CSE-6363 Assignment 1 Swetha Vijaya Raghavan 1001551229

This Assignment is implemented using the R Language to build a Linear Regression model.

```
# Setting up the environment and importing the Iris data
setwd('/Users/swethavijayaraghavan/Desktop/1001551229_Swetha_VijayaRaghavan')
library('MASS')
data<-read.csv('iris.txt',header=FALSE)
irisData<-data
irisData$V5<-NULL #Removed the classlabel
# Assinging the data in to respectivce A, Y matrices
A<-data.matrix(irisData, rownames.force = NA)
#Transpose of A
transposeA<-t(A)
Y<-matrix(data$V5)
# Categorizing the class labels into 3 Integer values
Y<-sapply(data$V5,switch,'Iris-setosa'=1,'Iris-versicolor'=2,'Iris-virginica'=3)
#1.Iris-setosa
#2.Iris-virginica
#3.Iris-versicolor
```

Training the Model using Linear Regression using the formula, Beta = $(AA^T)^{-1}A^TY$

```
# Applying Linear Regression using the formula
linearRegression <- function(A, Y){
   transposeA<-t(A)
   beta<-ginv(transposeA%*%A)%*%transposeA%*%Y
   return(beta)
  }
```

Now the trained model is used for classification. The predicted values are rounded up to their Integer values.

```
# Applying Classification
classification <- function(testing, beta, classLength){
   Yout <- round(testing%*%beta)
   Yout[ Yout > classLength ] <- classLength
   Yout[Yout < 1] <- 1
   return(Yout)
  }
```

Applying K-Fold Cross Validation to find error rate. The dataset is divided in to K Folds(In my code, out of K=3/5/10/50,. Based on this, divided into the train and test and train the model.

```
KFoldCrossValidation <- function(A,Y,K){
   randomindex<-sample(length(Y))
   error <- 0
   size<-length(Y)/K
   class <- unique(Y)
   for(i in 1:k){
     testing <- randomindex[ (size*(i-1)+1): (size*i) ]; # Divided the data into Testing Data
     training <- setdiff( 1:length(Y),  testing); # Rest as Training Set
     beta <- linearRegression(A[training, ], Y[training])
     Ypred <- classification(A[testing, ], beta, length(class))
     error <- error + sumOfSquaredError(Ypred, Y[testing])
  }
   error <- error/K
   return(error)
}
```

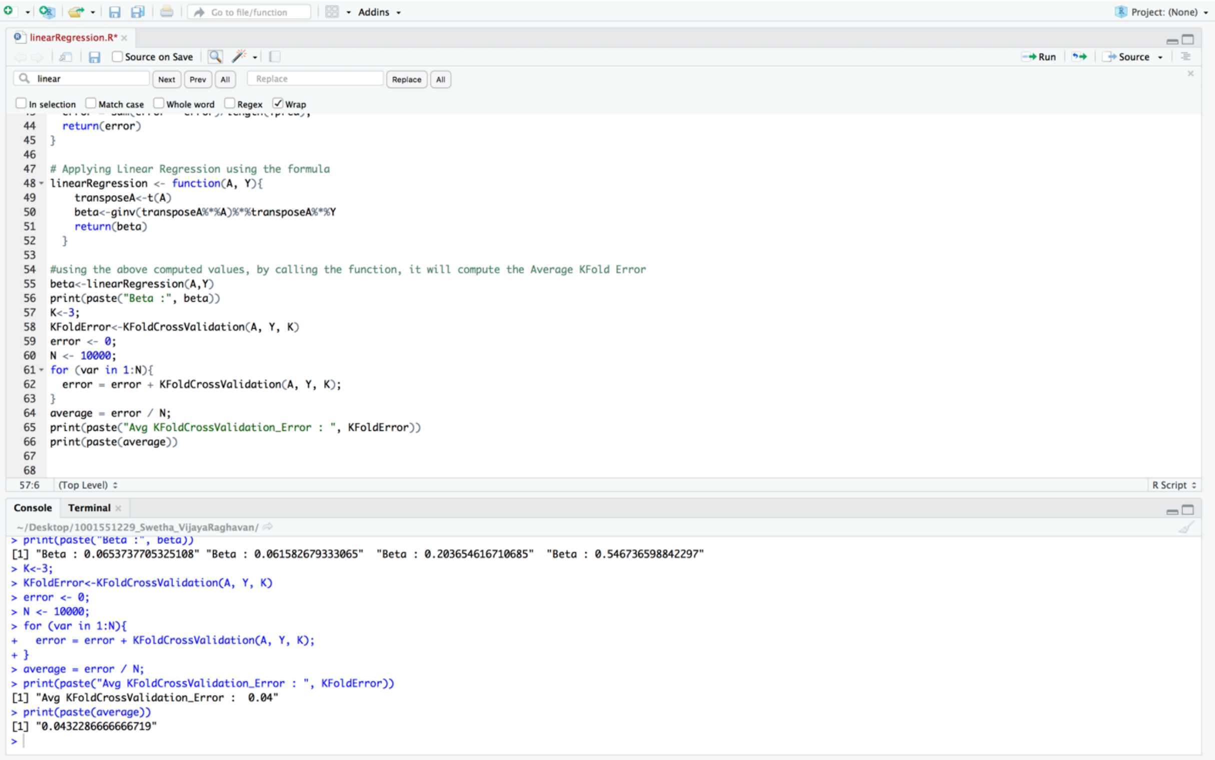
Sum of squared error – This finds the error rate by taking the difference of the actual Y value from the predicted Y values.

```
# Sum of squared error using the formula
sumOfSquaredError = function(Yact,Ypred){
  error = Ypred - Yact;
  error = sum(error * error)/length(Ypred);
  return(error)
}
```

```
#using the above computed values, by calling the function, it will compute the Average KFold Error
beta<-linearRegression(A,Y)
print(paste("Beta Values:", beta))
K<-10;
KFoldError<-KFoldCrossValidation(A, Y, K)
error <- 0;
N <- 10000;
for (var in 1:N){
  error = error + KFoldCrossValidation(A, Y, K);
}
average = error / N;
print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
print(paste(average))
```

Linear regression on A and Y matrix and found the average K-Fold-Error rate. I have tried using different K values, such as K=3,5,10 as K=10 takes less time to perform lower number of folds and the error rate is also less which is as shown below(screenshots):

K=3



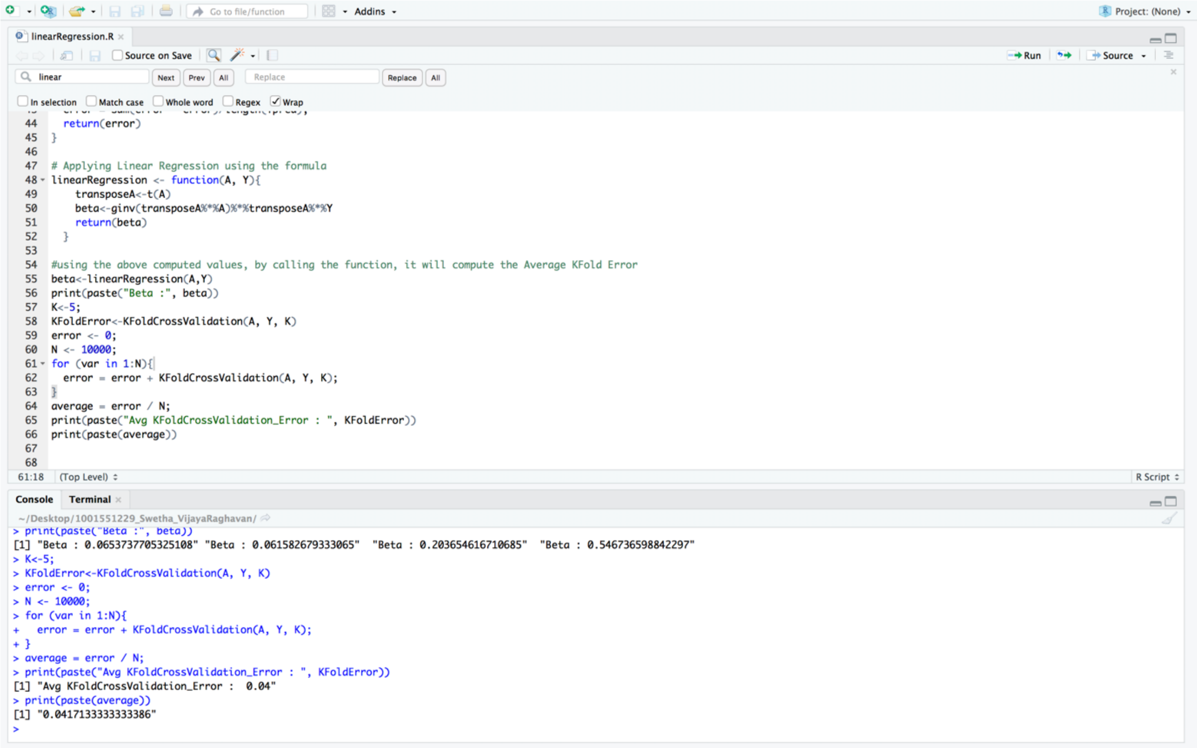
```
  return(error)
}

# Applying Linear Regression using the formula
linearRegression <- function(A, Y){
    transposeA<-t(A)
    beta<-ginv(transposeA%*%A)%*%transposeA%*%Y
    return(beta)
  }

#using the above computed values, by calling the function, it will compute the Average KFold Error
beta<-linearRegression(A,Y)
print(paste("Beta :", beta))
K<-3;
KFoldError<-KFoldCrossValidation(A, Y, K)
error <- 0;
N <- 10000;
for (var in 1:N){
  error = error + KFoldCrossValidation(A, Y, K);
}
average = error / N;
print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
print(paste(average))
```

```
> print(paste("Beta :", beta))
[1] "Beta : 0.0653737705325108" "Beta : 0.061582679333065"  "Beta : 0.203654616710685"  "Beta : 0.546736598842297"
> K<-3;
> KFoldError<-KFoldCrossValidation(A, Y, K)
> error <- 0;
> N <- 10000;
> for (var in 1:N){
+   error = error + KFoldCrossValidation(A, Y, K);
+ }
> average = error / N;
> print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
[1] "Avg KFoldCrossValidation_Error :  0.04"
> print(paste(average))
[1] "0.0432286666666719"
> 
```

K=5



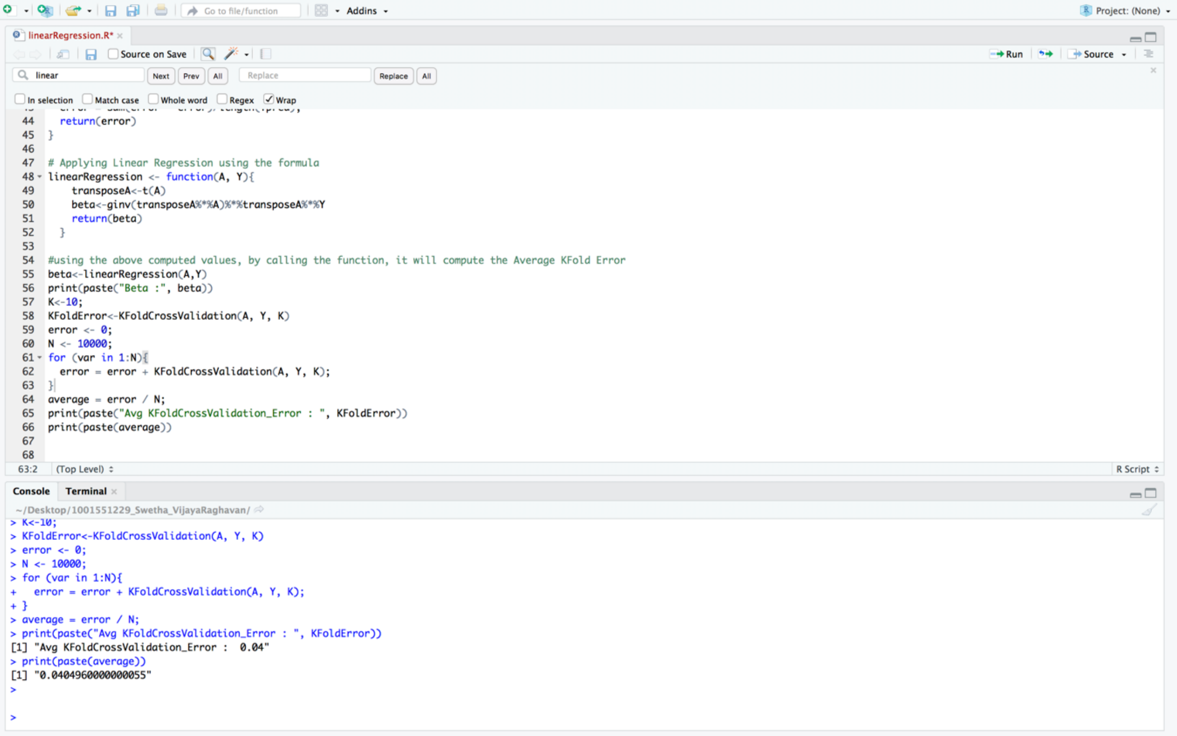
```
  return(error)
}

# Applying Linear Regression using the formula
linearRegression <- function(A, Y){
    transposeA<-t(A)
    beta<-ginv(transposeA%*%A)%*%transposeA%*%Y
    return(beta)
  }

#using the above computed values, by calling the function, it will compute the Average KFold Error
beta<-linearRegression(A,Y)
print(paste("Beta :", beta))
K<-5;
KFoldError<-KFoldCrossValidation(A, Y, K)
error <- 0;
N <- 10000;
for (var in 1:N){
  error = error + KFoldCrossValidation(A, Y, K);
}
average = error / N;
print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
print(paste(average))
```

```
> print(paste("Beta :", beta))
[1] "Beta : 0.0653737705325108" "Beta : 0.061582679333065"  "Beta : 0.203654616710685"  "Beta : 0.546736598842297"
> K<-5;
> KFoldError<-KFoldCrossValidation(A, Y, K)
> error <- 0;
> N <- 10000;
> for (var in 1:N){
+   error = error + KFoldCrossValidation(A, Y, K);
+ }
> average = error / N;
> print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
[1] "Avg KFoldCrossValidation_Error :  0.04"
> print(paste(average))
[1] "0.0417133333333386"
>
```

K=10



```
  return(error)
}

# Applying Linear Regression using the formula
linearRegression <- function(A, Y){
    transposeA<-t(A)
    beta<-ginv(transposeA%*%A)%*%transposeA%*%Y
    return(beta)
  }

#using the above computed values, by calling the function, it will compute the Average KFold Error
beta<-linearRegression(A,Y)
print(paste("Beta :", beta))
K<-10;
KFoldError<-KFoldCrossValidation(A, Y, K)
error <- 0;
N <- 10000;
for (var in 1:N){
  error = error + KFoldCrossValidation(A, Y, K);
}
average = error / N;
print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
print(paste(average))
```

```
> K<-10;
> KFoldError<-KFoldCrossValidation(A, Y, K)
> error <- 0;
> N <- 10000;
> for (var in 1:N){
+   error = error + KFoldCrossValidation(A, Y, K);
+ }
> average = error / N;
> print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
[1] "Avg KFoldCrossValidation_Error :  0.04"
> print(paste(average))
[1] "0.0404960000000055"
>

>
```

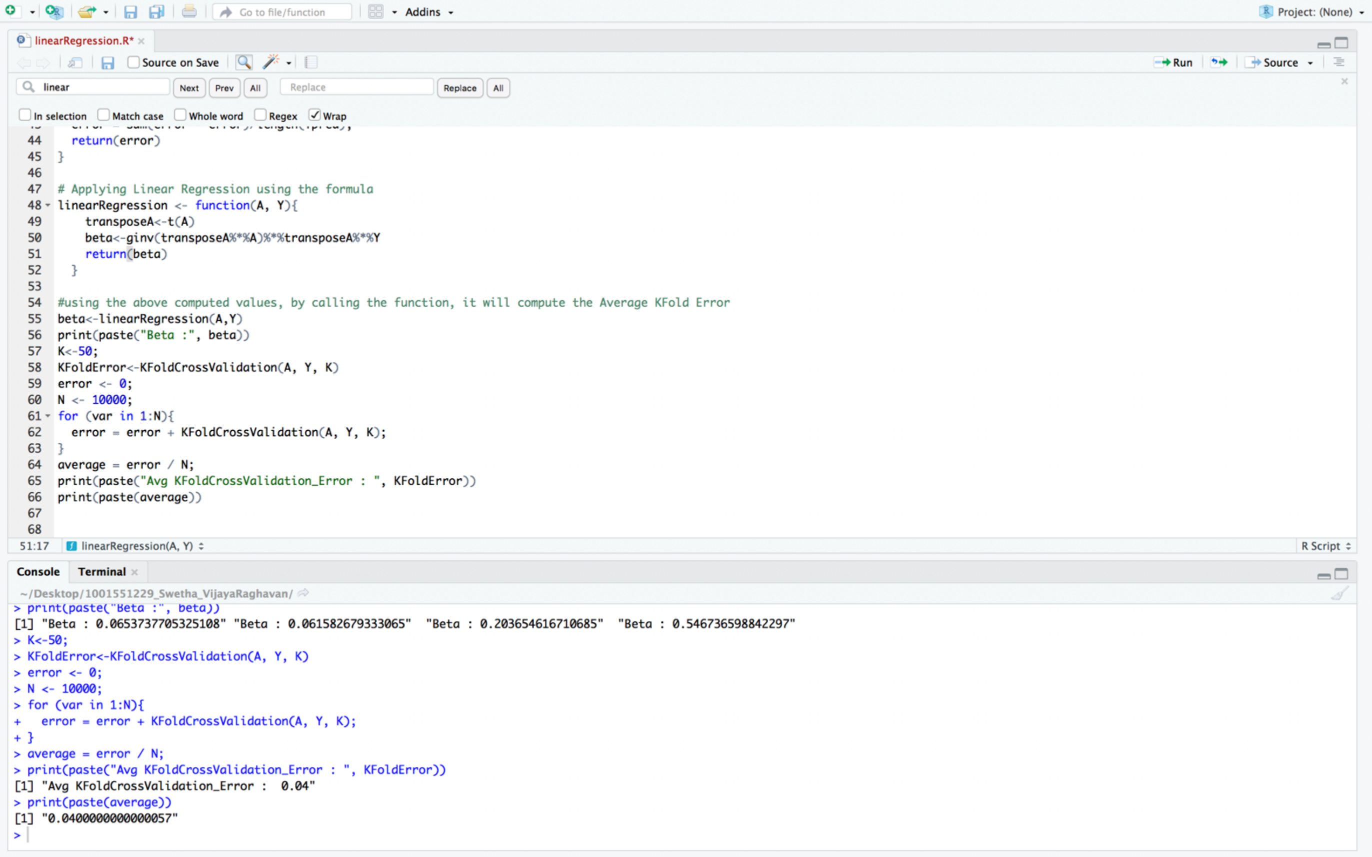
Hence if the error rate is 4% the accuracy can be said to be 96% for 10-Fold cross validation.

```
error <- 0;
N <- 10000;
for (i in 1:N){
  error = error +
KFoldCrossValidation(
A, Y, k);
}
avg = error / N;
print(paste("Average
Kfold Error : ",
KVError))
print(paste(avg))
```

Above code is used in order to find the ideal value by running K fold for N times which is 1000 in this case. By running this KFold for 10000 times on with k=10 we find the error rate which is efficient as well as ideal to calculate the error rate.

If we take k=50 the error is reduced little bit (0.0400000000000057)but takes lot of time to perform 30 fold cross validation and is not very efficient.

K = 50



```r
  return(error)
}

# Applying Linear Regression using the formula
linearRegression <- function(A, Y){
    transposeA<-t(A)
    beta<-ginv(transposeA%*%A)%*%transposeA%*%Y
    return(beta)
  }

#using the above computed values, by calling the function, it will compute the Average KFold Error
beta<-linearRegression(A,Y)
print(paste("Beta :", beta))
K<-50;
KFoldError<-KFoldCrossValidation(A, Y, K)
error <- 0;
N <- 10000;
for (var in 1:N){
  error = error + KFoldCrossValidation(A, Y, K);
}
average = error / N;
print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
print(paste(average))
```

```
> print(paste("Beta :", beta))
[1] "Beta : 0.0653737705325108" "Beta : 0.061582679333065"  "Beta : 0.203654616710685"  "Beta : 0.546736598842297"
> K<-50;
> KFoldError<-KFoldCrossValidation(A, Y, K)
> error <- 0;
> N <- 10000;
> for (var in 1:N){
+   error = error + KFoldCrossValidation(A, Y, K);
+ }
> average = error / N;
> print(paste("Avg KFoldCrossValidation_Error : ", KFoldError))
[1] "Avg KFoldCrossValidation_Error :  0.04"
> print(paste(average))
[1] "0.0400000000000057"
> 
```

References: (Wiki, Github, etc)

https://datascienceplus.com/linear-regression-from-scratch-in-r/
https://tutorials.iq.harvard.edu/R/Rstatistics/Rstatistics.html
https://www.tutorialspoint.com/r/r_linear_regression.htm
https://datascienceplus.com/how-to-apply-linear-regression-in-r/
https://github.com/AntoineGuillot2/Linear-Regression-R/blob/master/script.R
https://www.geeksforgeeks.org/simple-linear-regression-using-r/