**Summary of the Regression Task using Neural Network**

**Data Pre-processing**

Importing of libraries; seaborn, matplotlib, pandas and numpy and loading of the csv file into pandas’ data frame, the pandas dataframe displayed the 3 input features(numerical variables) and Unamed:3 column with missing values, and the output feature( continuous variables). The Unamed:3 column was deemed irrelevant and dropped from the dataframe. The output column was renamed as target.

The dataframe has 600 rows and 4 columns. A python function was created to check for missing values which gives none. The dataframe has 447 datapoints which were duplicated but are relevant to the modelling. The information of the dataframe was checked by creating a python function, all the 4 features have non-null int. values. A python function was created to check the unique values in the dataframe. The statistical summary of the dataframe was also evaluated using python function that gives the count, mean, std, min, max, 25%, 50%, 75% of the features.

Functions were created to check the numerical and categorical features of the dataframe, the categorical function return an empty list, meaning none. Strip function was used on the dataframe to ensure extra spaces were removed. All the features are numerical variables( continouse).

Outlier detection was done using the IQR score; The IQR is the first quartile subtracted from the third quartile; where Q3 is the 75th percentile of the data and Q1 is the 25th percentile of the data. Hence IQR = Q3-Q1. Outlier were identified by defining a threshold range as follows.

* (Lower threshold = Q1 - 1.5 \* IQR)
* (Upper threshold = Q3 + 1.5 \* IQR)

Hence any data points that fall outside of this range are considered outliers and were removed.

**Feature scaling**

Feature scaling of the input features was done using Standardscaler for standardization. This is because it helps to normalize the data and make it easier for machine learning algorithms to converge during training. Standardizing the features to a particular range can also help to prevent some features from dominating others in the learning process.

**Train\_test\_split**

The data was divided into x and y, x was assigned to the input variables by dropping the target feature from the dataframe, and y was assigned to the target feature. Then split into training and testing set in ratio 80:20 with training set as 80 and test set as 20, random was set to 42 to ensure the results were reproducible upon iteration.

**Neural Network Model Architecture**

Importing of libraries and using a sequential model with three dense layers. The first layer has 16 neurons and expects an input of 3 features. second layer has 8 neurons, and the output layer has just 1 neuron, and relu activation function for the hidden layers.

linear activation function for the output layer since we are dealing with continuous variables. The loss function used is mean squared error, which is commonly used for regression problems. The optimizer used is Adam, which is an efficient stochastic gradient descent algorithm, mean absolute error and mean squared error, root mean squared error as evaluation metrics.

Model Training was done by fitting the model on the training set using 100 epochs, batch size 16. Evaluation of the model performance was done use the following metrics.

* Mean squared error (MSE)= 2.060
* Root mean square error (RMSE) = 1.435
* Mean absolute erorr(MAE) = 1.131

Mean squared error (MSE) = 2.060: This is a measure of the average squared difference between the predicted and actual values. A lower MSE indicates that the model is performing better, with less error.

Root mean square error (RMSE) = 1.435: This is the square root of the MSE, and it provides a measure of the average difference between the predicted and actual values in the same units as the target variable. A lower RMSE indicates better performance and a smaller error.

Mean absolute error (MAE) = 1.131: This is a measure of the average absolute difference between the predicted and actual values. It provides a similar indication of the model's performance as MSE and RMSE, but with the difference being absolute instead of squared. A lower MAE indicates better performance and a smaller error.

Overall, the results suggest that the Neural Network model has a moderate to good level of performance in predicting the target variable, with relatively low levels of error.

The learning curve was plotted with Loss on the y-axis and Epoch on x-axis, the learning curve is a plot of the model's performance (usually measured by the loss function which is the mean square error) against the number of epochs (or iterations) of training.

During the training process, the model learns from the training data, and its performance gradually improves as it adjusts its weights and biases to minimize the loss function. The learning curve shows how the performance of the model changes over time as it is trained with more and more data. The plot shows if the error rate is reducing over epochs and at which epoch the model performance is optimal, so the lower the loss becomes, the better the model performance will be.

The lower the value of the loss function, the closer the predicted output is to the actual output, which means that the model is performing better. Therefore, if the loss function is lower, it indicates that the model has a better fit to the data and is making more accurate predictions. This means that the performance of the regression model is improving, as it is becoming better at predicting the target variable.

Based on the learning curve plot, the model's performance improves during the first few epochs, but eventually reaches a plateau around epoch 30. The loss (MSE) on the training data continues to decrease over time, while the loss on the validation data appears to level off around epoch 60. Based on the plot, the best epoch seems to be around epoch 100, as this is where the validation loss is lowest, and equivalent time duration of 10 seconds.