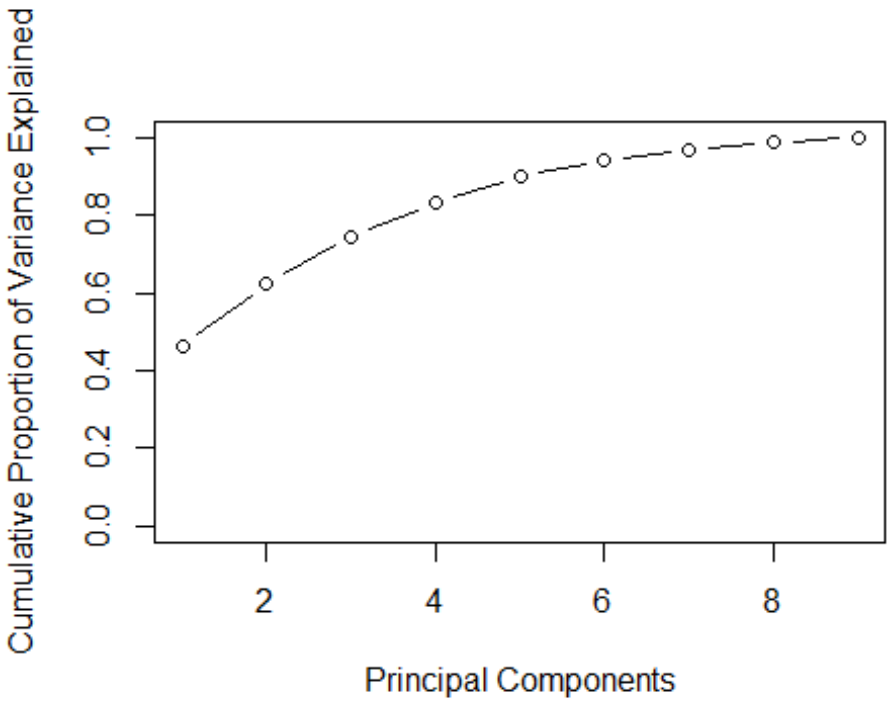
f) Based on the misclassification rates from c)-e), lasso delivers the lowest misclassification rate (5.9%), followed by ridge (7.1%) and logistic regression (8.8%). Considering that the insurance company cares about if customers would purchase the insurance plan, they may want to look at sensitivity. Sensitivity turned out to be the highest for logistic regression (0.25) and ridge and lasso were the lowest (both 0.08). If the company cares more about customers not purchasing the insurance, they they may look at the specificity. Specificity turned out to be the highest for lasso (0.99), followed by ridge (0.98) and logistic regression (0.95). Given that misclassification rates for all three of them were about the same, I would recommend using logistic regression as the company is likely to care about if the customers would purchase the insuruance plan and the sensitivity of logistic regression is the highest.

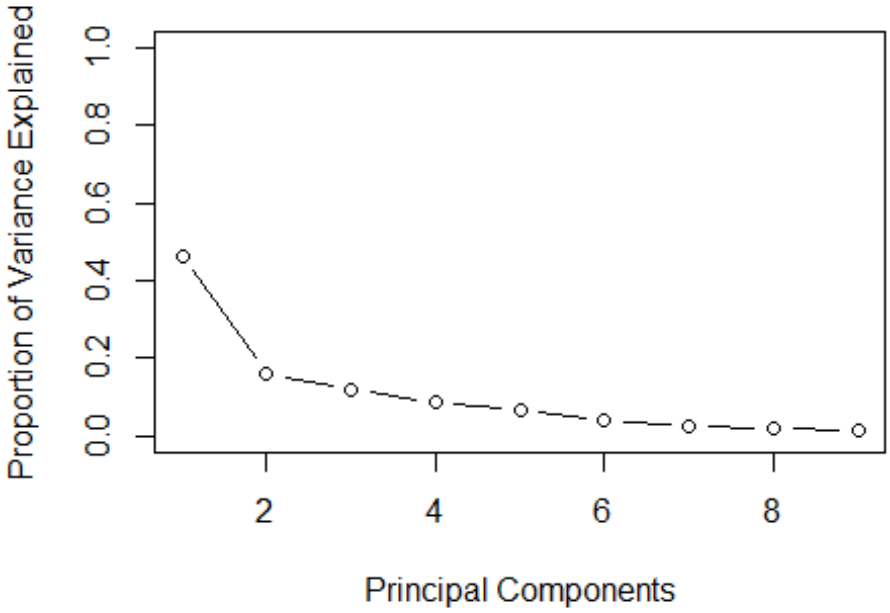
Question 2

a) Outliers are detected below and they are removed in the code.

b) Standardization is needed as the variance of variables are different. By standardizing the data set, variances are all equal to 1.

c) The cumulative Proportion of Variance Explained vs Principal Components plot shows that first 6 components explain 94% of the total variability in the data set and it seems to be a reasonable number given the screeplot.

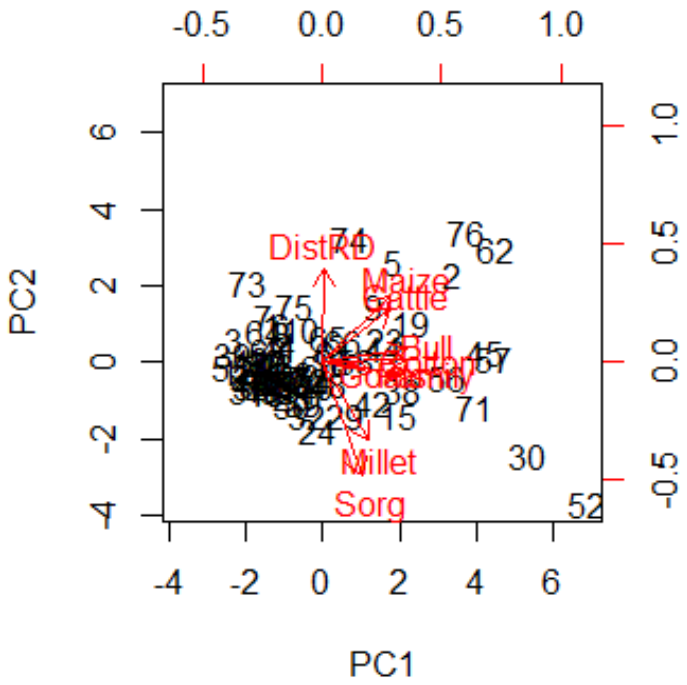




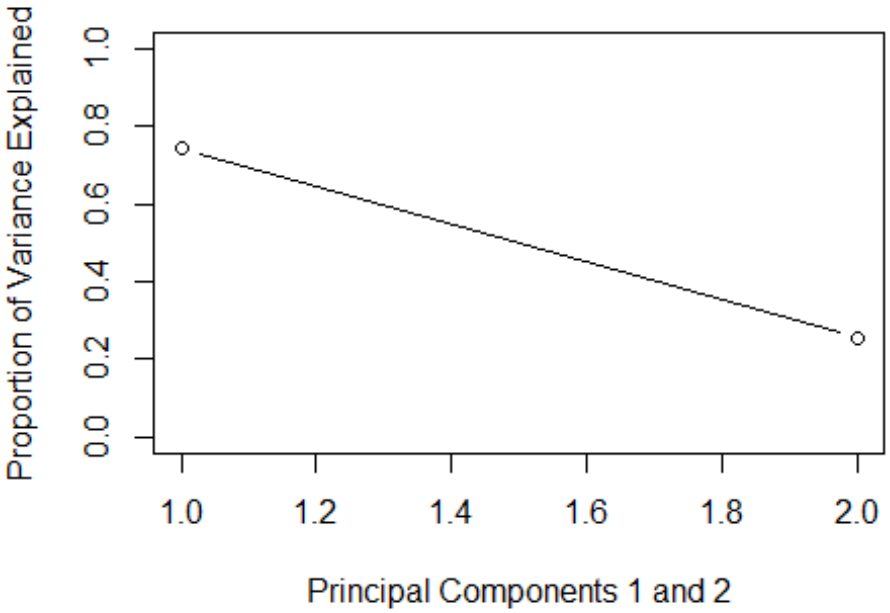
d) Below shows the correlations of standardized variables with the first and the second principal components.

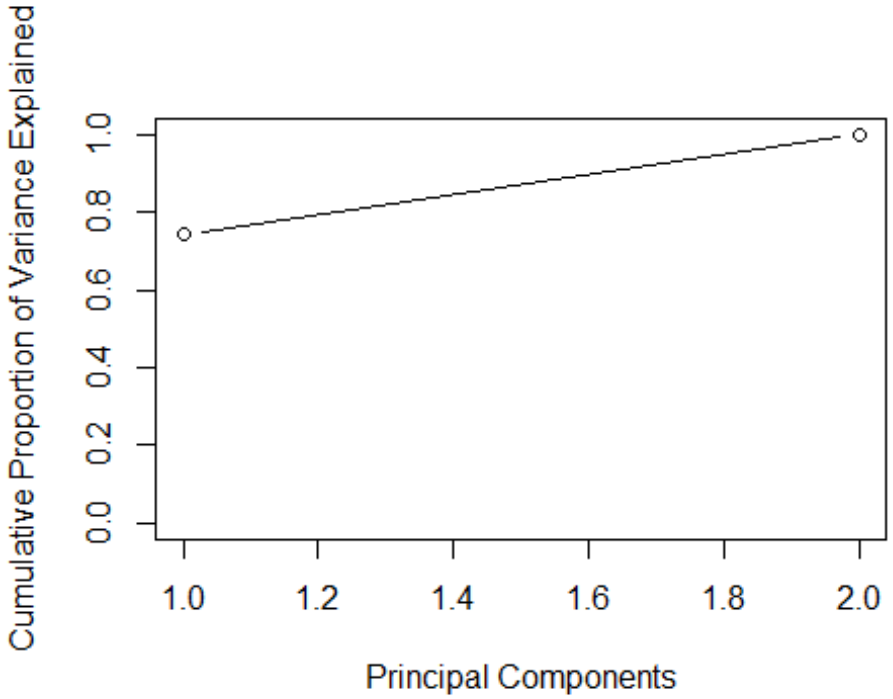
```
##             PC1          PC2
## 1  -1.30393150  0.997937492
## 2   3.38890329  2.289065005
## 3  -2.27679296  0.601201365
## 4  -0.91000528  0.505606461
## 5   1.86891203  2.614699367
## 6  -1.07293246  0.978670824
## 7  -1.51766264  1.193719433
## 8  -1.02200598  0.741119577
## 9   1.34244440  1.468147899
## 10 -0.59739906  0.858193282
## 11 -1.81903630  0.058104437
## 12 -1.53341661  0.056345705
## 13 -1.56968870 -0.700094472
## 14 -0.65032892 -0.747250828
## 15  1.93601846 -1.397451310
## 16 -1.39144099 -0.140300610
## 17 -0.76920659 -0.403753059
## 18 -2.16580770 -0.235598890
## 19  2.30459885  1.010409917
## 20 -1.41420842 -0.026425006
## 21 -1.76027288 -0.175811717
## 22 -0.34956763 -0.483273356
## 23  1.60517622  0.605145924
```

```
## 24 -0.14956323 -1.768533674
## 26 -1.84940513 -0.379449549
## 27 -1.00062374 -0.508348024
## 28 -0.78617265 -0.923003877
## 29  0.60966202 -1.402045737
## 30  5.30464872 -2.415470010
## 31 -1.95287973 -0.732608390
## 32 -0.39972997 -1.419275150
## 33 -0.79295797 -0.563819339
## 35 -1.81834706 -0.626423244
## 36  0.18749349  0.002721338
## 37 -1.26482993 -0.488058875
## 38  2.08312533 -0.756245687
## 39 -2.33680563  0.205383020
## 40 -1.37796526 -0.500958042
## 41 -1.13563154  0.389587079
## 42  1.28434683 -1.065521044
## 43 -1.23022312 -0.058032397
## 44  1.59976459  0.368275303
## 45  4.20089514  0.271888825
## 46 -1.52489770 -0.800720720
## 47 -1.92005139 -0.518910778
## 48  0.14018905 -0.565623819
## 49 -1.13295246 -0.628254294
## 50 -0.79426207 -1.135285728
## 51 -1.14867601 -0.739927553
## 52  6.84155345 -3.726393451
## 53 -1.48642891 -0.364459341
## 54 -0.24111256 -0.467607427
## 55 -2.41124468 -0.121868401
## 56  3.22891255 -0.406744200
## 57  4.41305579  0.106205703
## 58  0.86570556  0.011280239
## 59 -0.49742503 -1.057127191
## 60 -0.09246961 -0.136629876
## 61  0.31979498  0.264996109
## 62  4.47824881  2.937541122
## 63 -0.74305138 -0.281585558
## 64 -1.48721747  0.830760946
## 65  0.16439555  0.632159327
## 66  0.50862329  0.537842954
## 67 -1.08752106 -0.375598730
## 68 -1.40776960  0.331241897
## 70 -0.19748557 -0.634689449
## 71  3.93982583 -1.163111842
## 73 -1.93199041  2.048974819
## 74  0.67632855  3.225604330
## 75 -0.71215607  1.488159399
## 76  3.74292878  3.411301548
```



<Biplot of PC1 and PC2>





PC1 is defined by all the variables except DistRD. Based on the loading matrix of PC1, it represents that Family, Cotton, and Bull have higher weights (all above 0.4) and Maize, Sorg, Millet, and Goats are equally weighted (range from 0.2 to 0.35). This tells us that a farm defined by PC1 both harvests crops and raises cattles. PC2 is defined by all the variables except Family, Cotton, Bull, and Goats. Sorg is the variable that has the heighest weight (0.60), followed by DistRD (0.49), and Millet (0.41). PC2 tells us that a farm that is farther away from the road is likely to harvest large amount of crops such as Sorghum and Millet. Also, based on the biplot, one can see that Maize plays an important role in both PC1 and PC2 as its arrow forms a 45 degrees.

```
##                PC1          PC2         PC3         PC4         PC5
## Family 0.433842713 -0.065088695  0.09840025 -0.17120143  0.01132705
## DistRD 0.007587031  0.496670914 -0.56856059 -0.49561039 -0.37766811
## Cotton 0.446140316  0.008917253  0.13211700  0.02733684 -0.21870789
## Maize  0.352228405  0.352571495  0.38820350 -0.24020492 -0.07920345
## Sorg   0.203622111 -0.603667416 -0.11149246  0.05854254 -0.64457738
## Millet 0.240361102 -0.415159516 -0.11595977 -0.61632679  0.52696668
## Bull   0.445273680  0.068042477 -0.03038787  0.14559178 -0.02829987
## Cattle 0.355411548  0.284473439  0.01382636  0.37293370  0.21753184
## Goats  0.254549533 -0.048668251 -0.68695528  0.35078804  0.24867109
```

e) To determine which component explains “farm size”, one can look at the loadings of Cotton, Maize, Sorg, Millet. They represent hectares of those crops. Based on the loadings, PC1 and PC5 seem to explain farm size as their loadinns of Cotton, Maize, Sorg, Millet are higher than the other principal components. One can choose PC5, as the loading of distance from road is higher than that of PC1. Based on our observation

in part d) the farther away a farm is from the road, the bigger the farm size (or the land size) it has. Also, one can pick PC3 as a "goats and distance to road" component as the loadings of Goats and DistRD are the heighest in PC3 compared to the other principal components.

Question 3

a) - f) Given in the code.

g) Although all the methods below have low test error rates, I would recommend PLS. PLS yields the best outcome due to its error rate being the lowest (8.11%). This is because the variables in the Auto dataset are correlated to one another. PLS performs well when the variables have high correlations.

```
##                             LM Best.Subset    Ridge   Lasso     PCR    PLS
## Intercept              -35.814     -35.814 -28.1273 -35.294 -1.3780 -1.363
## cylinders                0.347       0.347  -0.0675   0.192 -1.4187 -1.412
## poly(displacement, 2)1  -5.217      -5.217 -18.1900   0.000  0.0528  0.353
## poly(displacement, 2)2   9.672       9.672  10.2874   8.535 -1.3690 -1.365
## poly(horsepower, 2)1   -43.683     -43.683 -38.1590 -40.883  0.2279  0.501
## poly(horsepower, 2)2    18.315      18.315  18.4755  17.476 -1.3559 -1.459
## poly(weight, 2)1       -71.146     -71.146 -48.8584 -73.207 -0.0130  0.263
## poly(weight, 2)2        15.645      15.645  13.2656  16.432  0.9425  0.742
## acceleration            -0.163      -0.163  -0.1333  -0.137  0.6615  1.018
## year                     0.783       0.783   0.7058   0.781  0.5192  0.428
## as.factor(origin)2       1.137       1.137   0.7853   1.137  0.6699  0.792
## as.factor(origin)3       1.217       1.217   1.2440   1.219 -1.4069 -1.117
## Test Error Rate          8.700       8.460   9.0458   8.893  8.4470  8.118
```

Section 2: R Code

1 a)

library(ISLR)

dim(Caravan) # 5822 observations with 86 variables (one of them being response)

str(Caravan)

head(Caravan$Purchase)

standardize the data

standardized.X = scale(Caravan[,-86])

str(standardized.X)

head(standardized.X)

var(Caravan[,1]) # 165

```
var(standardized.X[,1]) # notice it's standardized to 1
var(standardized.X[,2]) # standardized to 1

# add back the response (Purchase) to the standardized data
std.data = cbind(standardized.X, Caravan$Purchase)
std.data = data.frame(std.data)
colnames(std.data)[which(names(std.data) == "V86")] <- "Purchase"
str(std.data$Purchase)

# switch the purchase back to factor (Yes and no)
old.purchase = c('1','2')
new.purchase = factor(c('No', 'Yes'))
std.data$Purchase = new.purchase[match(std.data$Purchase, old.purchase)]
str(std.data$Purchase)
## 1 b)
test = 1:1000 # test ID
train.X = standardized.X[-test ,] # train X
str(train.X)
test.X  = standardized.X[test ,] # test x
str(test.X)
train.Y = Caravan$Purchase[-test] # train y
test.Y = Caravan$Purchase[test] # test y
## 1 c)
# fit a logistic regression model
# build model on train data
lr.fit = glm(Purchase ~ ., family = binomial, data = std.data, subset = -test)
coef(lr.fit)
```

```
# Estimated probabilities for test data
lr.prob = predict(lr.fit, std.data[test,], type = "response")
lr.prob
# replicate Nos 1000 times
#lr.pred = rep("No", 1000)

# Predicted classes (using 0.2 cutoff)
lr.pred = ifelse(lr.prob > 0.2, "Yes", "No")
# confusion matrix
table(lr.pred, test.Y)
#### 1.d)

library(glmnet)
# Create response and the design matrix (without the first column of 1s)
y = std.data$Purchase
str(y)
x = model.matrix(Purchase ~., std.data)[, -1]
length(x)
# set up a grid of lambda values in decreasing sequence
grid = 10^seq(10, -2, length = 100)

# relevel purchase to 1 and 2
std.data$Purchase = old.purchase[match(std.data$Purchase, new.purchase)]
y = as.numeric(std.data$Purchase)

# fit ridge regression (alpha = 0) for each lambda on the grid
ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)
```

```
coef(ridge.mod)
plot(ridge.mod, xvar = "lambda")
# find the best lambda using cross validation (binomial for the binary outcome variable)
set.seed(1)
cv.ridge <- cv.glmnet(x, y, alpha = 0, family = "binomial", type.measure = 'mse')
plot(cv.ridge)

############################## find the best lambda using test data

# fit model using training data
ridge.fit <- glmnet(x,y, alpha = 0, family = "binomial")

# best lambda = 0.06399002
best.lambda = cv.ridge$lambda.min;best.lambda

# Estimated probabilities for test data
# Get predictions on test set with best lambda and compute test MSE
x.test <- model.matrix(Purchase ~., std.data[test,])[,-1]
ridge.prob = predict(ridge.fit, x.test, s = best.lambda, type = "response")

# Predicted classes (using 0.2 cutoff)
ridge.pred = ifelse(ridge.prob > 0.2, "Yes", "No")


# fit model to training data
ridge.fit= glmnet(x,y, alpha = 0, family = "binomial", lambda = best.lambda)
#coef(model)
```

```
###

# Get predictions on test set with best lambda and compute test MSE
x.test <- model.matrix(Purchase ~., std.data[test,])[,-1]
ridge.prob = predict(ridge.fit, x.test, s = best.lambda, type = "response")
# convert probabilities to prediction
ridge.pred = ifelse(ridge.prob > 0.2, "Yes", "No")
length(ridge.pred)
table(pred = ridge.pred, true = test.Y)
################## 1 e)

# find the best lambda using cross validation (binomial for the binary outcome variable)
set.seed(1)
cv.lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial", type.measure = 'mse')
plot(cv.lasso)

# best lambda = 0.005565526
best.lambda.lasso = cv.lasso$lambda.min;best.lambda.lasso

# fit model to training data
lasso.fit= glmnet(x,y, alpha = 1, family = "binomial", lambda = best.lambda.lasso)
coef(lasso.fit)

###

# Get predictions on test set with best lambda and compute test MSE
```

#x.test <- model.matrix(Purchase ~., std.data[test,])[,-1]

lasso.prob = predict(lasso.fit, x.test, s = best.lambda.lasso, type = "response")

```

e) The confusion matrix for lasso is shown below.

```{r echo=FALSE}

# convert probabilities to prediction

library(caret)

lasso.pred = ifelse(lasso.prob > 0.2, "Yes", "No")

table(pred = lasso.pred, true = test.Y)

########### 2 a)

Mali = read.csv("C:/Users/jaemi/Desktop/STAT 4360/Projects/Project 5/mali.csv")

dim(Mali)

head(Mali)

# detect outliers

plot(Mali$Family, Mali$DistRD)

#boxplot(Family ~ DistRD, data = Mali)$out # 81, 52, 57 are outliers

#boxplot(DistRD ~ Cattle, data = Mali, plot = FALSE)$out # 80, 500, 500, 100, 100, 90

# remove outliers for Family vs DistRD

plot(Mali$Family,Mali$DistRD,xlab = "Family", ylab = "DistRD")

family.outlier = c(1:nrow(Mali))[Mali$Family > 140]

DistRD.outlier = c(1:nrow(Mali))[Mali$DistRD > 400]

outlier1 = c(family.outlier, DistRD.outlier)

text(Mali$Family[outlier1]+1, Mali$DistRD[outlier1]-15, labels = outlier1, cex=0.7)

```
# remove outliers for DistRd vs Cattle
plot(Mali$DistRD, Mali$Cattle, xlab = "DistRD", ylab = "Cattle")
DistRD.outlier2 = c(1:nrow(Mali))[Mali$DistRD > 400]
cattle.outlier = c(1:nrow(Mali))[Mali$Cattle > 80]
outlier2 = c(DistRD.outlier2, cattle.outlier)
text(Mali$DistRD[outlier2]+1, Mali$Cattle[outlier2]-3, labels=outlier2,cex=0.7)

# store data with no outlier
mali.no.outliers = Mali[-unique(c(outlier1, outlier2)),]
########## 2 b)
standardized.mali = scale(mali.no.outliers)
str(standardized.mali)
head(standardized.mali)
var(Mali[,1]) # 550
var(Mali[,2]) # 6533
var(standardized.mali[,1]) # notice it's standardized to 1
var(standardized.mali[,2]) # also 1
###########2 c)
pca.mali <- prcomp(standardized.mali, center = T, scale = T)
names(pca.mali)
pca.mali$center # check center
pca.mali$scale # all at 1
pca.mali$rotation # get the loading matrix

# get he score matrix
head(pca.mali$x)
```

```
# check covariance matrix of the scores

round(cov(pca.mali$x), 3)

# Display a biplot the results (shows both pc scores and loading vectors)

biplot(pca.mali, scale = 0)

# Compute the proportion of variance explained (PVE)

pc.var = pca.mali$sdev^2

pve.mali = pc.var/sum(pc.var)

pve.mali

cumsum(pve.mali)

# scree plot

plot(pve.mali, xlab = "Principal Components", ylab = "Proportion of Variance Explained",
ylim = c(0,1), type = 'b')

# plot of cumulative PVE

plot(cumsum(pve.mali), xlab="Principal Components", ylab = "Cumulative Proportion of
Variance Explained", ylim = c(0,1), type = 'b')

### 2 d)

pca.mali <- prcomp(standardized.mali, center = T, scale = T)


pca.mali

# extract loading matrix of 1st and 2nd PCs

pca.mali$rotation[,1:2]


# get he score matrix

head(pca.mali$x)


# check the calculation

#round(cov(pca$x), 3)
```

correlations of the standardized variables with the components and the cumulative percentage of the total variability explained by the two components.

correlation between PC1 and PC2

cor(pca.mali$rotation[,1], pca.mali$rotation[,2])

Compute the proportion of variance explained (PVE) using PC1 and PC2

pca.mali

pc.var.12 = (pca.mali$sdev[1:2])^2

pve.mali.12 = pc.var.12/sum(pc.var.12)

pve.mali.12

cumsum(pve.mali.12)

check covariance matrix of the scores

round(cov(pca.mali$x), 3)

get the scores for PC1 and PC2

get he score matrix

pca.mali$x[,1:2]

```

```{r echo=FALSE}

# Display a biplot the results (shows both pc scores and loading vectors)

#pca.mali$rotation[,1:2]

#pca.mali

biplot(pca.mali, scale = 0)

# scree plot using PC1 and PC2

plot(pve.mali.12, xlab = "Principal Components 1 and 2", ylab = "Proportion of Variance Explained", ylim = c(0,1), type = 'b')

```
cumulative percentage of the total variability explained by PC1 and PC2

plot(cumsum(pve.mali.12), xlab="Principal Components", ylab = "Cumulative Proportion of Variance Explained", ylim = c(0,1), type = 'b')

pca.mali$rotation[,1:5]

3 a)

library(ISLR)

str(Auto)

library(caret)

library(DAAG)

remove name variables

Auto = Auto[,-9]

fit a linear model with all variables

lm.fit = lm(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin), data = Auto)

lm.cv = train(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin), data = Auto, method = "lm", trControl=trainControl(method = 'cv', number = 10, savePredictions = TRUE, verboseIter = TRUE))

lm.cv$results # RMSE = 2.95 -> MSE = 8.7

lm.coef = coef(lm.fit);lm.coef

lm.test.error = 8.7

cv.lm = cv.lm(Auto, lm.fit, m = 10, seed = 1)

3 b)
```

```
library(leaps)

library(ISLR)


x = data.frame(model.matrix(lm.fit)[,-1])

y = Auto[,1]


# Total number of predictors in the data

totpred = ncol(x)


# best subset selection

best.sub = regsubsets(y ~ ., data = x, nvmax = totpred, method = "exhaustive")


# check adjusted R^2 to pick how many variables to use

best.sub.summary = summary(best.sub);best.sub.summary

plot(best.sub.summary$adjr2, xlab = "Model Number", ylab = "Adjusted R^2", pch = 19,
type = "b")

which.max(best.sub.summary$adjr2) # model 7 has the best R^2 = 0.861

best.sub.summary$adjr2


# lm reduced by best subset selection

lm.reduced.best.sub = lm(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) +
poly(weight, 2)+ acceleration + year + as.factor(origin), data = Auto)


# fit a linear model with cylinders, displacement, horsepower, weight, accelration, year,
and origin b/c these give you the biggest R^2

lm.cv.best.subset = train(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) +
poly(weight, 2) +year + as.factor(origin), data = Auto, method = "lm",
trControl=trainControl(method = 'cv', number = 10, savePredictions = TRUE, verboseIter =
TRUE))
```

```
lm.cv.best.subset$results # RMSE = 2.91 -> MSE = 8.46%


best.subset.test.error = 8.46

lm.reduced.best.sub.coef = coef(lm.reduced.best.sub)

### 3 c)

# ridge regression

library(glmnet)

y = Auto$mpg

x = model.matrix(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) +
poly(weight, 2) + acceleration +year + as.factor(origin), Auto)[,-1]


# Set up a grid of lambda values (from 10^10 to 10^(-2)) in decreasing sequence

grid <- 10^seq(10, -2, length = 100)


# Fit ridge regression for each lambda on the grid

ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)

plot(ridge.mod, xvar = "lambda")

# fit model using training data

ridge.mod = glmnet(x, y, alpha = 0, lambda = grid, thresh = 1e-12)


# use 10-fold cross validation to estimate test MSE from training data

set.seed(1)

cv.out <- cv.glmnet(x, y, alpha = 0)

plot(cv.out)


bestlam = cv.out$lambda.min; bestlam # 0.6487382
```

```
# Test MSE for the best value of lambda
ridge.pred = predict(ridge.mod, s = bestlam, newx = x)

# Refit the model on the full dataset
out <- glmnet(x, y, alpha = 0)

# Get estimates for the best value of lambda
ridge.coeff = predict(out, type = "coefficients", s = bestlam)

# initialize array to store 10 CV errors
cv.ridge.error = seq(1,10)

# create 10 folds CV
library(caret)
set.seed(1)
folds = createFolds(Auto$mpg, k=10)

# apply CV
i = 1
for(val in folds){
  # get sets in proper form
  new.y = Auto$mpg[-(c(val))]
  new.x = model.matrix(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) +
poly(weight, 2) + acceleration +year + as.factor(origin), Auto)[-(c(val)), -1]

  # fit ridge regression to training data
  test.ridge = glmnet(new.x, new.y, alpha = 0, lambda = bestlam)
```

```
  # get sets in proper form
  newx.model = model.matrix(mpg ~ cylinders + poly(displacement, 2) +
poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin),
Auto)[(c(val)), -1]

  # perform predictions on the fold that is left out
  test.ridge.pred = predict(test.ridge, s = bestlam, newx = newx.model)

  # store test error rate
  cv.ridge.error[i] = mean((test.ridge.pred - Auto$mpg[(c(val))])^2)

  # increment i
  i = i+1
}
# get avg test error rate
ridge.test.error = mean(cv.ridge.error);ridge.test.error
#### 3 d)
# fit model using training data
lasso.mod = glmnet(x, y, alpha = 1, lambda = grid, thresh = 1e-12)

# use 10-fold cross validation to estimate test MSE from training data
set.seed(1)
cv.out <- cv.glmnet(x, y, alpha = 1)
plot(cv.out)

bestlam = cv.out$lambda.min; bestlam # 0.6487382
```

```
# Test MSE for the best value of lambda
lasso.pred = predict(lasso.mod, s = bestlam, newx = x)

# Refit the model on the full dataset
lasso.out <- glmnet(x, y, alpha = 1, lambda = bestlam)

# Get estimates for the best value of lambda
lasso.coef = predict(lasso.out, type = "coefficients", s = bestlam)

library(caret)
# initialize array to store 10 CV errors
cv.lasso.error = seq(1,10)

# create 10 folds CV
set.seed(1)
folds = createFolds(Auto$mpg, k=10)

# apply CV
i = 1
for(val in folds){
  # get sets in proper form
  new.y = Auto$mpg[-(c(val))]
  new.x = model.matrix(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) +
poly(weight, 2) + acceleration +year + as.factor(origin), Auto)[-(c(val)), -1]

  # fit lasso regression to training data
  test.lasso = glmnet(new.x, new.y, alpha = 1, lambda = bestlam)
```

```
  # get sets in proper form
  newx.model = model.matrix(mpg ~ cylinders + poly(displacement, 2) +
poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin),
Auto)[(c(val)), -1]


  # perform predictions on the fold that is left out
  test.lasso.pred = predict(test.lasso, s = bestlam, newx = newx.model)


  # store test error rate
  cv.lasso.error[i] = mean((test.lasso.pred - Auto$mpg[(c(val))])^2)


  # increment i
  i = i+1
}
# get avg test error rate
lasso.test.error = mean(cv.lasso.error);lasso.test.error # 8.89
### 3 e)
library(pls)
library(ISLR)


x = model.matrix(lm.fit)[,-1]
y = Auto[,1]
set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

```
# fit PCR to training set using 10 fold CV

set.seed(1)

pcr.fit <- pcr(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin), data = Auto, subset = train, validation = "CV", segments = 10, scale = TRUE)

validationplot(pcr.fit, val.type = "MSEP")


# We see that lowest cross-validation error occurs when M = 9


#compute test MSE

pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 9)

pcr.test.error = mean((pcr.pred - y.test)^2);pcr.test.error #8.44


# Refit the model on the full dataset

pcr.fit <- pcr(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin), data = Auto, scale = TRUE, ncomp = 9)

pcr.coef = pcr.fit$coefficients[1:12];pcr.coef

### 3 f)


# fit PLS

set.seed(1)

pls.fit <- plsr(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) + poly(weight, 2) + acceleration +year + as.factor(origin), data = Auto, validation = "CV", segments = 10, scale = TRUE)

validationplot(pls.fit, val.type = "MSEP")


# We see that lowest cross-validation error occurs when M = 9


#compute test MSE
```

```
pls.pred <- predict(pls.fit, x[test, ], ncomp = 9)

pls.test.error = mean((pls.pred - y.test)^2);pls.test.error #8.11


# Refit the model on the full dataset

pls.fit <- plsr(mpg ~ cylinders + poly(displacement, 2) + poly(horsepower,2) + poly(weight,
2) + acceleration +year + as.factor(origin), data = Auto, scale = TRUE, ncomp = 9)

pls.coef = pls.fit$coefficients[1:12];pls.coef

### 3 g)

# tables from a)-f)

nums = c(-35.8141933, 0.3466165, -5.2168003, 9.6719319, -43.6834405, 18.3152127, -
71.1459336, 15.6451735, -0.1629983, 0.7825036, 1.1366002, 1.2166322)

best.sub.coef = as.numeric(nums);best.sub.coef


lm.result = as.data.frame(c(lm.coef[1:12], lm.test.error))

rownames(lm.result) = c("Intercept", "cylinders", "poly(displacement, 2)1",
"poly(displacement, 2)2", "poly(horsepower, 2)1", "poly(horsepower, 2)2", "poly(weight,
2)1", "poly(weight, 2)2", "acceleration", "year", "as.factor(origin)2", "as.factor(origin)3",
"Test Error Rate")

colnames(lm.result) = "LM"

lm.result


best.sub.result = as.data.frame(c(best.sub.coef[1:12], best.subset.test.error))

rownames(best.sub.result) = c("Intercept", "cylinders", "poly(displacement, 2)1",
"poly(displacement, 2)2", "poly(horsepower, 2)1", "poly(horsepower, 2)2", "poly(weight,
2)1", "poly(weight, 2)2", "acceleration", "year", "as.factor(origin)2", "as.factor(origin)3",
"Test Error Rate")

colnames(best.sub.result) = "Best Subset"

best.sub.result


ridge.result = as.data.frame(c(ridge.coeff[1:12], ridge.test.error))
```

```
rownames(ridge.result) = c("Intercept", "cylinders", "poly(displacement, 2)1",
"poly(displacement, 2)2", "poly(horsepower, 2)1", "poly(horsepower, 2)2", "poly(weight,
2)1", "poly(weight, 2)2", "acceleration", "year", "as.factor(origin)2", "as.factor(origin)3",
"Test Error Rate")

colnames(ridge.result) = "Ridge"

ridge.result


lasso.result = as.data.frame(c(lasso.coef[1:12], lasso.test.error))

rownames(lasso.result) = c("Intercept", "cylinders", "poly(displacement, 2)1",
"poly(displacement, 2)2", "poly(horsepower, 2)1", "poly(horsepower, 2)2", "poly(weight,
2)1", "poly(weight, 2)2", "acceleration", "year", "as.factor(origin)2", "as.factor(origin)3",
"Test Error Rate")

colnames(lasso.result) = "Lasso"

lasso.result


pcr.result = as.data.frame(c(pcr.coef[1:12], pcr.test.error))

rownames(pcr.result) = c("Intercept", "cylinders", "poly(displacement, 2)1",
"poly(displacement, 2)2", "poly(horsepower, 2)1", "poly(horsepower, 2)2", "poly(weight,
2)1", "poly(weight, 2)2", "acceleration", "year", "as.factor(origin)2", "as.factor(origin)3",
"Test Error Rate")

colnames(pcr.result) = "PCR"

pcr.result

pls.result = as.data.frame(c(pls.coef[1:12], pls.test.error))

rownames(pls.result) = c("Intercept", "cylinders", "poly(displacement, 2)1",
"poly(displacement, 2)2", "poly(horsepower, 2)1", "poly(horsepower, 2)2", "poly(weight,
2)1", "poly(weight, 2)2", "acceleration", "year", "as.factor(origin)2", "as.factor(origin)3",
"Test Error Rate")

colnames(pls.result) = "PLS"

pls.result

# combine all the dataframe

result = data.frame(lm.result, best.sub.result, ridge.result, lasso.result, pcr.result,
pls.result);result
```