****

**APPLIED ARTIFICIAL INTELLIGENCE (COMP-534)**

**ASSIGNMENT – II**

**Developing a Neural Network for Solving a Regression Problem**

**Kshitij Singh (Student ID: 201602136)**

**Amol Sahebrao Rathod (Student ID: 201593889)**

**Introduction**

**The libraries used in the program were:**

1. **Numpy** is used for mathematical calculations on matrices and arrays.
2. **Pandas** is used for handling data in the form of DataFrames.
3. **Matplotlib.pyplot** and **Seaborn** makes data visualization possible.
4. **Scikit-Learn** contains machine learning tools and algorithms.
5. **Tensorflow** is another machine learning and AI library which focuses mainly on tasks related to Deep Learning.
6. **Keras** contains tools for the implementation of artificial neural network building blocks. It provides a python interface for tensorflow.

The aim of this assignment is to train an artificial neural network for a regression problem of predicting price of houses and further use hyperparameter tuning to achieve the best possible model. The dataset used is House Sales in King County, USA from Kaggle.

Steps in **Data Cleaning**:

* The data was imported first and analyzed. There were no null values in the dataset.
* There was an extra column ‘id’ in the dataset which was to be removed.
* ‘Date’ column instances were changed to datetime format and year and months were extracted as separate columns.
* ‘zip code’ column was not used as it didn’t serve any purpose and there were latitude and longitude columns present in the data.
* Correlation was checked among the columns and it was found that ‘sqft\_above’ column had high correlation with ‘sqft\_living’ which in turn had high correlation with the target variable ‘price’. Hence, ‘sqft\_above’ column only had redundant information and it was also removed.
* The median price of houses was checked for each month and since it didn’t vary much with it, the month column was also removed.

Steps in **Data Preparation**:

* Dataset was divided into feature matrix and target variable array.
* Then they were randomly divided into Training, Validation and Test set with ratio 70:20:10 respectively.
* All three sets of the feature matrix were scaled between [0,1]
* A function to evaluate metric scores and loss graph was created for a model.

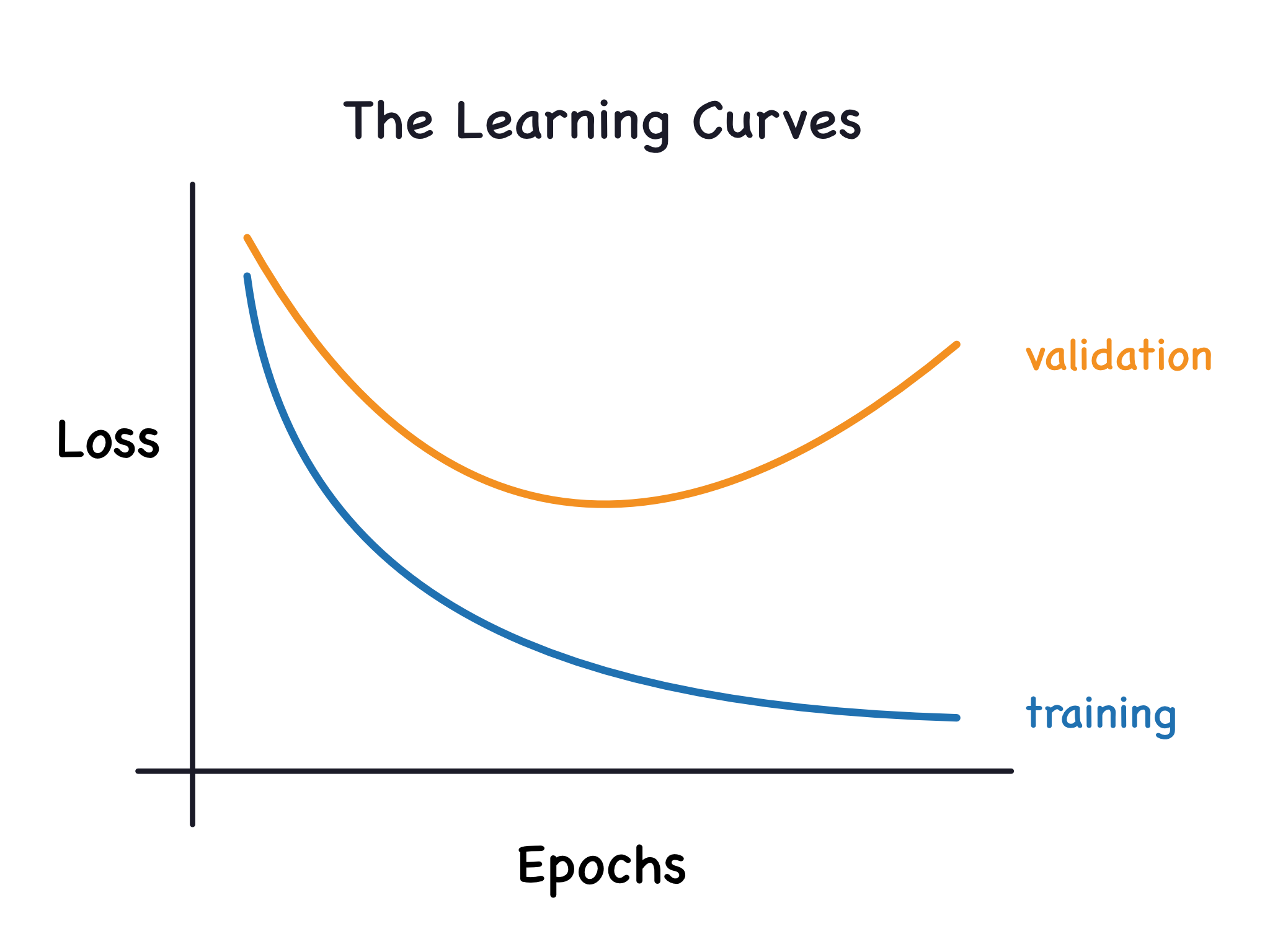
Then we developed a base model first with some randomly taken hyperparameter values. The idea is to test a list of values for each hyperparameter, one at a time fixing all others. Techniques such RandomizedSearchCV and GridSearchCV are not used because of their long compilation time. Instead, each value in the list of hyperparameter’s values is tested and evaluated one at a time.

**Evaluation**

The evaluation was done through a loss graph and metrics having mean squared error(mse), mean absolute error (mae), and explained variance score.

The better the model is in terms of accuracy, lower are its mse and mae values and higher is its explained variance score (max 1.0).

Overfitting was analyzed through the loss graph which shows the validation loss and training loss. The validation loss first decreases and after a certain epoch it starts increasing. The graph looks like this:

Image taken from Kaggle.com

Following is the list of hyperparameter values tested:

1. Activation Function: [relu, tanh, sigmoid]
2. Optimizer: [Adam, RMSprop, Adagrad]
3. Loss Function: [mse, mae] or [MeanSquaredError, MeanAbsoluteError]
4. Learning Rate: [0.001, 0.01, 0.1, 1]
5. Number of Hidden Layers: [1, 2]
6. Number of neutrons in hidden layer(s): [5, 10, 15, 20]
7. Batch Size: [8, 16, 32, 64, 128]
8. Epochs: [50, 100, 300, 500]

**The Base Model**:

Activation Function: ReLU; Optimizer: Adam; Loss Function: Mean Squared Error; Learning Rate: 0.01; Hidden Layer: 1; Neurons: 10; Batch Size: 32; Epochs: 50

Evaluations:

Mean Squared Error (MSE): 88554463131.5149

Mean Absolute Error (MAE): 189412.46115079825

Explained Variance Score: 0.31537178300065405

Then one by one each hyperparameter was tested for the values given above.

Only the given hyperparameter is changed, rest are of base model, i.e., fixed. For loss graph and better insight check the python notebook

1. **Activation Function**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 1.1 | tanh | 413446121564.6414 | 533635.0154004042 | 0.0 |
| 1.2 | sigmoid | 413449517765.6201 | 533638.1975294234 | 6.49e-13 |

The best model was with activation function: ReLU. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Optimizer**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 2.1 | RMSprop | 90248048711.71428 | 190680.6413697362 | 0.30142838204 |
| 2.2 | Adagrad | 0.301428382043267 | 535770.6820653009 | 0.000254633431 |

The best model was with Optimizer: Adam. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Loss Function**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 3.1 | Mean Abs. Error | 95625992720.05597 | 175596.9343583699 | 0.288024617929 |

The best model was with Loss Function: Mean Squared Error. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Learning Rate**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 4.1 | 0.001 | 416222271552.2214 | 536229.8757519667 | 0.0 |
| 4.2 | 0.1 | 44260592289.35190 | 129537.48113635203 | 0.656235176243 |
| 4.3 | 1 | 30370035961.75394 | 107705.1754233427 | 0.764238201734 |
| 4.4 | 2 | 28157634784.00942 | 106613.0305161094 | 0.783580518936 |

The best model was with Learning Rate: 1.0. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Number of hidden layers**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 5.1 | 2 | 29239499292.90591 | 105722.737260401 | 0.773938405138 |

The best model was with the number of hidden layers: 1. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Number of neurons in hidden laye**r:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 6.1 | 5 | 30216137022.7264 | 106165.8860154193 | 0.765221492471 |
| 6.2 | 15 | 28634651255.57621 | 104267.5645624855 | 0.777712121692 |
| 6.3 | 20 | 29008736324.78112 | 107241.5745520285 | 0.776619875038 |

The best model was with the number of neurons hidden layers: 15. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Batch Size**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 7.1 | 8 | 29709251272.07603 | 108188.331714014 | 0.770822080449 |
| 7.2 | 16 | 28587899319.61209 | 103295.69118662512 | 0.778882554567 |
| 7.3 | 64 | 30985545068.32525 | 109152.1656794091 | 0.759762090644 |
| 7.4 | 128 | 31148668098.85217 | 107615.5477905555 | 0.758020837547 |

The best model was with twitch size: 64. This value is fixed from now on. Rest are the same as it is. Check the python notebook for further analysis and comment why this was chosen.

1. **Epochs**:

| Model Name | Value of Hyperparameter | Mean Squared Error (MSE) | Mean Absolute Error (MAE) | Explained Variance Score |
| --- | --- | --- | --- | --- |
| 8.1 | 100 | 28081625873.91670 | 103060.2993849825 | 0.781855326614 |
| 8.2 | 300 | 28096858748.6325 | 104453.9031363879 | 0.782041564578 |
| 8.3 | 500 | 28292243745.78729 | 103919.0353297807 | 0.781485264063 |

The best model came out to be with epochs: 300.

Now our best three models are taken:

Model 4.3, Model 6.2, Model 8.2

They are further tuned a little (check python notebook) and the best among them came out to be model 8.2.1

**Hyperparameters of Model 8.2.1**:

Activation Function: ReLU; Optimizer: Adam; Loss Function: Mean Squared Error; Learning Rate: 1.0; Hidden Layer: 1; Neurons: 10; Batch Size: 64; Epochs: 300

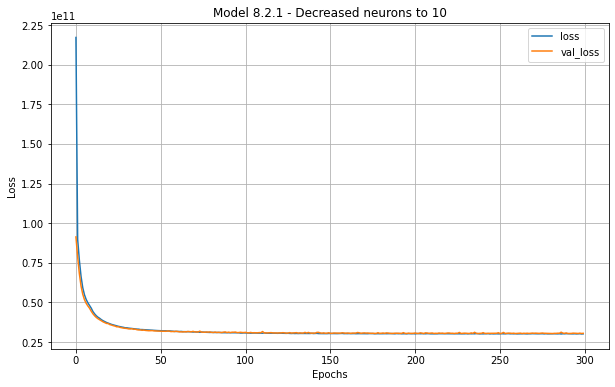
**Evaluation of Model 8.2.1**:

1. Validation Set -

Mean Squared Error (MSE): 30466837598.072002

Mean Absolute Error (MAE): 110807.1171423097

Explained Variance Score: 0.7634834273412523



1. Test Set -

Mean Squared Error (MSE): 25020542964.271538

Mean Absolute Error (MAE): 101698.9003454556

Explained Variance Score: 0.8271148747055983

Since the model is performing well on the test set, we feel it is the best model.

For any loss graph or detailed thought process, refer to the python notebook.

**Conclusion:**

The hyperparameter tuning is a crucial process in training a machine/deep learning model and it can also give us insights about the behavior of the algorithms and data.

Here we tuned hyperparameters by taking a set of testing values of each parameter and training the model by changing one parameter at a time fixing the rest. This process led us to achieve the best possible model for the given dataset.

**FINAL CONCLUSIONS**

* One of the challenges we faced was managing the schedule and meetings because we had different modules this semester and that's why we had to rely on online meetings which were a little less productive.
* It took us time to choose different parameters because we tried the hit and trial method.
* While developing the program we had to give a lot of time for understanding the data and cleaning the dataset. The dataset is of King County housing sales so we had research online about the housing market to understand the data. .
* Another challenge we faced was we did not have a higher computing power so it was taking longer to train the models.
* For future developments the program can be made with random search and grid search provided with high computational power

Tasks:

| Loading Dataset | Amol Rathod |
| --- | --- |
| Visualizing and Understanding dataset | Amol Rathod |
| Cleaning Dataset | Kshitij Singh |
| Splitting the Dataset | Amol Rathod |
| Hyperparameter Tuning | Kshitij Singh |
| Evaluation Metrics | Kshitij Singh, Amol Rathod |
| Model Selection | Kshitij Singh, Amol Rathod |
| Report | Kshitij Singh, Amol Rathod |