# Predicting Medical Insurance Cost using Multiple Linear Regression, Stepwise Regression, Ridge regression, Lasso Regression and Multilayer Perceptron models.

**Faisal Raiyan Huda**

**Abstract**
Human beings have been long living in a world full of risks. In the modern days, human beings are now protected by insurance companies. Insurance companies need to have high predictions to ensure their clients are protected and keep themselves sustained. In this study, using medical cost dataset from Kaggle, multiple linear regression, stepwise regression, ridge regression, lasso regression and multilayer perceptron models were developed and evaluated using their R2 and RMSE values. Though multilayer perceptron model outperforms the rest, the RMSE values found were high.

## 1. Introduction

It is a basic human necessity to have access to health care. Humans have been living in a world which poses a lot of risk and hazardous conditions. But one cannot stay at home and avoid these risks, but one can be insured against these risks and hazards. This is where insurance organizations come into play where these organizations protect individuals by using financial capital to guard them [1][2].

That is why, it is of utmost importance for insurance companies to be very precise with their predictions to ensure that they have sufficient capital to cover the clients for the unforeseeable future. Many factors are to be considered when it comes to predicting the insurance cost. Previous studies have tried to predicting insurance cost using machine learning. Approaches used were Linear regression, decision tree regression, gradient boosting, support vector machine and ridge regressor [1][2][3]. Previous study has found stochastic gradient boosting (SGB) is the best for predicting insurance cost. This study will directly compare multiple linear regression with one the studies that used the same dataset acquired from Kaggle [2][4] and additionally evaluate the prediction accuracy of a Multilayer perceptron (MLP) model along with stepwise, lasso and ridge using the dataset.

The aim of this study is to build multiple linear regression (MLR), stepwise regression, ridge regression, lasso regression and MLP models to predict the insurance cost. To achieve this aim, first the dataset is obtained from Kaggle, then the dataset is prepossessed, after which the dataset is standardized and split into 80/20 train and test dataset. After the machine learning models were developed their R2 and RMSE values were compared.

## 2. Data

The dataset is by Miri Choi, uploaded on Kaggle 5 years ago[1]. This dataset is related to the medical insurance cost of individuals in the different residential areas in the United States. The dataset consists of 7 columns and 1338 rows. Where is ‘age’ column represents the age of the individual, ‘sex’ is the gender (male or female), ‘bmi’ column is the Body Mass Index, ‘children’ represents the number of children of the individual, the ‘smoker’ column looks if the individual is a smoker(yes or no), ‘region’ represents the regional area in the US and lastly the ‘charges’ column is the medical fees billed by the health insurance. As observed, this dataset consists of continuous and categorical variables. The response variable will be the ‘charges’ and the rest of the columns will be the predictor variables.

## 3. Method

All the codes and files associated with this study can be found in the github.

### 3.1. Pre-processing of the Dataset

First, the dataset was checked for normality, missing values, and outliers. Fig. 1 represents the histograms for the continuous variables of this dataset.

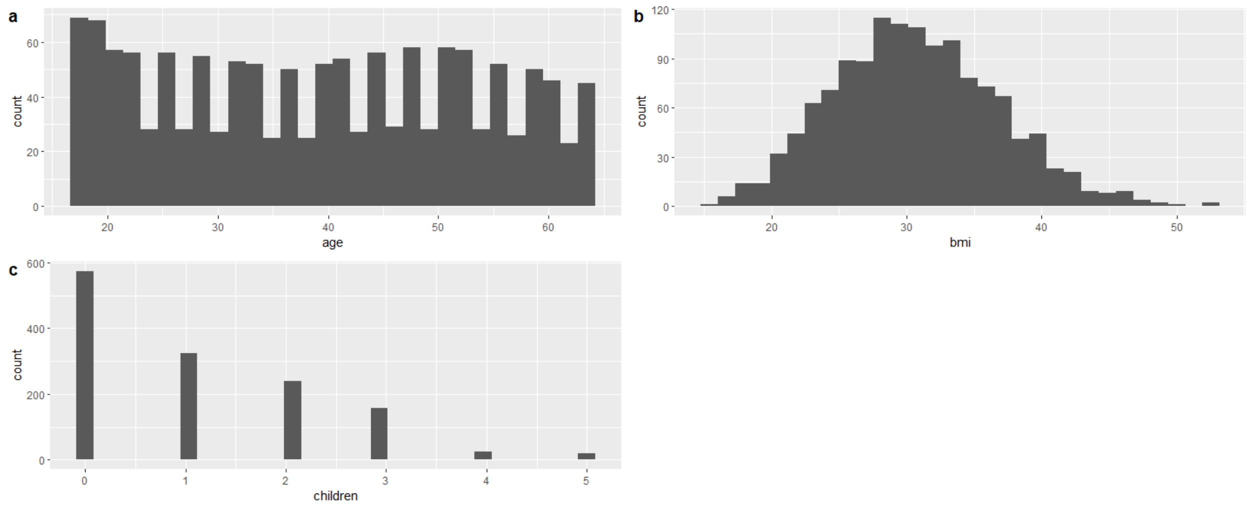


Fig. 1: Histogram of the continuous variables

It was observed there were some outliers for bmi, thus a robust threshold approach was utilized. This approach involves setting a threshold that defines the range of acceptable values and then replacing any values that fall outside that range with the maximum or minimum value (Code Block A in Appendix).

### 3.2. Standardization of variables and splitting of the Dataset

The continuous variables of the dataset were standardization first using the following equation –

$$y_{standard} = \frac{y_i - \bar{y}}{\sigma_y}$$

Where, $y_i$ represents the variable, $\bar{y}$ is the mean of the variable and $\sigma_y$ is the standard deviation of the variable. After standardization, the dataset was split into an 80/20 ratio of training set and test set.

## 3.3. Evaluation Metrics

### *3.3.1. Coefficient of Determination (R2)*

The R2 gives the fraction of the variability of y explained by the least squares linear regression of on x. It is an overall measure of how well the regression line approximates the actual data. The R2 is always between 0 and 1 and is converted to percentage [5]. It can be represented using the following equation –

$$R2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

Where, $\hat{y}_i$ is the predicted value of the variable.

### *3.3.2. Root Mean Squared Error (RMSE)*

The RMSE is an error metric that calculates the square root of mean of the squared differences between actual and predicted Y values. The formula below calculates RMSE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}$$

Where, N is the number of observations. The other variables mean the same as explained previously.

## 3.4. Multiple Linear Regression (MLR)

Mathematically, for the multiple linear regression model, the relationship can be written as

$$Y = \beta_o + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \in$$

Where response variable Y is the charges by medical insurance company in USD ($). And $X_p$ represents the *p*th predictor and $\beta_p$ is the *p*th coefficient. We interpret β as the average change on Y of a one-unit increase in X while keeping the other predictors constant.

Code Block C in the Appendix illustrates the application of MLR for this study, where an MLR model was fit using the training dataset, and the fitted model's prediction on the test dataset was evaluated using R2 and RMSE.

## 3.5. Stepwise Regression

Stepwise regression is a method used in predictive modeling to identify the most important predictors that contribute to the accuracy of the model. It involves an iterative process of adding and removing predictors until there is no significant reason to add or remove a variable [6]. In this study the stepwise regression was conducted through backward stepwise selection to find the most significant predictors for predicting insurance charge. Code Block D in the Appendix illustrates the application in this study. The model was evaluated using R2 and RMSE values.

## 3.6. Ridge and Lasso Regression

$$\sum_{i=1}^{n}(y_i - \beta_o - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \qquad (1)$$

$$\sum_{i=1}^{n}(y_i - \beta_o - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| \qquad (2)$$

(1) is the ridge regression equation while (2) represents the lasso regression equation. The two equations are slightly similar to MLR equation where the β represents the coefficient. Just like MLR, ridge also tries to find coefficient estimates that fit the data well by reducing the residual sum of squares (RSS). In equation (1), the second term is the shrinkage penalty $\lambda\sum_{j=1}^{p}\beta_j^2$ , where it small if the coefficients are close to zero. The tuning parameter $\lambda$ control the relative impact of the first term and second term of the equation on the regression coefficient estimates. The lasso equation (2) is also similar to ridge except the shrinkage penalty is $\lambda\sum_{j=1}^{p}|\beta_j|$ which also the coefficient estimates shrink to zero[7]. The two models were evaluated using R2 and RMSE. The application of ridge and lasso regression is in Code Block E and F of Appendix.

## 3.7. Multilayer Perceptron (MLP)

The dataset was standardized before splitting, thus, it was ensured no bias in the model. The MLP model consisted of 8 inputs. The inputs were fed into the neural network, which consists 1 hidden layer containing 6 neurons with bias input as shown in Fig. 2.

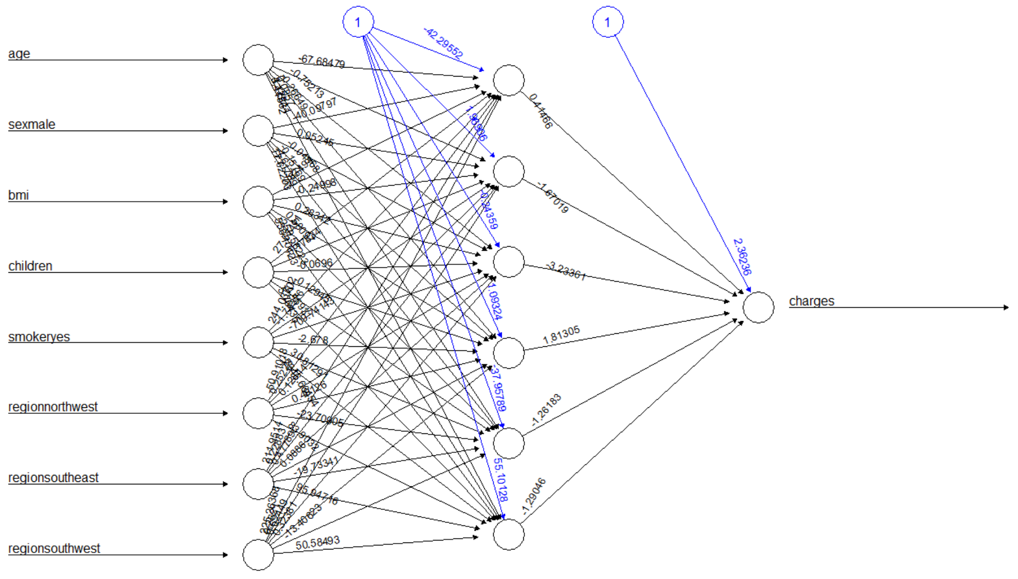


*Fig. 2: Neural network Architecture of the MLP model.*

The activation function used in the hidden layers was the sigmoid function where the output is between zero and one. The activation function's equation can be written as shown below -

$$g(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$$

As it is a regression problem, the output layer uses the following linear activation function -

$$g(z) = z$$

MLP has two main methods named feed-forward and back-propagation for optimizing the model and finding the optimal weights in the model. The number of neurons, value for threshold and stepmax were found through trail-and-error bias until the model was able to converge consistently. The model was evaluated using RMSE only. The application of MLP is in CODE BLOCK H in the Appendix.

## 4. Results and Discussion

Table 1: Summary of R2 and RMSE values for the models

| **Model** | **R2** | **RMSE** |
|---|---|---|
| MLR | 0.75 | 6372 |
| Stepwise Regression | 0.75 | 6371 |
| Ridge Regression | 0.75 | 6465 |
| Lasso Regression | 0.75 | 6465 |
| MLP | - | 5092 |

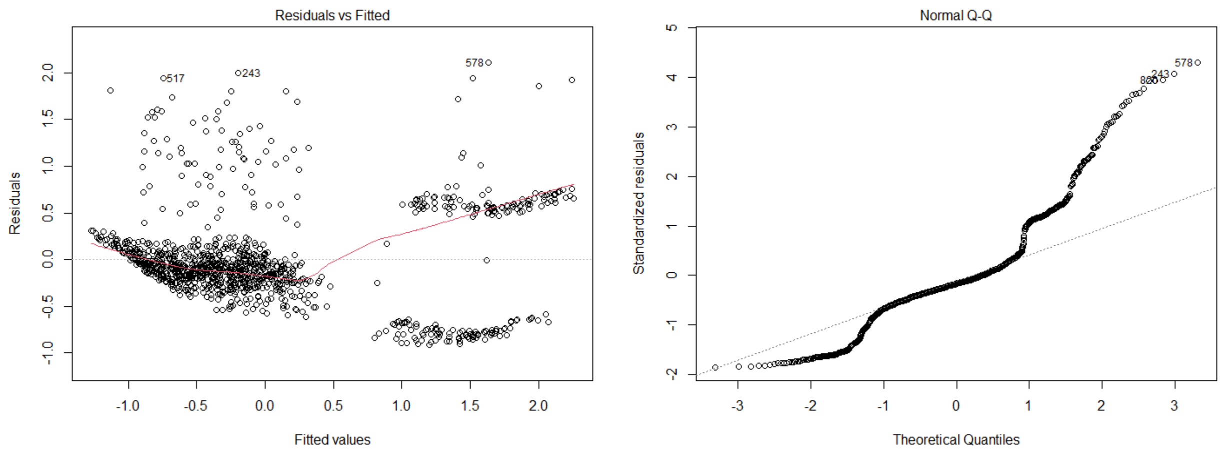
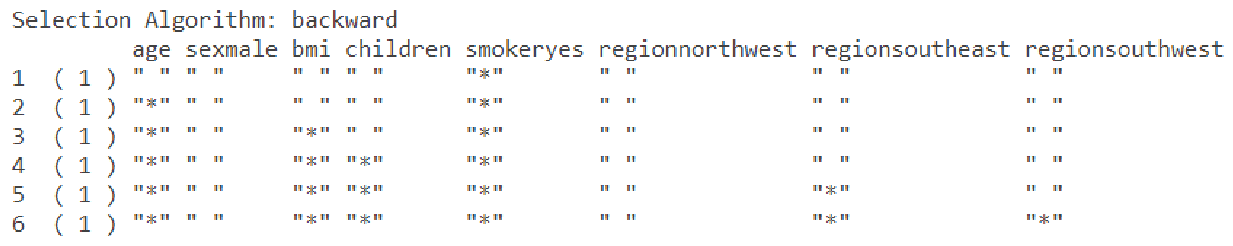


Fig. 3: Diagnostic plots of Linear Regression



```
Selection Algorithm: backward
         age sexmale bmi children smokeryes regionnorthwest regionsoutheast regionsouthwest
1  ( 1 ) " " " "     " " " "      "*"       " "             " "             " "            
2  ( 1 ) "*" " "     " " " "      "*"       " "             " "             " "            
3  ( 1 ) "*" " "     "*" " "      "*"       " "             " "             " "            
4  ( 1 ) "*" " "     "*" "*"      "*"       " "             " "             " "            
5  ( 1 ) "*" " "     "*" "*"      "*"       " "             "*"             " "            
6  ( 1 ) "*" " "     "*" "*"      "*"       " "             "*"             "*"            
```

Fig. 4: Output of stepwise regression on significant predictors

The the assumptions of the linearity of the fitted linear model were checked using the diagnostic plots in Fig. 3. It was observed in the residuals vs pitted plot of Fig. 3, the red line is not fairly linear, thus indicate of slightly non-linear patterns. It does deviate from the horizontal line but not severely, this could explain why the R2 value is 0.75 and not higher. However, the results are directly contradictory of the findings by the previous study [2], which had an accuracy rate of 81.3% using the dataset. From this study, with the diagnostic plot and R2 value, one can assume the predictions would be low but it maybe different approach used by the previous study [1] which used a 70/30 split and the dataset was not scaled. Additionally, since it is indicative of slight non-linear pattern, one can assume a MLP model to have better RMSE values, as MLP can better fit data which are not exactly linear. MLP models are found to be better in prediction with complex dataset, such prediction of harvest date, shelf-life of processed food and color changes in ginkgo biloba seeds, due its ability to better fit these data compared to MLR [8].

But it was observed that the RMSE value of MLP was 5092 compared to 6372 of MLR. It is a slight improvement of the RMSE by MLP. An RMSE value of 5092, would mean that on an average the prediction is off by 5092 USD, considering the range of the response variable is from 1,123 to 63,770. However, this is a high RMSE, as having almost 6000 USD off in medical insurance charge is not desirable. All the models had an RMSE between 5092 to 6465, with lasso and ridge regression having the same RMSE. MLP was better than the rest of the models with slightly less RMSE but overall, none of the models were able to predict the medical insurance charge very well. This brings the study to few considerations. First, it was seen not all the predictors are significant in Fig. 4, thus, an MLP model with the significant predictors may have better predictions. Second, no outliers were observed but more observations added to this dataset may improve the model's prediction accuracy. Thirdly, it is possible the standardization of the variables didn't work very well with MLP, instead for further studies, one can normalize using the minMax method for normalization.

## 5. Conclusion

MLR, stepwise, ridge, lasso and MLP models were developed to predict the medical insurance charge by using the dataset. The RMSE values for MLR, stepwise, ridge, lasso and MLP models were 6372, 6371, 6465, 6465, and 5092 respectively. MLP model had the lowest RMSE value compared to the other models. The R2 values for the four models except for MLP were 0.75. For further studies, one may use the significant predictors for creating the MLP model, try a different scaling method and have more data added to dataset.

## 6. References

1. Kulkarni M, Meshram DD, Patil B, More R, Sharma M, Patange P. Medical Insurance Cost Prediction using Machine Learning, Journal For Research in Applied Science and Engineering Technology, 2022. https://www.ijraset.com/research-paper/medical-insurance-cost-prediction-using-machine-learning
2. Bhatia K, Gill SS, Kamboj N, Kumar M, Bhatia RK. Health Insurance Cost Prediction using Maching learning, 5th International Conference on Computing Methodologies and Communication, IEEE, 2021. https://ieeexplore.ieee.org/document/9824201
3. Hassn CA, Iqbal J, Hussain S, Alsalamn H, Moshleh MAA, Ullah SS. A Computational Intelligence Approach for Predicting Medical Insurance Cost, Mathematical Problems in Engineering, 2021. https://www.hindawi.com/journals/mpe/2021/1162553/
4. Choi M. Medical Cost Personal Datasets [Internet]. Kaggle. 2018 [cited 2023Apr15]. Available from: https://www.kaggle.com/datasets/mirichoi0218/insurance
5. Zhang, Y. (2023) Week 1: Lecture Slides[PDF]. Thompson Rivers University:https://moodle.tru.ca/
6. Kassambara A, Gorenc J, Priya, Visitor. Stepwise regression essentials in R [Internet]. STHDA. 2018 [cited 2023Apr15]. Available from: http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/
7. James G, Witten D, Hastie T, Tibshirani R. An introduction to statistical learning: With applications in R. Boston: Springer; 2022.
8. Stangierski J, Weiss D, Kaczmarek A. Multiple regression models and artificial neural network (ANN) as prediction tools of changes in overall quality during the storage of spreadable processed Gouda Cheese. European Food Research and Technology. 2019;245(11):2539–47.

## 7. Appendix

# Appendix

Faisal

2023-04-15

```
library(ggplot2)
library(cowplot)
library(tidyverse)
library(caret)
library(dplyr)
library(tidyverse)
library(glmnet)
library(neuralnet)
library(MASS)
```

### CODE BLOCK A

```
DF = read.csv('./Data/insurance.csv')

#### Observe dataset and handle missing values ####

str(DF)
plot(DF)

#check for missing values
sum(is.na(DF))

#Check the outliers first
p1 <- ggplot(data=DF,aes(x=age))+geom_histogram()
p2 <- ggplot(data=DF, aes(x=bmi))+geom_histogram()
p3 <- ggplot(data=DF, aes(x=children))+geom_histogram()
p <- plot_grid(p1,p2,p3, ncol=2, labels="auto")
p



#boxplot of bmi
ggplot(data=DF, aes(y=bmi))+geom_boxplot()

#handle the outliers of bmi
# First Quantile:
Q1 = quantile(DF[,3])[2]
# Third Quantile:
Q2 = quantile(DF[,3])[4]
# Inner Quantile distance:
```

```r
IQR = Q2-Q1

# Lower bound Quantile Range:
lo = Q1-1.5*IQR
# Upper bound Quantile Range:
up = Q2+1.5*IQR
# Put outliers greater than upper quantile equal to upper quantile range:
if(sum(DF[3]>up)){
  indexu <- which(DF[3]>up)
  DF[[3]][indexu] = up
} else if(sum(DF[3]<lo)){
  indexl <- which(DF[3]<lo)
  DF[[3]][indexl] = lo}


#Checking BMI outlier again
ggplot(data=DF, aes(y=bmi))+geom_boxplot()

#### ####
```

## CODE BLOCK B

```r
#### EDA of the dataset and Scaling ####
#EDA for smoker
ggplot(data=DF, mapping = aes(x=smoker,y=charges,fill=smoker))+geom_boxplot()

#Charges and children
ggplot(data=DF, mapping = aes(x=as.factor(children), y=charges,))+geom_boxplo
t()

#Charges and sex
ggplot(data=DF, mapping = aes(x=sex,y=charges,fill=sex))+geom_boxplot()

#Charges and region
ggplot(data=DF, mapping = aes(x=region,y=charges))+geom_boxplot()

#Charges with sex and status of smoker
ggplot(data=DF, mapping = aes(x=sex,y=charges,fill=smoker))+geom_boxplot()

#Charges and BMI
ggplot(data=DF, mapping = aes(x=bmi,y=charges,color=bmi))+geom_point(size=5)

#### ####

##Scale the numeric variables ####
SDF <- DF
SDF$age <- (DF$age - mean(DF$age) ) / sd(DF$age)
SDF$bmi <- (DF$bmi - mean(DF$bmi) ) / sd(DF$bmi)
SDF$children <- (DF$children - mean(DF$children) ) / sd(DF$children)
SDF$charges <- (DF$charges - mean(DF$charges) ) / sd(DF$charges)
```

```
##Check for right scaling
max(SDF[,7])
min(SDF[,7])
mean(SDF[,7])
#### ####
```

## CODE BLOCK C

```
#### Linear Model ####
set.seed(7)

random_sample <- createDataPartition(SDF$charges,
                                     p=0.8,list=FALSE)
train.set <- SDF[random_sample,]
test.set <- SDF[-random_sample,]


LinearM <- lm(charges ~.,data = train.set)
summary(LinearM)


plot(LinearM)

predictions <- predict(LinearM, test.set)
predictiona = (predictions*sd(DF[,7])+mean(DF[,7]))
test.a = (test.set$charges*sd(DF[,7])+mean(DF[,7]))
data.frame(R2 = R2(predictions,test.set$charges),
           RMSE = RMSE(predictiona,test.a))
```

## CODE BLOCK D

```
## Step model ####

step.model = train(charges~.,data=train.set,
                   method="leapBackward",
                   tuneGrid=data.frame(nvmax=1:6),
                   trControl=train_control)

step.model$results
step.model$bestTune

summary(step.model$finalModel)

predictions = predict(step.model,test.set)
predictiona = (predictions*sd(DF[,7])+mean(DF[,7]))
test.a = (test.set$charges*sd(DF[,7])+mean(DF[,7]))
data.frame(R2 = R2(predictiona,test.set$charges),
           RMSE = RMSE(predictiona,test.b))
```

## CODE BLOCK E

```
## Ridge ####
set.seed(7)
```

```r
x <- model.matrix(charges~.,train.set)[,-1]
y <- train.set$charges

#Best lambda using cv
cv <- cv.glmnet(x,y,alpha = 0)
cv$lambda.min

#Fit the final model on the train set
ridge <- glmnet(x=as.data.frame(x),y,alpha = 0,lambda = cv$lambda.min)
coef(ridge)

#Make predictions on test set
x.test <- model.matrix(charges~.,test.set)[,-1]
predictions <- ridge %>% predict(x.test) %>% as.vector()
predictionsa <- (predictions*sd(DF[,7])+mean(DF[,7]))
test.a = (test.set$charges*sd(DF[,7])+mean(DF[,7]))
data.frame(
  R2 = R2(predictionsa,test.set$charges),
  RMSE = RMSE(predictionsd,test.a)
)
```

### CODE BLOCK F

```r
## Lasso ####
set.seed(7)
cv <- cv.glmnet(x,y,alpha=1)
cv$lambda.min

lasso <- glmnet(x,y,alpha=1,lambda=cv$lambda.min)
coef(lasso)
plot(cv)
#Make predictions on the test data
x.test <- model.matrix(charges~.,test.set)[,-1]
predictions <- lasso %>% predict(x.test) %>% as.vector()
predictionsa <- (predictions*sd(DF[,7])+mean(DF[,7]))
test.a = (test.set$charges*sd(DF[,7])+mean(DF[,7]))
data.frame(
  R2 = R2(predictionsa,test.set$charges),
  RMSE = RMSE(predictionsd,test.a)
)

## ####
```

### CODE BLOCK G

```r
## ElasticNet ####
set.seed(7)
cv <- cv.glmnet(x,y,alpha=0.5)
cv$lambda.min

elastic <- glmnet(x,y,alpha=0.5,lambda=cv$lambda.min)
```

```
coef(elastic)

#Make predictions on the test data
x.test <- model.matrix(charges~.,test.set)[,-1]
predictions <- elastic %>% predict(x.test) %>% as.vector()
predictionsa <- (predictions*sd(DF[,7])+mean(DF[,7]))
test.a = (test.set$charges*sd(DF[,7])+mean(DF[,7]))
data.frame(
  R2 = R2(predictionsa,test.set$charges),
  RMSE = RMSE(predictionsd,test.a)
)

## ####
```

## CODE BLOCK H

```
## Neural Network ####
dataset1 <- model.matrix(
  ~age+sex+bmi+children+smoker+region+charges,
  data=DF
)
dataset <- model.matrix(
  ~age+sex+bmi+children+smoker+region+charges,
  data=SDF
)

train1 <- model.matrix(
  ~age+sex+bmi+children+smoker+region+charges,
  data=train.set
)
test1 <-model.matrix(
  ~age+sex+bmi+children+smoker+region+charges,
  data=test.set
)

#Train NN
nn <- neuralnet(charges~age+sexmale+bmi+children+smokeryes+regionnorthwest+re
gionsoutheast+regionsouthwest,
                data=train1,hidden = c(6),
                linear.output = TRUE, threshold = 0.02,stepmax = 100000)

plot(nn)
#predict on test data
predict.nn <- compute(nn,test1[,-10])

#Caculate MSE
predict.nn2 <- predict.nn$net.result * sd(dataset1[,10]) +mean(dataset1[,10])
test.r <- (test1[,10])* sd(dataset1[,10]) +mean(dataset1[,10])
MSE.nn <- sum((test.r - predict.nn2)^2)/nrow(test1)
print(MSE.nn)
RMSE <- sqrt(MSE.nn)
```

```
print(RMSE)

## ####
```