The project code consists of 5 main parts:-

Hypertuning.ipynb:-

The code was used to hypertune some parameters using keras tuner. The parameters are : no of hidden layers(from 2 to 20), the number of neurons at each layer(from 5 to 50, keeping the steps = 5). The loss function was taken mean absolute error, and the activation function was relu for hidden layers, and linear for output function. The keras randomly searched from the given search space by keeping max trials 100(that means it will randomly select model 100 times) and executions per trial to 3. The search was performed using x\_train and y\_train to train, and x\_test and y\_test to to, which was created by randomly separating the data in 4:1 format. After the search, the optimal number of epochs is searched out of 500 epochs, based on minimum loss. The info about the optimal result is given below.

Optimal number of epochs: 487

Optimal number of hidden layers: 8 (9 including the output layer)

Optimal learning rate: 0.01

Optimal neurons each layer: 40, 50, 50, 35, 30, 45, 15, 35 and 1 (including the output layer)

Mean error: 13.88422934214274

Optimal\_activation\_function\_and\_optimizer.ipynb:-

The part of code used GridSearchCV to find best for the rest parameters. The parameters are: activation function( among relu, softmax, tanh, sigmoid) and optimizer(among adam, SGD, RMSprop). The search was performed using x\_train and y\_train to fit, which was created byrandomly taking 80 percent of the data. The number of epochs was kept at 487. The info about the optimal search is given below.

Mean error: 10.736630

Optimal activation function: relu

Optimal optimizer: adam

Create\_model.ipynb:-

The part of code was used to create the model according to best parameters. However, applying the optimal learning rate leaded to not decreasing the error value. So the learning rate was left to default, leading to proper reduction of loss. Later the model was tested with the test data, which have a graph representation in the ipynb file.

The value of error after 487 epochs: 1.5137

One hot.ipynb:-

The code was used to convert non numerical data types to different columns with Boolean value using one hot encoding.

GA.ipynb:-

The code used geneticalgorithm from python library that took limits for all the inputs and the pre saved model and weight file. The fitness function used the model to predict the output and return the fitness value. A graph is implemented to show the predicted values vs the given values. The values related to GA: population size=100, mutation probability=0.1, crossover probability=0.5, crossover type=uniform. The final value of each parameter and value of the time according to them is given below.

[ 5. 60. 0. 0. 1. 0. 1. 0. 1. 0.]

Objective function:

-73.95179748535156

Predicted output:

73.95179748535156