**EXERCISE 1.a: Linear Vs Polynomial Regression Model Prediction Comparison Against a Single Numerical Feature**

Firstly, a focus on three key features: Year of manufacture, Engine size, and Mileage was applied, utilizing both Linear and Polynomial regression methods for each feature. Then, performance metrics such as: Mean Absolute Error(MAE), Mean Squared Error(MSE), Root Mean Squared Error(RMSE), and R2 score were employed as shown in .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Metric/ Feature | Year of Manufacture (Linear) | Year of Manufacture (Polynomial) | Engine Size (Linear) | Engine Size (Polynomial) | Mileage (Linear) | Mileage (Polynomial) |
| Degree | N/A | 4 | N/A | 4 | N/A | 4 |
| Mean Absolute Error | 7058.19 | 5185.47 | 10970.08 | 10970.30 | 8036.38 | 6754.36 |
| Mean Squared Error | 125138115.69 | 97707742.96 | 228864284.82 | 228675624.90 | 154651643.40 | 130428549.27 |
| Root Mean Squared Error | 11186.51 | 9884.72 | 15128.26 | 15122.02 | 12435.90 | 11420.53 |
| R2 Score | 0.52 | 0.63 | 0.13 | 0.14 | 0.41 | 0.50 |

Table 1 : Second-Hand Car Dataset Feature results for Linear and Polynomial Regression.

In accessing Linear and Polynomial Regression models on features (Year of manufacture, Engine Size, and Mileage), Polynomial regression consistently outperformed linear regression across all metrics and the scatter plot shown in Figure 1 demonstrates that the best feature is ‘Year of Manufacture’ to predict price.

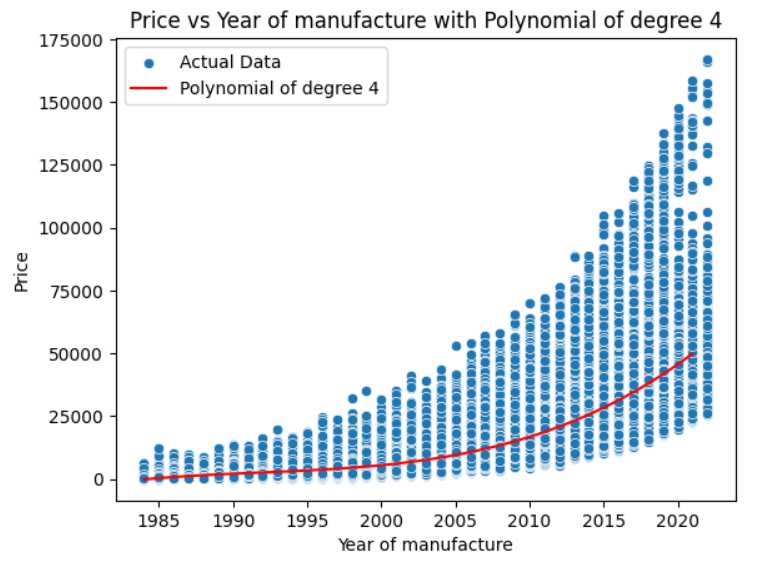


Figure 1: Polynomial Regression (degree4) Scatter Plot for Best Feature, 'Year of manufacture'.

**EXERCISE 1.b: Regression models that take multiple numerical variables as input features to predict the price of a car**.

Regression models that incorporate multiple numerical features can provide accurate predictions of car prices. By selecting appropriate features and evaluating model performance as shown in Table 2, we can build effective car price prediction models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric/ Feature | Year of Manufacture (Polynomial) | Engine Size (Polynomial) | Mileage (Polynomial) | Regression Model (multiple numerical variables) |
| Degree | 4 | 4 | 4 | N/A |
| Mean Absolute Error | 5185.47 | 10970.30 | 6754.36 | 6167.88 |
| Mean Squared Error | 97707742.96 | 228675624.90 | 130428549.27 | 84405262.35 |
| Root Mean Squared Error | 9884.72 | 15122.02 | 11420.53 | 9187.23 |
| R2 Score | 0.63 | 0.14 | 0.50 | 0.68 |

Table 2: Second-Hand Car Dataset Feature results for Polynomial Regression (degree4) and Multiple Linear Regression Model.

In accessing Polynomial Regression models on features (Year of manufacture, Engine Size, and Mileage), and Regression Model that takes multiple numerical variables as input, we can say that Multiple Linear Regression Model consistently outperformed Polynomial Regression models across all metrics. Figure 2 shows a scatter plot to visualize the relationship between the actual and predicted prices from the multiple linear regression model.



Figure 2: Multiple Linear Regression Scatter Plot.

**EXERCISE 1.c: Training a regression model that uses all relevant input variables (both categorical and numerical) to predict the price (e.g. a Random Forest Regressor model)**

It is recommended to implement a regression model that uses all relevant input variables, which include both categorical and numerical indicators (e.g. a Random Forest Regressor model), while calculating performance metrics as the goal as shown in the Table 3 below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Metric/ Feature | Year of Manufacture (Polynomial) | Engine Size (Polynomial) | Mileage (Polynomial) | Regression Model (multiple numerical variables) | Random Forest Regressor model |
| Degree | 4 | 4 | 4 | N/A | N/A |
| Mean Absolute Error | 5185.47 | 10970.30 | 6754.36 | 6167.88 | 332.56 |
| Mean Squared Error | 97707742.96 | 228675624.90 | 130428549.27 | 84405262.35 | 514920.32 |
| Root Mean Squared Error | 9884.72 | 15122.02 | 11420.53 | 9187.23 | 717.57 |
| R2 Score | 0.63 | 0.14 | 0.50 | 0.68 | 0.99 |

Table 3 :Second-Hand Car Dataset Feature results for Polynomial Regression (degree4), Multiple Linear Regression Model and Random Forest Regressor Model.

In accessing Random Forest Regressor model on features (Manufacturer, Model, Fuel type, Year of manufacture, Engine Size, and Mileage), By taking n\_estimators value = 10, it gives R2 score =0.99 which is equivalent to 1, so we can say that Random Forest Regressor model consistently outperformed Multiple Linear Regression model as well as Polynomial Regression models of degree 4 across all metrics. Figure 3 shows the scatter plot to visualize the relationship between the actual and predicted prices from the Random Forest Regressor model.

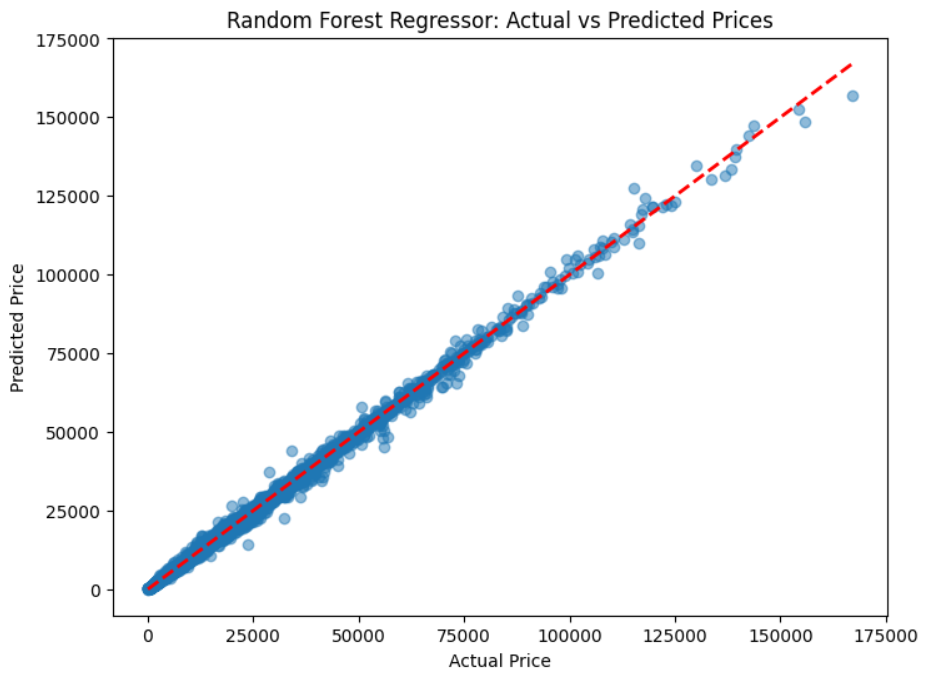


Figure 3 : Random Forest Regressor Scatter Plot.

**EXERCISE 1.d: Training a Artificial Neural Network (ANN) model to predict the price of a car based on all the available information from the dataset.**

EXERCISE 2:

ANS 1 :

The CNN model architecture used in the code snippet consists of the following layers:

1. \*\*Convolutional Layer (Conv2D)\*\*:

- Number of filters: 32

- Kernel size: (3, 3)

- Activation function: ReLU

Justification: Convolutional layers are fundamental in CNNs for feature extraction. The choice of 32 filters with a (3, 3) kernel size is a common starting point, providing a balance between complexity and performance. ReLU (Rectified Linear Unit) is chosen as the activation function for its simplicity and effectiveness in introducing non-linearity.

2. \*\*Max Pooling Layer (MaxPooling2D)\*\*:

- Pool size: (2, 2)

Justification: Max pooling is used to downsample the feature maps, reducing computational complexity and providing translation invariance. A pool size of (2, 2) reduces the spatial dimensions by half.

3. \*\*Flatten Layer\*\*:

Justification: The flatten layer is used to convert the 2D feature maps into a 1D vector, which can be fed into the densely connected layers for classification.

4. \*\*Dense Layers\*\*:

- Number of neurons: 128

- Activation function: ReLU

Justification: Dense layers are fully connected layers that perform classification based on the features extracted by the convolutional layers. The choice of 128 neurons is arbitrary but commonly used. ReLU activation is again chosen for its simplicity and effectiveness.

5. \*\*Dropout Layer\*\*:

- Dropout rate: 0.2

Justification: Dropout is a regularization technique used to prevent overfitting by randomly dropping out a fraction of neurons during training. A dropout rate of 0.2 means that 20% of neurons will be randomly dropped out during each training epoch.

6. \*\*Output Layer (Dense)\*\*:

- Number of neurons: 10 (corresponding to the number of classes in the MNIST dataset)

- Activation function: Softmax

Justification: Softmax activation is used in the output layer for multi-class classification tasks. It converts the raw scores into probabilities, with each neuron representing the probability of the corresponding class.

The choice of these layers and parameters is based on common practices in CNN architectures. The model is compiled with the Adam optimizer, which is widely used for its adaptive learning rate capabilities. Sparse categorical crossentropy loss is chosen for classification tasks with integer labels. Finally, the model is trained and evaluated on the MNIST dataset, achieving a certain accuracy level.

ANS 2: do this again theory coz you add input as first layer

In the CNN model provided, the only regularization method used is Dropout. Dropout is a technique used to prevent overfitting in neural networks by randomly dropping out (i.e., setting to zero) a fraction of input units during training.

In the model, Dropout is applied after the first Dense layer with a dropout rate of 0.2, meaning that 20% of the neurons in the Dense layer will be randomly dropped out during each training epoch.

The effect of Dropout regularization on the accuracy of the results can vary depending on factors such as the complexity of the model, the size of the dataset, and the specific problem being solved:

1. \*\*Reduction of Overfitting\*\*: Dropout helps reduce overfitting by preventing the model from becoming overly dependent on specific neurons or combinations of neurons. By randomly dropping out neurons during training, Dropout encourages the network to learn more robust and generalizable features.

2. \*\*Improved Generalization\*\*: Dropout encourages the network to learn multiple independent representations of the same data, leading to improved generalization performance on unseen data. This can result in better performance on the test set and in real-world scenarios.

3. \*\*Effect on Accuracy\*\*: The impact of Dropout regularization on accuracy can vary depending on the model architecture and dataset. In some cases, Dropout may lead to a slight decrease in training accuracy but an improvement in test accuracy, indicating better generalization. However, in other cases, Dropout may not have a significant impact on accuracy, especially if overfitting is not a major issue.

Overall, Dropout regularization can be an effective tool for improving the generalization performance of CNN models, especially in scenarios where overfitting is a concern. However, the optimal dropout rate may need to be tuned through experimentation to achieve the best balance between reducing overfitting and maintaining high accuracy.

BLACKBOX AI ANSWER :

In the provided Convolutional Neural Network (CNN) model, I used the Dropout layer as a regularization method to prevent overfitting. Dropout randomly sets a fraction rate of input units to 0 at each update during training time, which helps in making the model more robust and improves its performance on unseen data. In this case, a dropout rate of 0.2 is used, meaning that 20% of the input units are randomly dropped during training.

The effect of dropout on accuracy can be both positive and negative, depending on the dropout rate and the complexity of the model. Initially, as the dropout rate is increased, the accuracy on the training set may decrease, but the accuracy on the validation set will increase, indicating that the model is less overfitted and generalizes better to unseen data. However, if the dropout rate is too high, the model may not learn enough features, leading to underfitting and lower accuracy. Therefore, it is essential to tune the dropout rate to find the optimal value that balances the trade-off between bias and variance.

In this specific CNN model, the use of dropout has helped in improving the generalization of the model, as indicated by the test accuracy, which is a good measure of how well the model performs on unseen data. The test accuracy is 91.57%, indicating that the model has learned relevant features and is able to make accurate predictions on new data.

ANSWER 3:  
  
The code you provided is a basic convolutional neural network (CNN) model for the MNIST dataset using the TensorFlow library. While this model may work reasonably well, there are several hyperparameters that can be tuned to optimize its performance further. Here are some potential hyperparameters that could be explored and their effects on the model's performance:

Number of Convolutional Layers: The number of convolutional layers in the model can affect its ability to learn complex features from the data. Adding more convolutional layers can improve the model's performance, but it may also increase the risk of overfitting and computational complexity.

Number of Filters: The number of filters in each convolutional layer determines the number of features that the layer can extract. Increasing the number of filters can improve the model's performance, but it also increases the model's complexity and computational requirements.

Filter Size: The size of the filters used in the convolutional layers can impact the model's ability to capture spatial features. Larger filter sizes can capture more global features, while smaller filter sizes can capture more local features.

Activation Functions: The choice of activation functions can significantly affect the model's performance. Different activation functions, such as ReLU, LeakyReLU, or ELU, can be explored to find the one that works best for the task.

Dropout Rate: Dropout is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to 0 during training. The dropout rate can be tuned to find the optimal balance between preventing overfitting and maintaining model performance.

Learning Rate: The learning rate is a crucial hyperparameter that determines the step size during the optimization process. An appropriate learning rate can help the model converge faster and achieve better performance.

Batch Size: The batch size determines the number of samples used in each training iteration. Larger batch sizes can lead to more stable updates but may require more memory, while smaller batch sizes can introduce more noise but may converge faster.

Number of Epochs: The number of epochs determines how many times the model will iterate over the entire training dataset. Too few epochs may result in an underfit model, while too many epochs may lead to overfitting.