Boston House Prices

Data Preprocessing

- Data Loading
    - The dataset is downloaded from the UCI Machine Learning Repository and loaded into a Pandas DataFrame.
    - Proper column names are assigned to the DataFrame for better readability and accessibility.
- Exploratory Data Analysis (EDA)
    - Basic EDA is performed to get a sense of the data, including displaying the first few rows, checking the shape of the dataset, and generating descriptive statistics.
    - This helps in understanding the distribution of the data, identifying any anomalies, and deciding on the next steps for preprocessing.
- Handling Missing Values
    - A check for missing values is performed to ensure data integrity.
    - If missing values were present, appropriate steps would need to be taken, such as imputing missing values or dropping rows/columns.
- Visualization
    - Boxplots are created for each feature to identify outliers. Outliers can significantly affect the performance of machine learning models, especially neural networks, and addressing them is crucial.
    - Scatter plots with regression lines are generated for each feature against the target variable to visualize linear relationships and understand how each feature contributes to the target variable.
- Feature Scaling
    - The features are normalized using MinMaxScaler, which scales the data to a range of [0, 1].
    - Neural networks are sensitive to the scale of input data, and having features on the same scale helps in speeding up the training process and achieving better performance.

- Data Splitting
    - The data is split into features (X) and the target variable (y).
    - It is further divided into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the test set is used to evaluate the model's performance on unseen data.
    - The data is converted to PyTorch tensors to facilitate the training process with a neural network.

Model Architecture

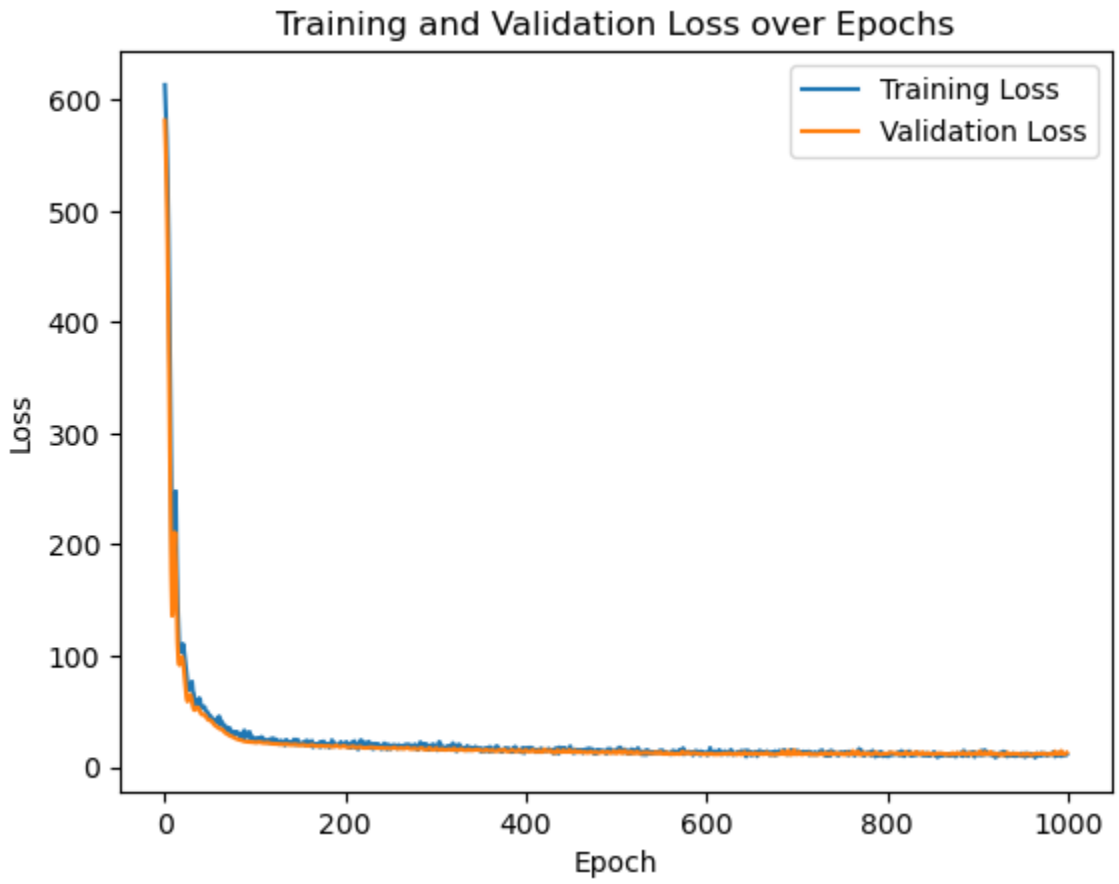
- Neural Network
    - The model is defined as a class in PyTorch, inheriting from nn.Module.
    - It consists of an input layer, two hidden layers, and an output layer.
- Input Layer
    - The size of the input layer is determined by the number of features in the dataset.
    - The data is passed to the first hidden layer after going through the input layer.
- Hidden Layers
    - Each hidden layer is fully connected, meaning each neuron in a layer is connected to every neuron in the next layer.
    - The activation function used in the hidden layers is ReLU (Rectified Linear Unit). ReLU is a popular choice due to its simplicity and the ability to mitigate the vanishing gradient problem.
    - Dropout is applied after the activation function, serving as a form of regularization to prevent overfitting. Dropout works by randomly setting a fraction of input units to 0 at each update during training, which helps to make the model more robust.
- Output Layer
    - The output layer has a single neuron since this is a regression task (predicting a continuous value).
    - No activation function is applied at the output layer for regression tasks.

Choice of Hyperparameters

- Learning Rate
  - Value: 0.01
  - Significance: The learning rate controls the size of the steps that the optimizer takes while updating the weights. A learning rate that is too high can cause the model to converge too quickly and potentially overshoot the minimum cost. A learning rate that is too low can cause the model to learn very slowly, or not converge at all.
  - Choice: A value of 0.01 is a common starting point, but it might require tuning based on the specific problem and dataset.
- Optimizer
  - Type: Adam
  - Significance: The optimizer is responsible for updating the weights of the network to minimize the loss. Adam is a popular choice due to its combination of the benefits of two other extensions of stochastic gradient descent: AdaGrad and RMSProp.
  - Choice: Adam is known for being effective and efficient, making it a reasonable default choice.
- Loss Function
  - Type: Mean Squared Error (MSE)
  - Significance: The loss function measures the difference between the model's predictions and the actual target values. MSE is a common loss function for regression problems.
  - Choice: MSE is appropriate for this task as it is a regression problem, aiming to minimize the average squared difference between predicted and actual values.
- Epochs
  - Value: 1000
  - Significance: An epoch is one complete pass through the entire training dataset. The number of epochs is the number of times the learning algorithm will work through the entire training dataset.
  - Choice: Training for a large number of epochs can capture more information from the data, but it also risks overfitting. The chosen value seems relatively high, and monitoring for overfitting would be essential.

- Dropout Rate
    - Value: Not specified in the visible part of the notebook
    - Significance: Dropout is a regularization technique where randomly selected neurons are ignored during training. It helps in preventing overfitting.
    - Choice: The dropout rate is a hyperparameter that can be tuned. Common values are between 0.2 and 0.5.
- Number of Hidden Layers and Neurons
    - Significance: The number of hidden layers and the number of neurons in each layer define the capacity of the network, affecting its ability to model complex relationships.
    - Choice: The notebook specifies a network with two hidden layers. The number of neurons in each layer would significantly affect the model's performance and should be tuned based on the task.

Learning Curves



- X-Axis (Epochs): Represents the number of passes through the entire training dataset. As the number of epochs increases, the model has more opportunities to learn from the data.
- Y-Axis (Loss): Represents the model's error or loss. Lower values indicate better performance.
- Training Loss: Typically decreases over time as the model learns from the training data.
- Validation Loss: Shows how well the model is performing on unseen data.
- Convergence: Both training and validation loss decrease and stabilize over time.

Evaluation metrics on the test set

- Mean Absolute Error (MAE)
    - Description: The average of the absolute differences between the predicted and actual values.
    - Interpretation: A MAE of 0 indicates perfect predictions. The higher the MAE, the larger the error in predictions.
- Mean Squared Error (MSE)
    - Description: The average of the squares of the differences between the predicted and actual values.
    - Interpretation: Like MAE, a MSE of 0 indicates perfect predictions. MSE penalizes larger errors more heavily than MAE, making it more sensitive to outliers.
- Root Mean Squared Error (RMSE)
    - Description: The square root of the MSE.
    - Interpretation: RMSE is on the same scale as the target variable, making it easier to interpret than MSE. A lower RMSE indicates better performance.
- R-Squared (R2)
    - Description: A statistical measure that represents the proportion of the variance for the dependent variable that's predicted from the independent variables.
    - Interpretation: Ranges from -∞ to 1. A value of 1 indicates perfect predictions, 0 indicates that the model is no better than simply predicting the mean of the target variable for all observations, and negative values indicate a very poor model.

- Mean Absolute Percentage Error (MAPE)
    - Description: The average of the absolute percentage differences between the predicted and actual values.
    - Interpretation: Expressed as a percentage, lower values are better with 0 being a perfect prediction. However, MAPE can be highly skewed if there are values close to 0 in the target variable.

Result

Evaluation metrics on Test data

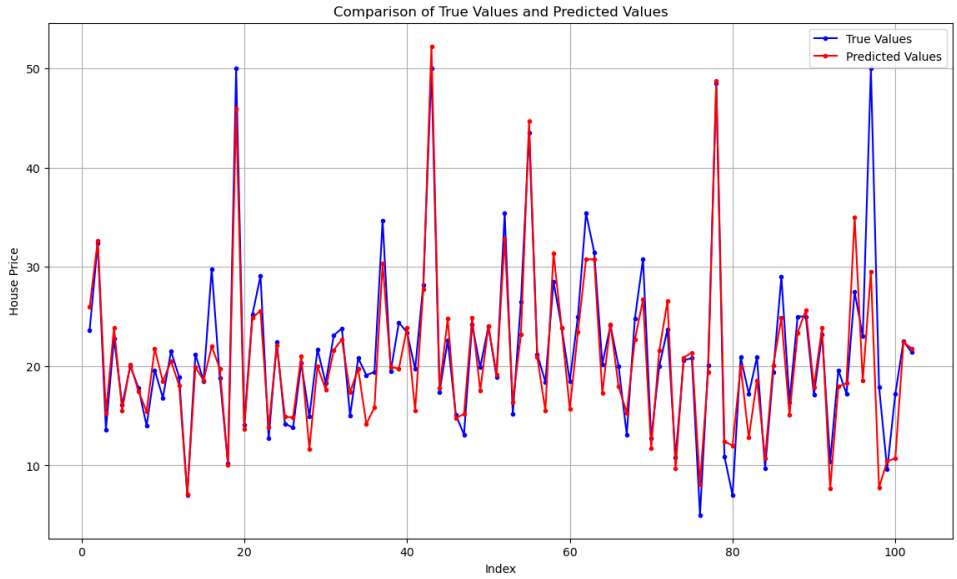
Mean Absolute Error (MAE): 2.070977658851474

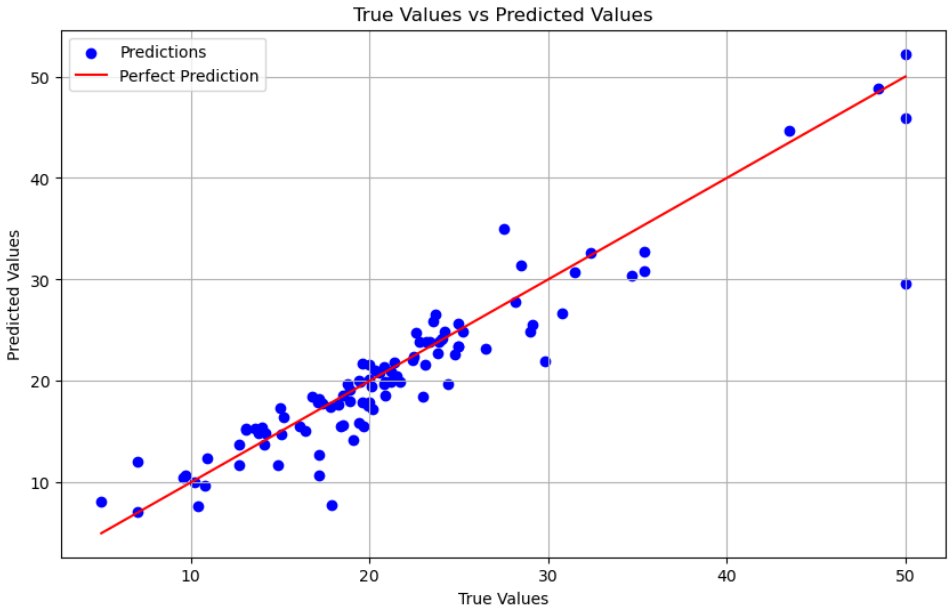
Mean Squared Error (MSE): 10.965661324561026

Root Mean Squared Error (RMSE): 3.3114439938735227

R-Squared (R2): 0.8504691718878351

Mean Absolute Percentage Error (MAPE): 10.433946765483856%





Summary

- Architecture:
  - The architecture is simple and well-suited for a regression task, with sufficient complexity to capture non-linear relationships.
- Hyperparameters:
  - The chosen hyperparameters are standard and provide a good balance, but may require fine-tuning based on the specific characteristics of the dataset.
- Learning Curves:
  - Observing the learning curves is crucial for diagnosing issues in the training process and ensuring that the model is learning effectively.
- Final Thoughts:
  - Continuous monitoring and adjustment of the architecture and hyperparameters are essential for achieving optimal performance.
  - Comparing the model's performance with benchmarks or baseline models can provide additional insights and validation of the results.