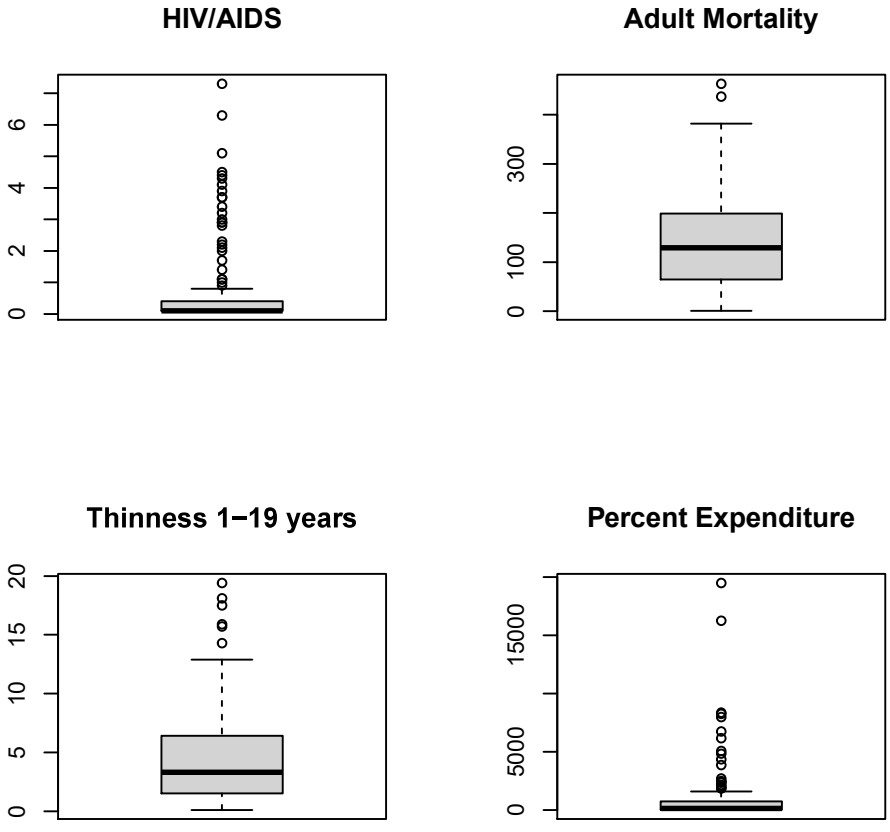
Life Expectancy and the Developing World (Midterm Project)

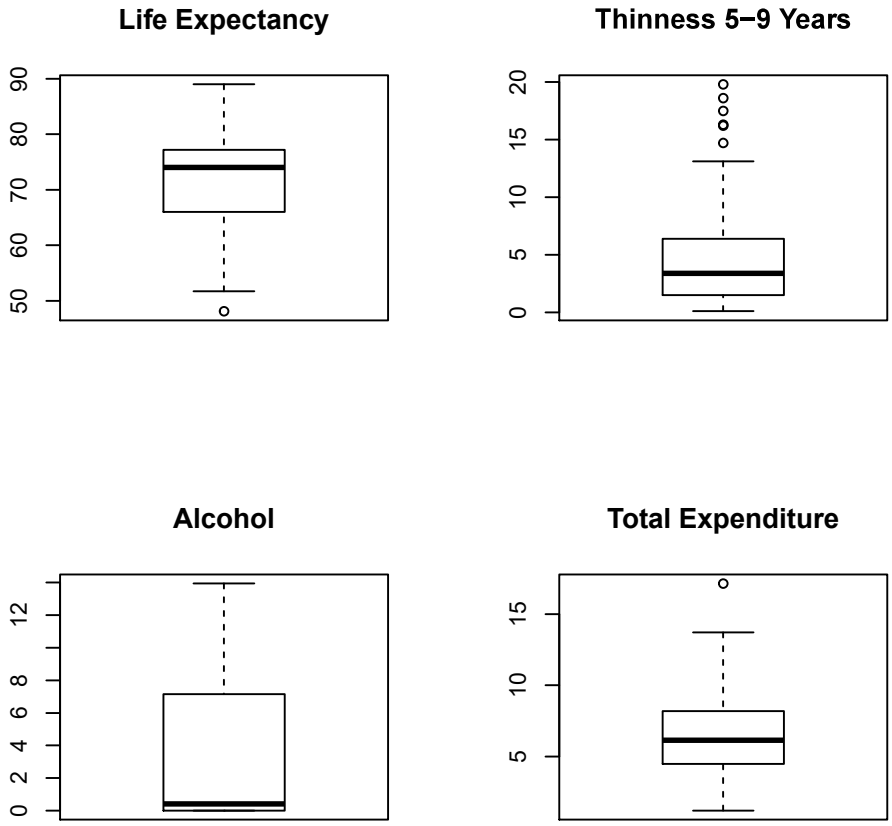
George Kuliner

February 28th, 2018

Introduction

For this project, the data set provided contains various statistics for different countries. The goal for our regression models is to obtain the lowest train and test mean squared error, for the prediction of **Life Expectancy**. For the classification models, the goal is to obtain the lowest train and test error rate for the prediction of **Status**. The status variable is either developed or developing.
Before we get into the models, it is useful to observe the predctors themselves.





Observing the graph for HIV/AIDS and percent expenditure, it is clear to see a varied distribution. Thus, one might conclude that it is not a good predictor to use for any model. At a less extreme, we can see that Adult Mortality and Alcohol have little variation, making them prime predictors for models. This observation proved to be true as we trained the models.

Regression Models

Before we explain the different models used, it is beneficial to first examine the correlation matrix for Life Expectancy.

```
> cor(datat[-1])[1,] # Negative is to exclude the status
       LifeExpectancy        Adult.Mortality          infant.deaths
            1.0000000             -0.7471698             -0.3422640
              Alcohol percentage.expenditure                    BMI
            0.5043812              0.3638928              0.4681632
   under.five.deaths      Total.expenditure               HIV.AIDS
           -0.3582252              0.3323861             -0.6337951
 thinness..1.19.years     thinness.5.9.years             dummy_Stat
           -0.5374488             -0.5718202             -0.5166101
```

At a first glance, it appears that the best predictors are Adult Mortality, HIV Aids, and thinness 5 to 9 years. We will explore these correlations more in depth as we progress through the models.

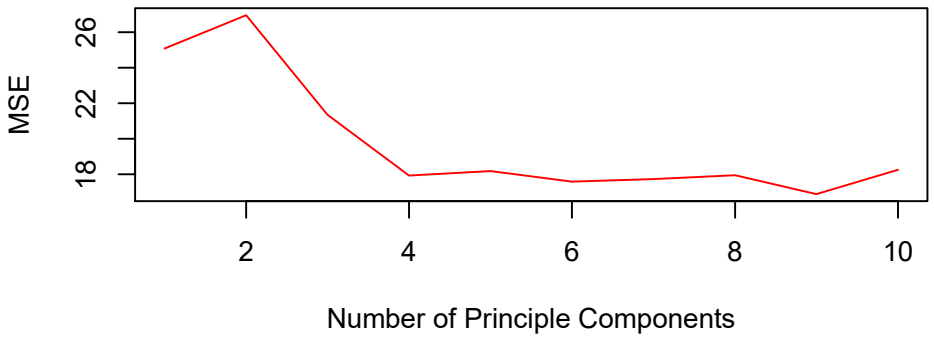
a Full Multiple Linear Regression

The full linear regression model did not perform well in training, but did in testing. As can be seen on the table for regression models.

b Multiple Linear Regression: PC predictors

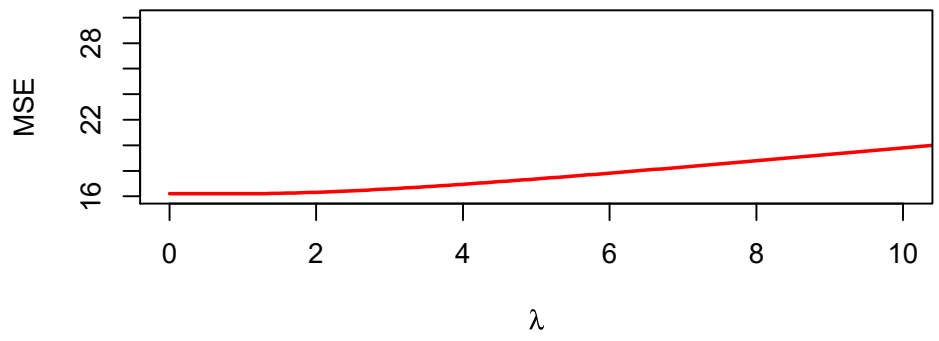
The multiple linear regression with PC predictors, had an inverse relationship with its full MLR counterpart. The training MSE was considerably lower than the full MLR training MSE, but the testing MSE was higher.

Five principle components were used to fit the training data. We can observe in figure 1 that five principle components yields the lowest MSE. The overuse of principle components most likely led to overfitting. This explains the lower testing MSE, as compared to the full MLR.



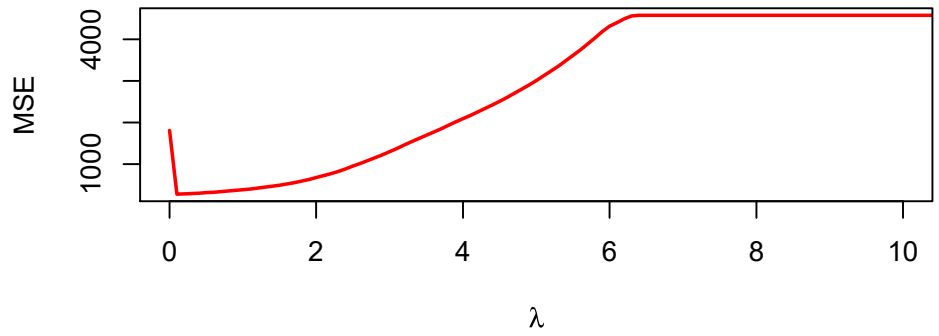
c Ridge Regression

For ridge regression, the model performed fairly well. Observing the graph below, we can see that as λ steadily increases from 0, the MSE does as well. The best λ for this model was $\lambda = 1$.



d Lasso Regression

Lasso regression was very similar to ridge. This is most likely due to both regression models having low λ values. For this data set, the lowest MSE was found when $\lambda = 0.1$.

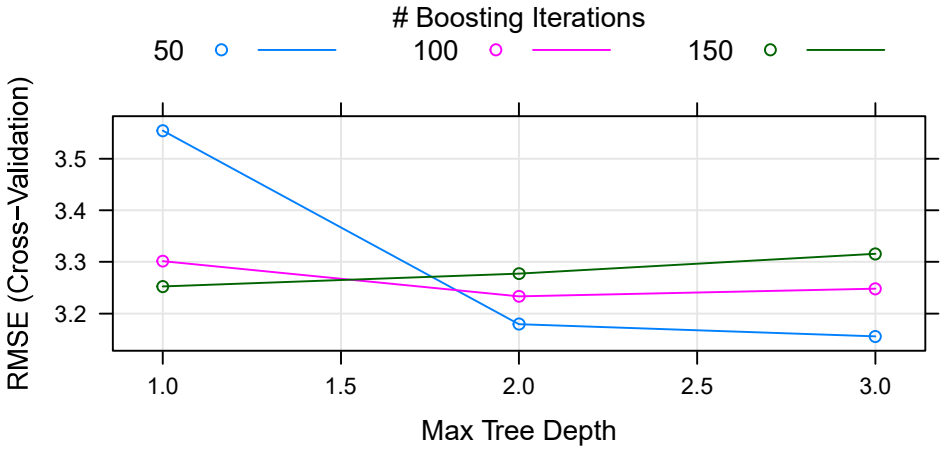


e Gradient Descent : Regression

The loss function for regression gradient descent was the linear regression cost function. Once evaluated, the model performed relatively poorly compared to the other models. This is due to the high variation in MSE caused by small changes to the tuning parameters. Such as the learning rate or number of iterations.

f Bonus Model: Boosted Tree

The boosted tree model performed the best out of all regression models. This is due to the lower number of observations in the testing data set. Tree based regression models fair well when observations are low relative to predictors. The model was trained for all possible predictors. For this boosted tree model, a maximum interaction depth of 3, boosting iterations of 50, and shrinking parameter of 0.1 was used on the final model.



g Results:

Model	Training MSE	Testing MSE
Full MLR	45.56	17.31
MLR (PC Predictors)	15.57	25.46
MLR Ridge	16.21	23.93
MLR Lasso	16.56	21.86
Gradient Descent	33.74	24.60
Boosted Tree	9.87	5.31

Classification

As we did with the regression models, it is first beneficial to observe the correlation matrix.

```
> cor(datat[-1])[12,]
       LifeExpectancy        Adult.Mortality          infant.deaths
            -0.5166101             0.3178033             0.1487456
              Alcohol percentage.expenditure                   BMI
            -0.4096808            -0.3441035            -0.1830496
     under.five.deaths      Total.expenditure               HIV.AIDS
             0.1434442            -0.3629035             0.2061912
  thinness..1.19.years     thinness.5.9.years            dummy_Stat
             0.3855883             0.4009390             1.0000000
```

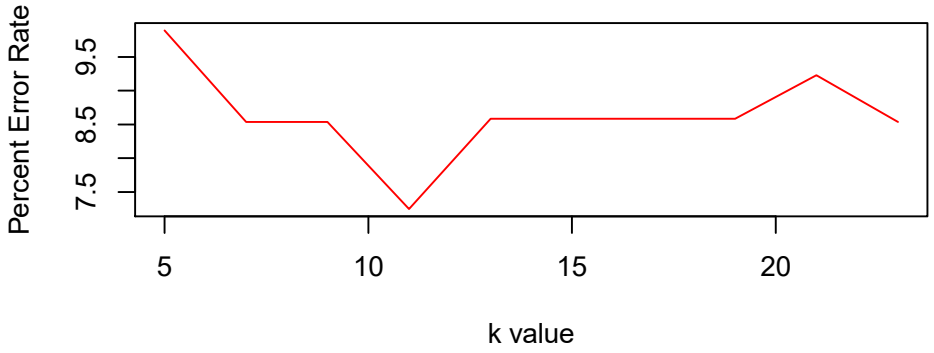
From the correlation matrix, we can see that the best predictors for status are life expectancy and alcohol. As we did with the regression model, we will observe these correlations in more detail.

h Logistic Regression

Logistic regression performed at an average range for this data set. The best predictors for this specific model were alcohol, thinness 5-9 years, adult mortality, and percentage expenditure.

i K Nearest Neighbors

k nearest neighbors has low training error rate, but high testing error rate. This is not uncommon for small data sets. For the final model, a k value of $k = 11$ produced the best results. Observing the figure below, one can see the effect of k on the percent error rate.



j Linear Discriminant Analysis

LDA proved rather average during training and testing. There were lack of extremes on either end. For the LDA model, it was found that the best predictors were thinness 5-9 years, alcohol, total expenditure, life expectancy, and adult mortality.

```
> ctrl <- trainControl(method="cv", summaryFunction=twoClassSummary, classProbs=T,
+                      savePredictions = T)
> rfFit <- train(Class ~., data=Sonar, method="rf", preProc=c("center", "scale"),
+                trControl=ctrl)
> library(pROC)
> selected_indecies = rfFit$pred$mtry == 2
> install.packages("plotROC")
> library(plotROC)
> library(ggplot2)
> ggplot(rfFit$pred[selected_indecies,], aes(m=M, d=factor(obs, levels=c("R", "M")))) + geom_roc(hju
> rfFit$preds
> roc.plot(dummy_Stat, rfFit$pred$Developing)
> install.packages("mlbench")
> library(mlbench)
> data(Sonar)
> names(Sonar)
```

k Quadratic Discriminant Analysis

QDA performed almost identically to LDA, with a slight variation in training error rate. It reference to their predictors, the only change from LDA was the swap of total expenditure for percentage expenditure.

l Naive Bayes

Naive Bayes did not perform as well in training as expected. When evaluated on the test data set, it had a error rate identical to QDA and LDA.

m Gradient Descent: Classification

For gradient descent classification, a logistic loss function was used. Similar to the regression gradient descent, the error rate was highly dependent on the tuning parameters, such as iterations and learning rate. Gradient descent had a high error rate for both training and testing. It was found that the best predictors for this loss function of gradient descent were life expectancy and adult mortality.

n Bonus Model: AdaBoost

This model performed well and consistently in both training and testing data. The best predictors for this model were the same as LDA.

o Results:

Model	Training PER	Testing PER
Logistic Regression	11.27%	10.71%
LDA	11.29%	7.14%
QDA	11.25%	7.14%
kNN	7.25%	21.43%
Naive Bayes	11.25%	7.14%
Gradient Descent	35.33%	17.86
AdaBoost	7.23%	7.14%

Appendix: Code for Regression and classification models

- Create the data frame

```
> datas = read.csv('datatest.csv')
> dftest = data.frame(datas)
> rownames(dftest) = dftest$X
> dftest$X = NULL
```

- Boosted Tree

```
> set.seed(1)
> boostfit = train(LifeExpectancy~.-Status-dummy,data = df,
+                  method = "bstTree",trControl=trainControl(method = "cv",
+                                                            number  = 5))
> preds = predict(boostfit$finalModel,dftest)
> testmse = mean((preds-dftest$LifeExpectancy)^2)
> trainmse = getTrainPerf(boostfit)$TrainRMSE^2
```

- LDA

```
> ldafit = train(Status~thinness.5.9.years+Alcohol+Total.expenditure+
+                  LifeExpectancy+Adult.Mortality,data = df,
+                method = "lda",
+                trControl=trainControl(method = "cv", number  = 5))
> preds = predict(ldafit$finalModel,cbind(dftest$thinness.5.9.years,
+                                         dftest$Alcohol,dftest$Total.expenditure,
+                                         dftest$LifeExpectancy,dftest$Adult.Mortality))
> tester = 1-(mean(preds$class==dftest$Status))
> trainer=1-getTrainPerf(ldafit)$TrainAccuracy
```

- QDA

```
> qdafit = train(Status~thinness.5.9.years+Alcohol+percentage.expenditure+
+                  Adult.Mortality+LifeExpectancy,data = df,method = "qda",
+                trControl=trainControl(method = "cv", number  = 5))
> qdapreds = predict(qdafit$finalModel,cbind(dftest$thinness.5.9.years,
+                                            dftest$Alcohol,dftest$percentage.expenditure,
+                                            dftest$Adult.Mortality,dftest$LifeExpectancy))
> tester = 1-(mean(qdapreds$class==dftest$Status))
> trainer=1-getTrainPerf(qdafit)$TrainAccuracy
```

- AdaBoost

```
> adafit = train(Status~thinness.5.9.years+Alcohol+Total.expenditure+Adult.Mortality+
+                  LifeExpectancy,data = df,method = "adaboost",
+                trControl=trainControl(method = "cv", number  = 5))
> adapreds = predict(adafit$finalModel,dftest)
> tester = 1-mean(adapreds$class==dftest$Status)
> trainer=1-getTrainPerf(adafit)$TrainAccuracy
```

- Naive Bayes

```
> nbfit = train(Status~thinness.5.9.years+Alcohol+Total.expenditure+Adult.Mortality+
+                 LifeExpectancy,data = df,method = "naive_bayes",
+               trControl=trainControl(method = "cv", number  = 5))
> nbpreds = predict(nbfit,newdata =dftest,type='raw')
> tester = 1-mean(nbpreds==dftest$Status)
> trainer=1-getTrainPerf(nbfit)$TrainAccuracy
```

- K nearest neighbor

```
> knnfit = train(Status~thinness.5.9.years+Alcohol+Total.expenditure+
+                  Adult.Mortality+LifeExpectancy,data = df,
+                method = "knn",trControl=trainControl(method = "cv", number  = 5),
+                tuneLength = 10)
> knnpreds = predict(knnfit,dftest)
> tester = 1-mean(knnpreds == dftest$Status)
> trainer=1-getTrainPerf(knnfit)$TrainAccuracy
```

- MLR with PC predictors

```
> pcfit = train(LifeExpectancy~., data=df[1:12], method = "lm", preProcess = "pca",
+                      trControl = trainControl(method = "cv",
+                                               number=5,
+                                               preProcOptions = list(pcaComp = 9)))
> pcpreds = predict(pcfit,dftest)
> testmse = mean((pcpreds-dftest$LifeExpectancy)^2)
> trainmse = getTrainPerf(pcfit)$TrainRMSE^2
```

- Full MLR

```
> fulllmfit = train(LifeExpectancy~.,data = df[1:12],method = "lm",
+                   trControl=trainControl(method = "cv", number  = 5))
> fullpreds = predict(fulllmfit,dftest)
> testmse = mean((fullpreds-dftest$LifeExpectancy)^2)
> trainmse = getTrainPerf(fulllmfit)$TrainRMSE^2
```

- Ridge Regression

```
> ridgefit = train(LifeExpectancy~.,data = df[1:12],method = "glmnet",
+                  trControl=trainControl(method = "cv", number  = 5),
+                  tuneGrid = expand.grid(alpha=0,lambda=1))
> ridgepreds = predict(ridgefit,dftest)
> testmse = mean((ridgepreds-dftest$LifeExpectancy)^2)
> trainmse = getTrainPerf(ridgefit)$TrainRMSE^2
```

- Lasso Regression

```
> lassofit = train(LifeExpectancy~.,data = df[1:12],method = "glmnet",
+                  trControl=trainControl(method = "cv", number  = 5),
+                  tuneGrid = expand.grid(alpha=1,lambda=0.1))
> lassopreds = predict(lassofit,dftest)
> testmse = mean((lassopreds-dftest$LifeExpectancy)^2)
> trainmse = getTrainPerf(lassofit)$TrainRMSE^2
```

- Logistic Regression

```
> trn_ctrl_log = trainControl(method="cv", number=5)
> log_mod = train(Status~ Alcohol+thinness.5.9.years+Adult.Mortality
+                 +percentage.expenditure+LifeExpectancy,
+                 data=df,trControl=trn_ctrl_log, method="glm", family="binomial")
> logpreds = predict(log_mod,dftest)
> tester = 1-(mean(logpreds==dftest$Status))
> trainer = 1-getTrainPerf(log_mod)$TrainAccuracy
```

- Gradient Descent: Classification

```
> set.seed(13)
> sigmoid <- function(z) 1/(1 + exp(-1*z))
> my.logistic<-function(par, X,y, alpha, plot=FALSE)
+ {
```

```
+
+   n <- ncol(X)
+   m <- nrow(X)
+   ll<- rep(NA, m)
+   thetaall <- matrix(NA, n, m)
+   X<-cbind(1,X)
+
+   thetaall<-theta
+   for (i in 1:m)
+   {
+     dim(X)
+     length(theta)
+     hx <- sigmoid(X %*% theta) # matrix product
+     theta <- theta + alpha * (y - hx)[i] * X[i, ]
+     logl <- sum( y * log(hx) + (1 - y) * log(1 - hx) ) #direct multiplication
+
+     ll[i] <- logl
+
+     thetaall = cbind(thetaall, theta)
+   }
+
+
+
+   return(list(par=theta, loglik=logl))
+ }
> grad_desc_coeff = function(X, y ){
+   theta <-rep(0,(ncol(X)+1)) # intercept and 5 regerssors
+   delta <- 1
+   while (delta > 0.0001) {
+   ans <- my.logistic(theta, X,y, alpha=0.0005,plot=TRUE)
+   theta.new <- ans$par
+   delta <- max(abs(theta - theta.new))
+   theta <- theta.new
+   }
+   return(list(int = ans$par[1], b1 = ans$par[2], b2 = ans$par[3]))
+ }
> theta = rep(0,3)
>


> X <- cbind(LifeExpectancy,Adult.Mortality)
> X.orig <- X
> X <- scale(X.orig)
> #cf = grad_desc_coeff(X,y)
>
>
> pred_func = function(x1, x2, coef){
+   g = 1 + exp(coef$int + coef$b1*x1 + coef$b2*x2)
+   return(1/g)
+ }
> compar_func = function(x1, x2 , y_dat,coef=cf){
+   lst = rep(NA, length(y_dat))
```

```
+
+   for (j in 1:length(lst)) {
+     lst[j] = pred_func(x1[j], x2[j], coef)
+   }
+   prob_lst = rep(0, length(y_dat))
+   prob_lst[lst>0.5]=1
+   return(1 - mean(y_dat == prob_lst))
+ }
> cv_grad = function(x1, x2, y_dat){
+   X <- cbind(x1,x2)
+   X.orig <- X
+   X <- scale(X.orig)
+   folds = cut(seq(1, nrow(x)), breaks=5, labels=F)
+
+   for (i in 1:5) {
+     test = which(folds==i, arr.ind = T)
+     x_new = X[-test]
+     y_new = y_dat[-test]
+
+     print(grad_desc_coeff(x_new, y_new))
+   }
+ }


> cv_grad = function(x1, x2, y_dat){
+   X <- cbind(x1,x2)
+   X.orig <- X
+   X <- scale(X.orig)
+   folds = cut(seq(1, nrow(X)), breaks=5, labels=F)
+   int_lst = rep(NA, 5)
+   b1_lst = rep(NA, 5)
+   b2_lst = rep(NA,5)
+
+   er_lst = rep(0,5)
+   for (i in 1:5) {
+     test = which(folds==i, arr.ind = T)
+     x_new = X[-test,]
+     y_new = y_dat[-test]
+
+     cf = grad_desc_coeff(x_new, y_new)
+     er_lst[i]=compar_func(x1[test], x2[test], y_dat[test], coef=cf)
+
+     int_lst[i] = cf$int
+     b1_lst[i] = cf$b1
+     b2_lst[i] = cf$b2
+
+   }
+   return(list(er_mean = mean(er_lst), int_best = mean(int_lst),
+               b1_best = mean(b1_lst), b2_best = mean(b2_lst)))
+ }
> test_err_ad_coef = cv_grad(LifeExpectancy, Adult.Mortality, datat$dummy_Stat)
> print(1 - test_err_ad_coef$er_mean)
```

```
> best_mod_compar = function(x1, x2){
+   g = 1 + exp(test_err_ad_coef$int_best +
+                 test_err_ad_coef$b1_best * x1 +
+                 test_err_ad_coef$b2_best * x2)
+   return(1/g)
+ }
> compar_best_func = function(x1, x2 , y_dat){
+   lst = rep(NA, length(y_dat))
+
+   for (j in 1:length(lst)) {
+     lst[j] = best_mod_compar(x1[j], x2[j])
+   }
+   prob_lst = rep(0, length(y_dat))
+   prob_lst[lst>0.5]=1
+   return(1 - mean(y_dat == prob_lst))
+ }
> 1 - compar_best_func(datas$LifeExpectancy, datas$Adult.Mortality, datas$dummy_Stat)
>
```

- Gradient Descent Regression

```
> cost <- function(X, y, theta) {
+   sum( (X %*% theta - y)^2 ) / (2*length(y))
+ }
> fit = lm(LifeExpectancy~Alcohol + Adult.Mortality,data=datat)
> fit$coefficients
> grad_reg_coef = function(x1, x2, y,  alpha=0.01, num_iters=1000){
+   X <- cbind(x1,x2)
+   X.orig <- X
+   X <- scale(X.orig)
+
+   X <- cbind(1,X)
+
+   cost_history <- double(num_iters)
+   theta_history <- list(num_iters)
+   theta <- matrix(c(0,0,0), nrow=3)
+
+   for (i in 1:num_iters) {
+   error <- (X %*% theta - y)
+   delta <- t(X) %*% error / length(y)
+   theta <- theta - alpha * delta
+   cost_history[i] <- cost(X, y, theta)
+   theta_history[[i]] <- theta
+   }
+   return(theta)
+ }
> grad_reg_coef(Alcohol, Adult.Mortality, LifeExpectancy)
> reg_mod = function(x1, x2, cf){
+   return((cf[1]+5.04) + x1*(cf[2] + -2.7) + x2*(cf[3]+ 5.42))
+ }
> test_reg_func = function(x1, x2 , y_dat,cf){
```

```
+   lst = rep(NA, length(y_dat))
+
+   for (j in 1:length(lst)) {
+     lst[j] = reg_mod(x1[j], x2[j], cf)
+   }
+   return(mean((lst-y_dat)^2))
+ }
> test_reg_func( Alcohol, Adult.Mortality, LifeExpectancy,
+                grad_reg_coef( Alcohol, Adult.Mortality, LifeExpectancy))
> test_reg_func(datas$Alcohol, datas$Adult.Mortality,
+               datas$LifeExpectancy, grad_reg_coef( Alcohol, Adult.Mortality, LifeExpectancy))
```