Neural Networks Project

Name: Jay Jagtap
Student ID: 200311438

1. **Training and Testing Set Distribution:**
   As described in the assignment, we take first 2000 samples as training samples and the remaining samples are.
   We check the corresponding mean, and quantile ranges between training and test set. They are found to be commensurate and thus we can say that the training and testing set are analogous and represents the same data distribution. Please check .ipynb for dataset statistics.

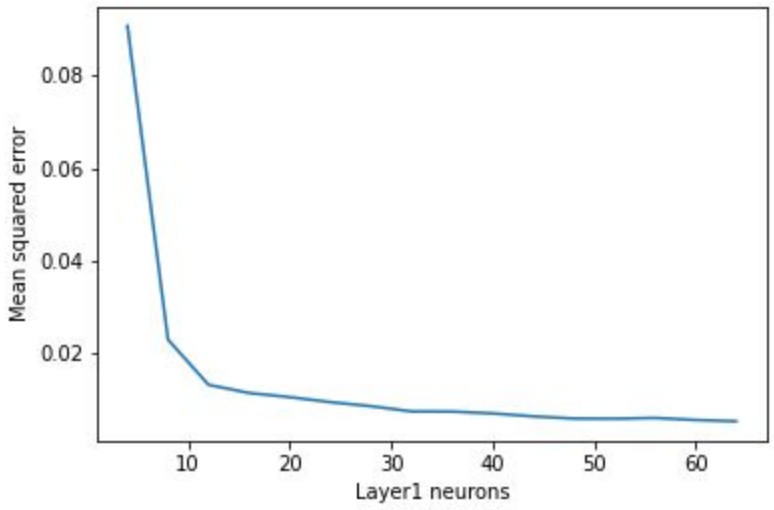
2. **Scaling**
   The raw numbers in the data are big in magnitude, hence we use scaling to bring the numbers in scale.
   Formula: z = (x - u) / std
   z is the new data point
   x: old point
   std: standard deviation

3. **Hyperparameter tuning Strategy**
   As explained by the professor in section 10.10 of the book: Hyperparamter selection, we use the same strategy. Following steps are followed to select the best hyperparamerters:

   1. Run a gridsearch to select the number of neurons in the first layer.
   2. Run a gridsearch for subsequent layers to find optimal number of neurons. (2nd layer , 3rd layer, 4th layer...and so on)
   3. Stop adding layers, once you detect saturation.
   4. Compare Neural Networks with different number of layers, select the best.
   5. Use gridsearchCV for finding redularization and learning rate on the selected model.
   6. Finally find the best number of epochs to fit the model.

4. **GridSearch for 1st Layer of Neural Network:**

   Given the nature of the data, we give the following range for number of neurons for 1st layer:
   units: [4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64]
   We run the model for 10 epochs each and we calculate the MSE(mean squared error for each instance, following is the plot of the graph:

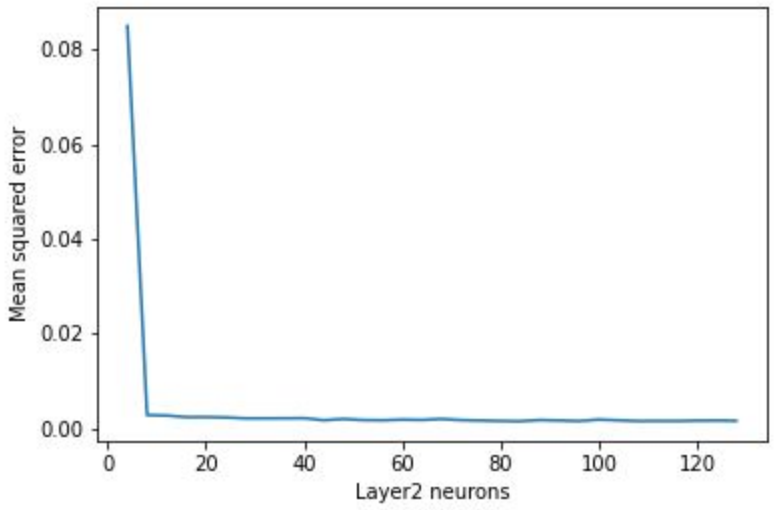


We can see that, the mse decreases rapidly as we increase the number of neurons, we select 64 neurons as it gives the best parameters we still have a signiuficatn slope.

MSE and Best Param for layer 1: 0.005429 using {'units': 64}

5. **Layer 2 GridSearch number of neurons.**
   We can see that layer 1 fits well and is able to capture the variance and patterns in data, we now test by adding layer 2 to layer 1
   Range of Neurons for layer2: [4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128]

   We get the following MSE vs neurons plot for Layer 2:

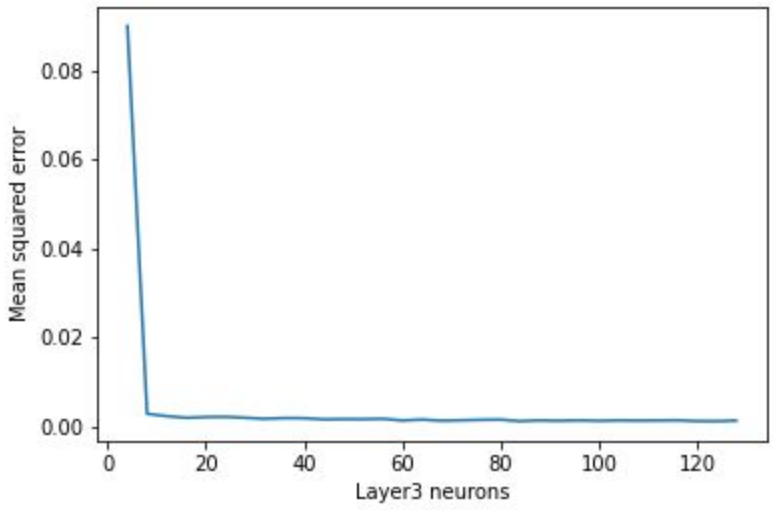


We get a elbow pont at neurons = 16, we see flat line after neurons =16 and hence we select the value at the elbow point

Layer 2 Neurons: 16

6. **3rd layer of Neural Network:**

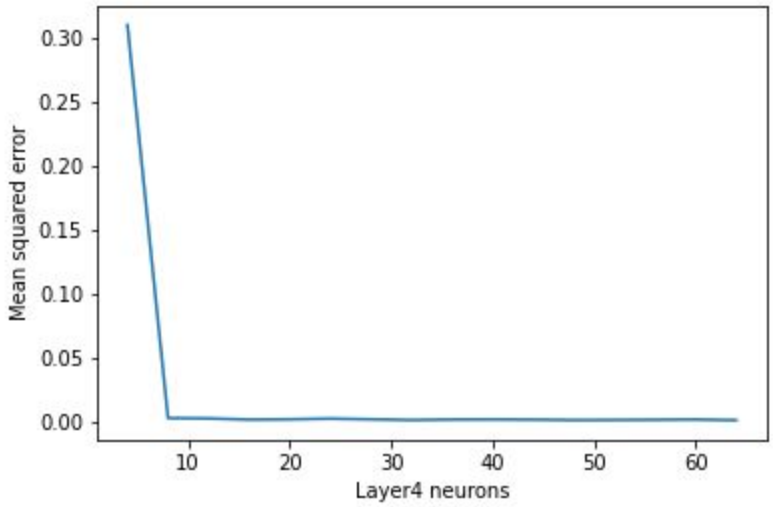
We add layer 3 to the previous 2 layer network.



We get a elbow at neurons = 12, we add model.add(Dense(12)) to the neural network

7. **4th layer of Neural Networks**

We add the 4th layer to the previous 3 layers and perform the same experiment.

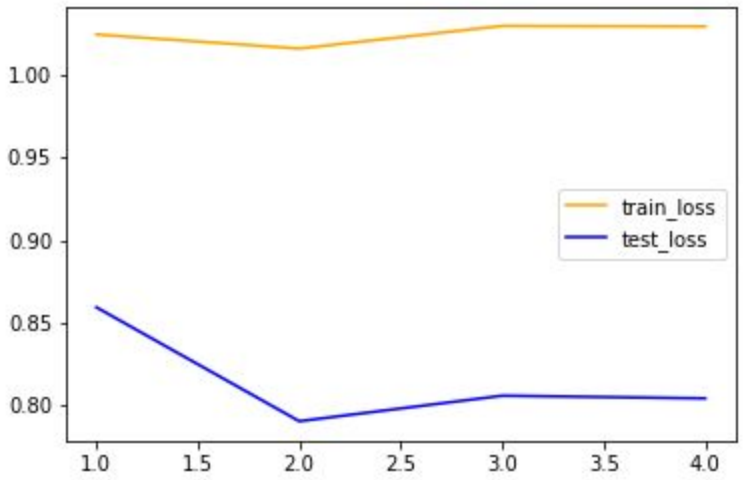


elbow point at neurons = 8

We can see that after first 2 layers, the number of neurons for the subsequent layers keep on decreasing, which suggests that neural network is able to capture the data patterns without adding any further layers.

8. **Compare Neural Network models with 1,2,3, and 4 hidden layer**

We save models obtained in step 4,5,6 and 7 and run them against test and train data. Following is the plot of MSE vs number of layers:



Based on the following graph, we can see the test and training MSE increases at layers = 2 . so we select the 2 layer Neural Network as our best model.

Model Summary

Layer (type)	Output Shape	Param #
dense_613 (Dense)	(None, 64)	384
dense_614 (Dense)	(None, 16)	1040
dense_615 (Dense)	(None, 1)	17

Total params: 1,441
Trainable params: 1,441
Non-trainable params: 0

9. **Gridsearch for combination of optimizer, learning rate and legurization parameters**

Now that we have finalized the number of layers and corresponding neurons, we run gridsearech for the best combination of optimizer, regularization and learning rate.

In brief, optimizers change how gradient descent behaves.
learning rate determines how fast the gradient descent algorithm finds global minima, a low value will take a lot of time to converge and on the other hand, a high learning rate makes the algorithm difficult global minima as it oscillates between local minimas.

Regularizers allow you to apply penalties on layer parameters or layer activity during optimization.
For a learning rate we have a range from 0.001, 0.1 with increments of 0.01. We prefer, a greater learning rate if it gives the same result as it helps in a faster convergence.

We do a extensive search for the above parameters with the following ranges:

```
learning_rate = np.arange(0.001, 0.1, 0.01)
regularizer = [0.1,0.01, 0.001, 0.001, 0.0001]
optmizers=[keras.optimizers.Adam,keras.optimizers.SGD,keras.optimizers.RMSprop]
```

A total of 150 combinations are documented in the .ipynb file.
We get the best params as follows:
{'lr': 0.02,
'optimer': <class'tensorflow.python.keras.optimizer_v2.adam.Adam'>,
'regulartiztion': 0.001}

10. Find best number of epochs to be trained the model for:

Following is epochs and MSE
{'nb_epoch': 10} 0.008706
{'nb_epoch': 20} 0.010005
{'nb_epoch': 30} 0.008081
{'nb_epoch': 40} 0.009148
{'nb_epoch': 50} 0.005879
{'nb_epoch': 60} 0.006845
{'nb_epoch': 70} 0.007324
{'nb_epoch': 80} 0.006449
{'nb_epoch': 90} 0.007685

Best MSE for epochs = 50

11. Calculate MSE on the test set with the Neural Network.

Mean Squared Error for Test DataSet: **0.008**

12. Calculate the coefficient for Multivariable Regression and evaluate with test dataset:

regression Coefficients: [[0.93991494 0.09039432 0.12019567 0.14235682 0.19749983]]
regression intercept: [0.00232456]
r2 value(Regression score): 0.9686213971459987**
Mean Squared Error for Test DataSet: 0.022677513667044066

13. Comparison between ANN and Multivariable regression

Even though multivariate regression model gives a decent MSE of 0.02, the Neural Network model far outperforms it with MSE 0.008, in terms of sheer numbers it is 88% better than the multivariate regression model.
This is possible as our neural network has a total of 1441 trainable parameters as compared to 5 regression coefficients of multivariable regression model(). The activation functions introduce non linearity in the data which allows to capture the variance better. Also, all trainable parameters are updated with every forward and backward pass of the algorithm. This makes the ANN much superior to the multivariable regression model.