Chi-Wen, Chen

Nikunj Agrawal

Suraj Shinde

Manav Iyanna Kuttaiah

Abstract

The objective of this project is to use statistical learning models to predict how often individuals wear a mask around people who do not live in their home using survey data

ISEN 613 Project

Texas A&M university

The objective of the ISEN 613 Spring 2023 Course Project II is to employ statistical learning models to predict mask-wearing frequency among individuals around non-cohabitants, utilizing survey data collected from A&M students during the early stages of the COVID-19 pandemic. To ensure the data's quality and compatibility for model training, we performed a data clean-up by imputing the median values into missing value cells (NA) and converting all columns to numeric types. This preprocessing step enabled us to effectively train both machine learning and deep learning (neural network) models on the dataset. The report highlights the development, evaluation, and comparison of various statistical models, ranging from basic methods like linear regression to more complex tree-based techniques. The project aims to determine the best model to predict mask-wearing behavior on a scale of 1 (Never) to 5 (Always), with performance measured using mean square error.

**Executive Summary**

Upon examining numerous models, it was revealed that random forest regression is not suitable for predicting a target variable with equal to or less than five unique results, as is the case in this project. Despite the unsuitability of the random forest model, elastic net and lasso regressors emerged as two of the best machine learning algorithms for this task. However, their performance was still notably worse than that of the neural network model, which is a deep learning technique.

The neural network model demonstrated the best performance among all tested models, given its ability to effectively handle the complexity of the dataset, which consists of 154 input variables and one target variable. The neural network model's success is derived from its ability to consider the complex relationships among all input variables, making it difficult to pinpoint specific features as the most crucial. Moreover, the neural network model yielded a test error of 0.8844608, highlighting its efficient predictive capability.

In conclusion, the top-performing model for predicting mask-wearing behavior is the deep learning-based neural network, which outperforms other machine learning algorithms, such as elastic net and lasso regressors. The chosen model can effectively predict mask-wearing frequency among individuals, ultimately supporting public health efforts to mitigate the spread of COVID-19, even with a relatively small dataset.

**Contributions**

1- Chi-Wen, Chen: GLM Model (Lasso, Ridge, Elastic Net), Preprocessing (Dealing with N/A)

2- Suraj Shinde: PCA, Neural Network, Preprocessing (lags)

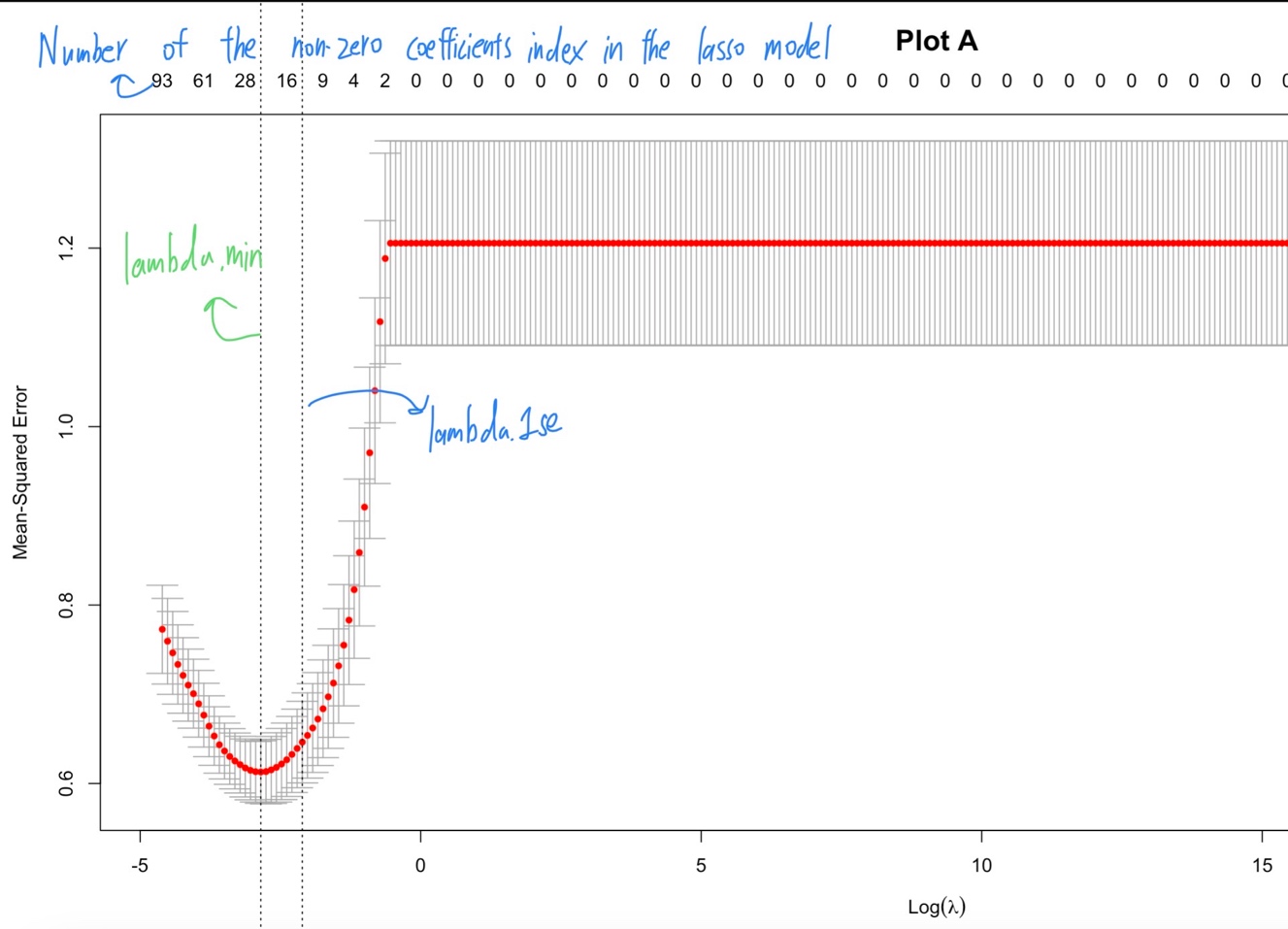
3- Manav Iyanna Kuttaiah: SVM with 4 kernels and tuning, Forward subset selection

4- Nikunj Agrawal: Random Forest, Tree, Pruned tree, and Forward subset selection

**First model: Lasso regressor**

**Technical Summary:**

We aim to perform Lasso regression and evaluate the performance of the model on the train and test sets, and also identify the most important variables for predicting the target variable. The dataset is split into train and test sets, with 70% of the data used for training and the remaining 30% for testing.

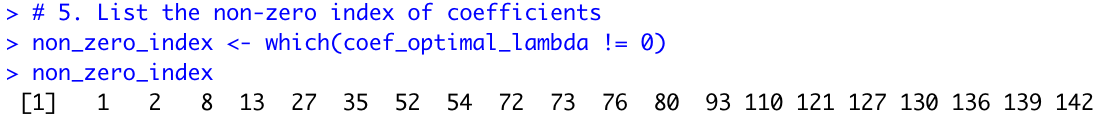


**Summary for plot A:**

**1.** This line of code fits a Lasso regression model with cross-validation using the cv.glmnet() function from the glmnet package. The purpose of this code is to find the optimal lambda value for the Lasso regression by minimizing the cross-validated mean squared error (MSE).

**2.** A range of lambda values is generated using 10^seq(10, -2, length = 200), where lambda is the regularization parameter in Lasso regression.( lambda\_seq <- 10^seq(10, -2, length = 200)

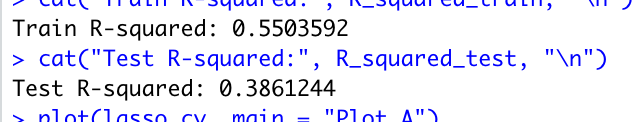
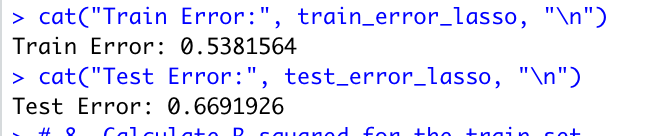
**3.** The x-axis represents the log10(lambda) values, which it ranges from a lower value (less regularization) to a higher value (more regularization). On the other hand, the y-axis represents the cross-validated mean squared error (MSE). The value on the top of the plot represents the number of the non-zero coefficients indices that we obtained from the glmnet model with the optimal lambda.

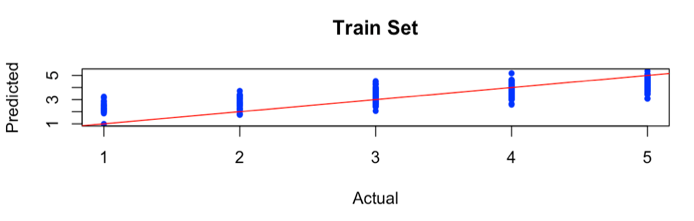
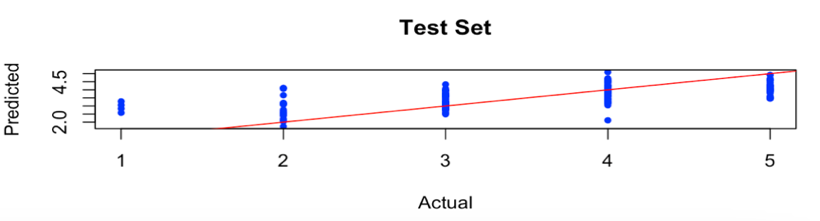
****

**4.** From left to right on the x-axis, the lambda value increases, resulting in more regularization in the Lasso regression model. This means that more coefficients are being shrunk towards zero, and some of them become exactly zero, effectively eliminating some variables from the model.

**5.** The curve in the plot A represents the cross-validated MSE for different log10(lambda) values. The goal is to find the lambda value that minimizes the cross-validated MSE, striking a balance between underfitting (high bias) and overfitting (high variance).

**6.** There are two vertical dashed lines in the plot. The first dashed line (from left to right) represents the lambda value that gives the minimum cross-validated MSE (lasso\_cv$lambda.min). This is the optimal lambda value that provides the best model performance. The second dashed line represents the lambda value that gives the most regularized model, such that the cross-validated MSE is within one standard error of the minimum cross-validated MSE (lasso\_cv$lambda.1se). This lambda value represents a simpler model with slightly worse performance but potentially better generalization to new data.





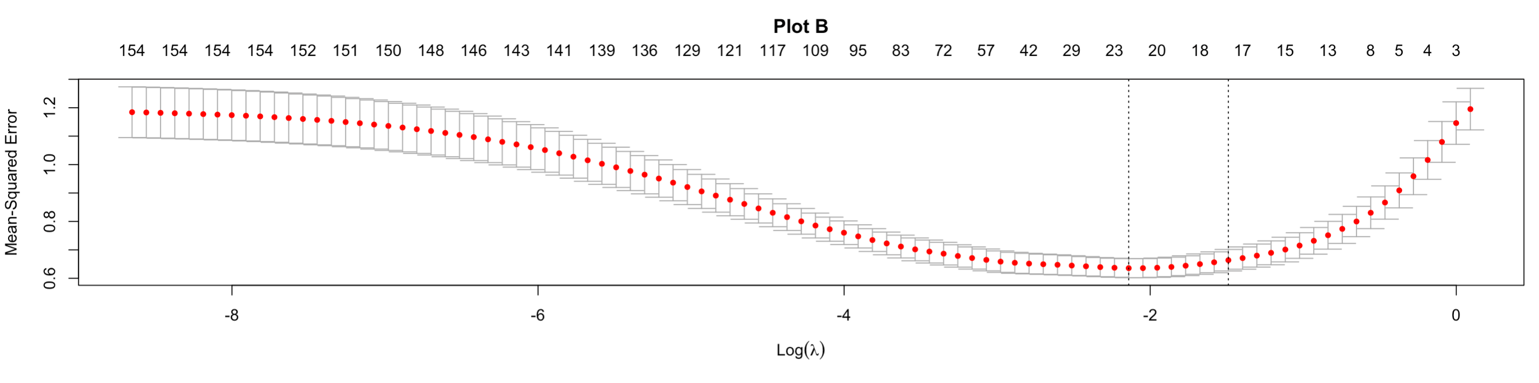
**↑** The red line represents the ideal situation where the predicted values perfectly match the actual values. The blue dots represent the predicted values of the target variable, and the y-axis shows the corresponding actual values of the target variable. Each blue dot represents a single data point.Since we’ve already known the model performs poorly, the dots scattered far from the red line

Based on the Lasso regressor model trained on the dataset, the test R-squared value was found to be 0.3861244, which indicates that the model explains approximately 38.61% of the variation in the target variable. The test error was found to be about 0.67, which means that on average, the predictions of the model are off by 0.67 units from the actual values.

For the sake of the observation above, these values suggest that the Lasso model may not be a good fit for the given dataset even after multiple approach of tuning, as the R-squared value is relatively low and the test error is relatively high. The further analysis may be needed to determine if the model is suitable for the given task.

**Second model: Elastic net regressor**

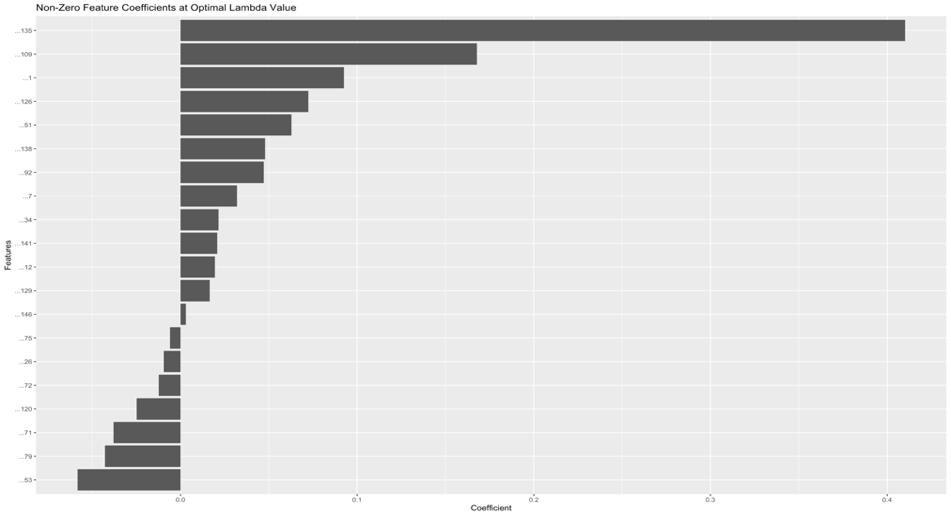
Our Elastic Net model is implemented using the glmnet and caret packages in R. The Elastic Net model is a regularized regression technique that combines both Lasso (L1) and Ridge (L2) regularization methods. The objective is to find a balance between these two regularization techniques to prevent overfitting while retaining important features.



**Summary for Plot B:**

1.In this model the dataset is split into train and test sets with also 70% of the data used for training and the remaining 30% for testing as we do it Lasso model. A 10-fold cross-validation is performed to find the optimal values of the lambda (regularization parameter) and alpha (mixing parameter) for the Elastic Net model.

2. The lambda value that minimizes the cross-validated mean squared error (MSE) is chosen as the optimal lambda, and an alpha value of 0.5 is used, representing an equal mix of Lasso and Ridge regularization.

3. The plot provides insights into the trade-off between model complexity and regularization strength. As lambda increases (moving right along the x-axis), the model becomes more regularized, leading to a simpler model with fewer non-zero coefficients.

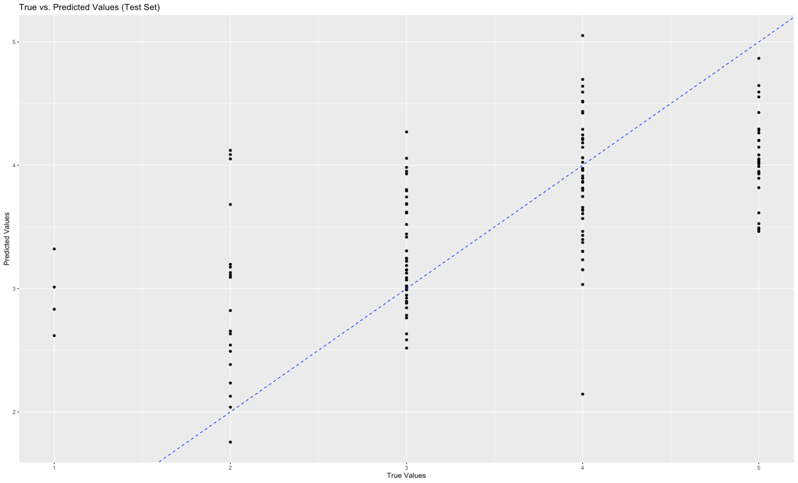
**Plot C:** This bar plot shows the coefficient values for each feature in the Elastic Net model using the optimal lambda value. The magnitude of each coefficient represents the importance of the corresponding feature in the model. A larger magnitude indicates a stronger effect on the model's predictions.

**Plot C**

**Plot D and Summary:** The scatter plot is titled "True vs. Predicted Values (Test Set)" compares the true target values (y\_test) with the predicted values generated by the Elastic Net model (y\_test\_pred\_elastic\_net) on the test dataset. In this plot, the x-axis represents the true values, while the y-axis represents the predicted values. Each point on the plot represents a single observation in the test set, and the location of the point reflects the true and predicted values for the observation(model).

Meanwhile, the dashed blue line in the plot with a slope of 1 and an intercept of 0 is also plotted, which represents the ideal scenario where the true values are equal to the predicted values. For example, if all the points lie exactly on this line, it indicates a perfect prediction.

The scatter plot explained the model's performance on unseen data. the model would accurately predicting the target variable If the points are closely clustered around the dashed blue line. Conversely, if the points are scattered and far from the line, it suggests that the model's predictions are not very accurate.

In this specific scenario, the points in the scatter plot appear to be significantly scattered, indicating that the Elastic Net model's performance on the test set is relatively bad. This observation aligns with the test R-squared value of 0.389 and the test error of 0.667, both of which suggest that the model's predictive performance is very likely to be not the best model.

**Plot D**

A picture containing text

Description automatically generatedA picture containing graphical user interface

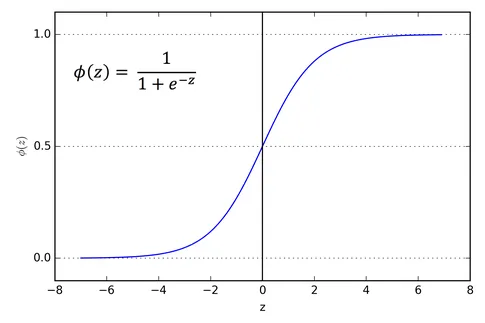
Description automatically generated

**Third Model: Neural Network**

As one of the deep learning algorithms, neural networks have demonstrated exceptional performance across a wide range of tasks by leveraging their capacity to learn complex patterns and representations from large volumes of data. By utilizing it in our analysis, we aim to capitalize on its inherent strengths and capabilities, enabling us to generate more accurate and reliable predictions.

##working of a neural network.

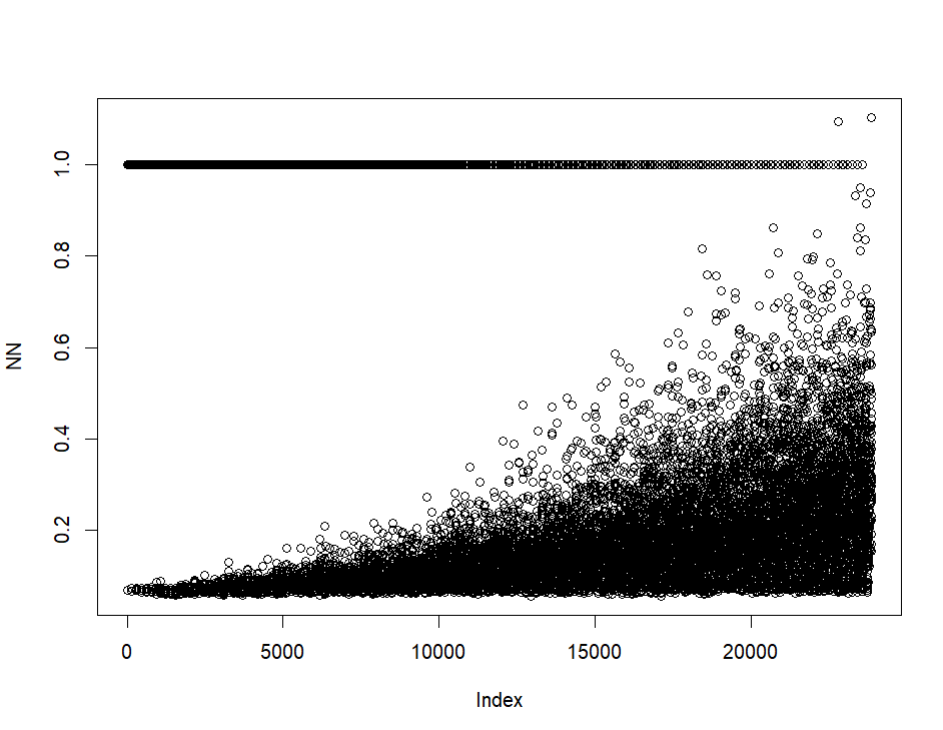
The code splits the dataset into training and validation sets using a 70-30 ratio, controlled by a random seed for reproducibility. We first started building the neural network by keeping a sigmoid activation function, which is basically predicts/ spits output in 0 to 1 range.



Courtesy- [Activation Functions in Neural Networks](https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

For example, if the inputs are in negative the resulting output will be close to 0 and if the input is anywhere in the positive range, it will give a value closer to 1. The sigmoid function in our model is applied using (function(x) 1.0 / (1.0 + exp(-x))). In order to make useful predictions we make normalize the data, along with that the run time for the code reduces, drastically upon using a normalized data. We then follow for effect, which will be sum of square (as it is a regression problem). We want to apply the sigmoid function to the final output layer, so as to make the predictions in the desired range that 0 and 1. In order to receive a hidden layer which provides us with the lowest test Mean Square Error (MSE).





**Plot E**

**Plot E:** The plot visualizes the test mean square errors for various combinations of neurons in the hidden layers. This plot helps us to understand the impact of changing the number of neurons in each hidden layer on the model's performance. The plot should show a pattern where the lowest test MSE values are found around the optimal neuron combination (in the range of 1000 -15000 index), we get the exact number of neurons using the matrix and finding the exact location of minimum Test MSE.

After finding all the possible combinations and after the completion of hidden layers we are able to get the lowest Test MSE (0.0559). we then put the values in the matrix and look for the corresponding value. the row represents the number of neurons in hidden layer 1 and columns represents the Number of neurons in hidden layer 2, we get that the lowest comes out to be at 15 and 85 neurons in hidden layer 1 and hidden layer 2 respectively.

Train\_MSE# Train MSE - 0.05644622

## [1] 0.05644622

#Finding the test MSE

K<-compute(nn, test[, 1:154])$net.result  
Test\_MSE <- sum((K-test[, 155])^2)/nrow(test)  
Test\_MSE# Test MSE - 0.05597384

## [1] 0.05597384

#Finding the R-square value

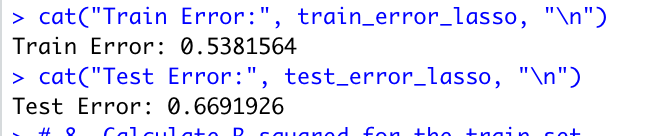
After applying all these changes, we are receiving the test mean square error as 0.05597384.

**Best Model Summary**

In this study, we compared the test error performance of three different models, including Lasso regression, Elastic Net regression, and a Neural Network, on a dataset with 154 predictors and over 400 observations. We treated the target variable as a continuous outcome, thus framing the problem as a regression task. Our objective was to find the most suitable model for predicting the target variable, considering the trade-offs between computational complexity and predictive accuracy.

The test error results of the comparison are as follows:

**↓Lasso**

1. Lasso Regression:

- Test MSE: 0.6691926

- Train MSE : 0.5381564

**↓Elastic Net**

A picture containing text

Description automatically generated2. Elastic Net Regression:

- Test MSE: 0.6667404

- Train MSE: 0.5434317

**↓Neural Network**

3. Neural Network:

## [1] 0.05597384##Test MSE(Normalized Data Set)

- Test MSE(N): 0.05597384

- Train MSE(N): 0.0564462

From the test error results, it is evident that the Neural Network model significantly outperforms both Lasso and Elastic Net regression models in terms of predictive accuracy. Although the Neural Network model is computationally more expensive, it yields a much lower test error (0.05597384) in comparison to the Lasso (0.6691926) and Elastic Net (0.6667404) models.

Considering the large number of predictors (154) and the limited number of observations (over 400), the Neural Network's ability to capture complex relationships and interactions among predictors plays a critical role in achieving better predictive performance. The advantage of using a deep learning/neural network approach in this scenario is its capacity to automatically learn feature representations and model complex, non-linear relationships without relying on manual feature engineering or strict assumptions about the underlying data distribution.

In conclusion, despite its higher computational cost, the Neural Network model is the preferred choice for this dataset due to its superior test error performance. The enhanced predictive accuracy provided by the Neural Network model outweighs the additional computational resources required, making it the most suitable model for this specific regression task.

**Evaluating the test points (Normalized Dataset)**

#Normalizing the data.  
normalize <- function(x) {  
 return ((x - min(x)) / (max(x) - min(x)))  
}   
Train <- as.data.frame(lapply(Trd, normalize))  
  
normalize <- function(x) {  
 return ((x - min(x)) / (max(x) - min(x)))  
}   
Test <- as.data.frame(lapply(TS, normalize))  
  
nn <- neuralnet(Train$X155 ~ ., data =Train, **hidden = c(16,29),**  
 act.fct =(function(x) 1.0 / (1.0 + exp(-x))),err.fct = "sse",  
 linear.output = T,threshold=0.7,stepmax = 10e5)

#Finding the test MSE  
K<-compute(nn, Test[, 1:154])$net.result   
Test\_MSE <- sum((K-Test[, 155])^2)/nrow(Test)  
Test\_MSE# Test MSE - 0.0552788

## [1] 0.0552788

#Finding the R-square value  
RSS<-sum((Test[, 155]-K)^2)  
TSS<-sum((Test[, 155]-mean(Test[, 155])^2))  
1-(RSS/TSS)# R- square value 0.7676385

## [1] 0.7676385

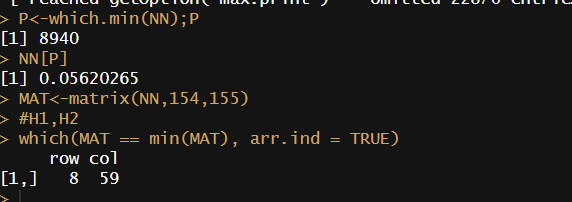
**Evaluating the test points (Denormalized Dataset)**

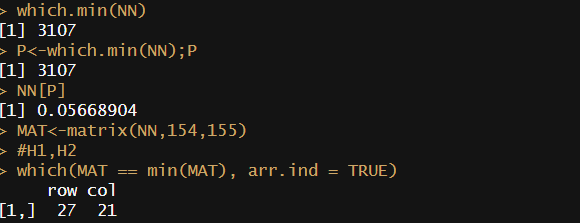
nn <- neuralnet(Train$X155 ~ ., data =Train, hidden = c(16,29),  
 act.fct =(function(x) 1.0 / (1.0 + exp(-x))),err.fct = "sse",  
 linear.output = T,threshold=0.7,stepmax = 10e5)  
  
#from the model build using the train data  
  
K<-compute(nn, Test[, 1:154])$net.result  
  
## denormalize the data in order to receive the test MSE  
  
x=1:5  
denormalized = (K)\*(max(x)-min(x))+min(x)  
  
#Finding the test MSE  
Test\_MSE <- sum((denormalized - TS[, 155])^2)/nrow(TS)  
Test\_MSE# Test MSE - 0.8844608

## [1] 0.884460

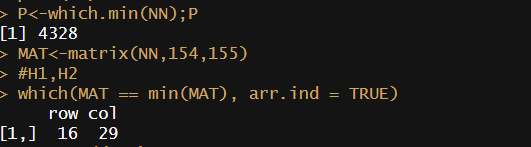
**Steps to improve the best model**

Using different settings(hyperparameters in model nn), we received different value of test MSE





We will use this model setting and will place hidden neurons in hidden layer 1 as 16 and number if neurons in hidden layer 2 as 29.



nn <- neuralnet(Train$X155 ~ ., data =Train, **hidden = c(16,29),**  
act.fct =(function(x) 1.0 / (1.0 + exp(-x))),err.fct = "sse",  
 linear.output = T,threshold=0.7,stepmax = 10e5)

**Results for Normalized Dataset**

Conclusion: The difference in this model and the previous model is **the number of hidden layers** were being changed and from [15,85] to [16,29] and the threshold value being changed from 0.6 to 0.7. we have managed to receive a lower test MSE in of **0.0552**, while previously we obtained the MSE of **0.0559**. Hence for this case we did obtain the subtle improvement for the output.

**Results for Denormalized Dataset**

The test MSE that we managed to receive for the denormalized data set via neural network is 0.8844608 and the R-square value comes out to be 0.3014.