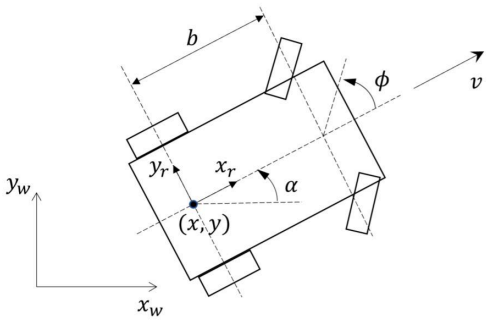
Name: Onkar P. Kher
UID: 120407062
Directory ID: okher

Final Project Report

1. Introduction

The goal of this project was to develop models capable of predicting two key outputs—translational velocity (v) and rotational velocity (w)—of a car, based on input features derived from sensor data. Three machine learning models were implemented and evaluated: Neural Networks, Linear Regression, and Decision Tree Regression.

Car Like Robot



$$\dot{x} = v\cos\alpha$$
$$\dot{y} = v\sin\alpha$$
$$\dot{\alpha} = \frac{v}{b}\tan\phi.$$

The control inputs to this model is $u = [v, \omega]$, where $\omega \triangleq \dot{\alpha}$ is the rotational speed of the robot. These can be transformed to the command actions (v, ϕ), where v is the translational speed and ϕ is the steering angle obtained by $\phi = \tan^{-}1\,(\omega b/v)$.

Each model followed a structured machine learning pipeline, including data preprocessing, model implementation, hyperparameter tuning, training, and evaluation on a test dataset. The pipeline ensured consistency across models and enabled a fair comparison of their performances.

2. Data Preprocessing

Proper data preprocessing is essential for ensuring the models converge effectively and perform optimally. The training dataset was split into two parts: training (80%) and validation (20%) sets for training and hyperparameter tuning respectively. Features were normalized for models like Neural Networks and Linear Regression to improve numerical stability during training.

For all models, a concatenation of three training and validation sets were used for final training, while a concatenation of two test sets were reserved for evaluation to ensure an unbiased assessment of generalization performance. This was done due to memory and computation constraints of the working environment. The following steps were carried out on all the datasets before using them for training, validation and evaluation on any models.

2.1. Data Loading

The preprocessed datasets were loaded from Google Drive, consisting of three main components:

- Feature Matrices: Input features representing sensor data.
- Target Variables: Corresponding translational velocity (v) and rotational velocity (w).

The data was divided was available as the following datasets:

- Training Set: Used for learning the relationships between the input features and outputs.
- Validation Set: Used for hyperparameter tuning and monitoring performance to ensure that the model generalizes well to unseen data. Split from the training set.
- Test Set: Reserved for evaluating the final model's performance.

2.2. Data Splitting and Shuffling

Before splitting the data, it was shuffled to ensure that the training, validation, and test sets are representative of the overall data distribution. Shuffling the data prevents any potential bias caused by sequential patterns in the dataset, such as sensor recordings collected in time order or grouped data. By shuffling the dataset, each subset (training, validation, and test) contained a random mix of data points, which helped improve the model's robustness and generalization.

The training dataset was then split into two parts:

1. Training Set (80%): Used to train the model.
2. Validation Set (20% of training): A portion of the training data was separated for hyperparameter tuning and to monitor model performance.

2.3. Feature Scaling and Saving

Scaling the features was an essential step, particularly for models like Neural Networks and Linear Regression, which are sensitive to the magnitude of input values. Standardization (Z-score normalization) was applied, where each feature was transformed to have:

- A mean of 0 and
- A standard deviation of 1.

Standardization ensures that all features contribute equally to the learning process and prevents features with larger magnitudes from dominating the model's optimization process. It also improves the convergence speed and numerical stability of the training process.

For consistency, the scaling parameters (mean and standard deviation) were computed from the training set and then applied to the validation and test sets. This approach avoids information from the test set being incorporated into the training process. After completing the preprocessing steps, the scaled features and corresponding target variables were saved into separate files. This ensured that the preprocessed data could be reloaded quickly in subsequent steps without re-running the preprocessing pipeline.

The saved datasets included:

- Training, validation, and test feature matrices (scaled).
- Corresponding target variables for translational and rotational velocities.

3. Learning Pipeline

After the data preprocessing step, the machine learning pipeline was followed in the following steps:

3.1. Unknown Target Function f

In the learning setup, the unknown target function f represents the true relationship between the input features and the output labels. In our case, the target function describes how the input

Name: Onkar P. Kher
UID: 120407062
Directory ID: okher
features X map to the output velocities y = (v, w) , where v represents translational velocity and w represents rotational velocity.

- Goal: Approximate this unknown function f using machine learning models (Neural Networks, Linear Regression, and Decision Tree Regression).
- Context: Since the true function f is unknown, our models aim to learn it using training examples.

3.2. Training Examples

The training examples consist of labeled data points (x_n, y_n) , where:

- x_n : Input feature vectors representing sensor readings or other relevant information.
- y_n : Corresponding true labels, v (translational velocity) and w (rotational velocity).

Approach:

- The training dataset was split into training and validation data to ensure effective model training and evaluation.
- Models were trained on the training data and validated on the validation set to tune hyperparameters.
- Once hyperparameters were optimized, the final model was trained on the combined training and validation sets and evaluated on the test dataset.

Purpose:

- The training examples serve as input for the learning algorithm to iteratively refine the models to approximate the unknown target function.

3.3. Hypothesis Set $\mathcal{H}$

The hypothesis set $\mathcal{H}$ represents the set of all functions that a model can generate to approximate the target function f. Each model has its own hypothesis set:

3.3.1. Linear Regression

Linear Regression served as a baseline model for predicting v and w. This model assumes a linear relationship between the input features and the target outputs, making it simple to train and interpret. The model was trained on the combined training and validation sets. Predictions were made on the test set, and metrics such as MSE, MAE, and R^2 were computed for both v and w. Scatter plots of actual vs. predicted values were created to visualize the model's performance. Although Linear Regression provided a reasonable starting point, it lacked the capacity to model the complex, non-linear dependencies in the data, resulting in higher errors compared to the other models.

The hypothesis set $\mathcal{H}$ consists of all possible linear mappings between input features X and outputs y. Each individual hypothesis is defined by a linear equation $y = Xw + b$.

3.3.2. Decision Tree Regression

Decision Tree Regression was used as a non-linear model to capture more complex relationships between the input features and target outputs. A maximum depth of 10 was set to control overfitting while maintaining interpretability. Similar to Linear Regression, the Decision Tree model was trained on the combined training and validation sets. Test set predictions were evaluated using MSE, MAE, and R^2 metrics. Scatter plots were

also generated to compare actual vs. predicted values for v and w. The Decision Tree model performed better than Linear Regression and was computational less expensive than the Neural Network model. However, the results were inaccurate as compared to the Neural Network model, as indicated by the gap between training and test errors.

The hypothesis set $\mathcal{H}$ comprises all possible decision tree with various splits and depths. Each tree represents a piecewise constant function that maps inputs X to outputs y.

3.3.3. Neural Networks

Neural Networks were used to capture complex, non-linear relationships between input features and the target outputs (v and w). The network architecture consisted of:

- Input layer: Matching the number of features in the input data.
- Three hidden layers: Containing 128, 64, and 32 neurons, respectively, all with ReLU activation.
- Output layer: Two neurons (one each for v and w) without activation for regression tasks.

The Adam optimizer was used to train the model, and the mean squared error (MSE) was chosen as the loss function. Hyperparameter tuning was performed to optimize the batch size and learning rate. The performance of the neural network was evaluated using MSE and mean absolute error (MAE). Additionally, learning curves were plotted to observe the model's behavior on training and validation sets as the dataset size varied. The Vapnik-Chervonenkis (VC) generalization bound has been incorporated in the implementation of the neural network model to estimate the model's generalization error. Mathematically, $E_{out} \leq E_{in} + confidence\ term$, where the confidence term is calculated as $= \sqrt{\frac{h.ln\left(\frac{2N}{h}\right) + ln(\frac{4}{\delta})}{N}}$ which grows with model complexity and decreases with more training samples. This V.C. bound provides a theoretical upper bound on the true generalization error based on the model complexity and the training sample size.

The hypothesis set $\mathcal{H}$ consists of all possible decision boundaries (non-linear) that the neural network can generate by adjusting its weights across layers. Each configuration of weights defines a unique hypothesis.

3.4. Learning Algorithm *A*

The learning algorithm A refers to the method used to train each model. The goal of the algorithm is to search the hypothesis set $\mathcal{H}$ for the best hypothesis g that minimizes the error between the predictions and the true outputs.

3.4.1. Linear Regression

The weights *w* are determined using the least squares method to minimize the MSE. The closed-form solution or optimization algorithms like gradient descent are used.

Name: Onkar P. Kher
UID: 120407062
Directory ID: okher

Linear Regression Model Performance on Test Data:

	MSE	**MAE**	**R^2**
Translational Velocity (*v*)	0.1291	0.2906	0.0618
Rotational Velocity (*w*)	0.0705	0.1790	-0.0275

3.4.2. Decision Tree Regression

The learning algorithm splits the data into regions by minimizing the variance in the output The closed-form solution or optimization algorithms like gradient descent are used.

Decision Tree Regression Performance on Test Data:

	MSE	**MAE**	**R^2**
Translational Velocity (*v*)	0.1465	0.2606	-0.0647
Rotational Velocity (*w*)	0.1051	0.2092	-0.5323

3.4.3. Neural Network

The learning algorithm involves backpropagation and gradient descent (using the Adam optimizer). Weights are iteratively updated based on the Mean Squared Error (MSE) loss function until convergence or early stopping.

Neural Network performance on unseen/ Test data:

Validation Error (In-sample error): 0.019

VC Generalization Bound (E_out): 0.2193903191433006 (Upper bound on E_out)

Test Loss (E_test): 0.14580602943897247

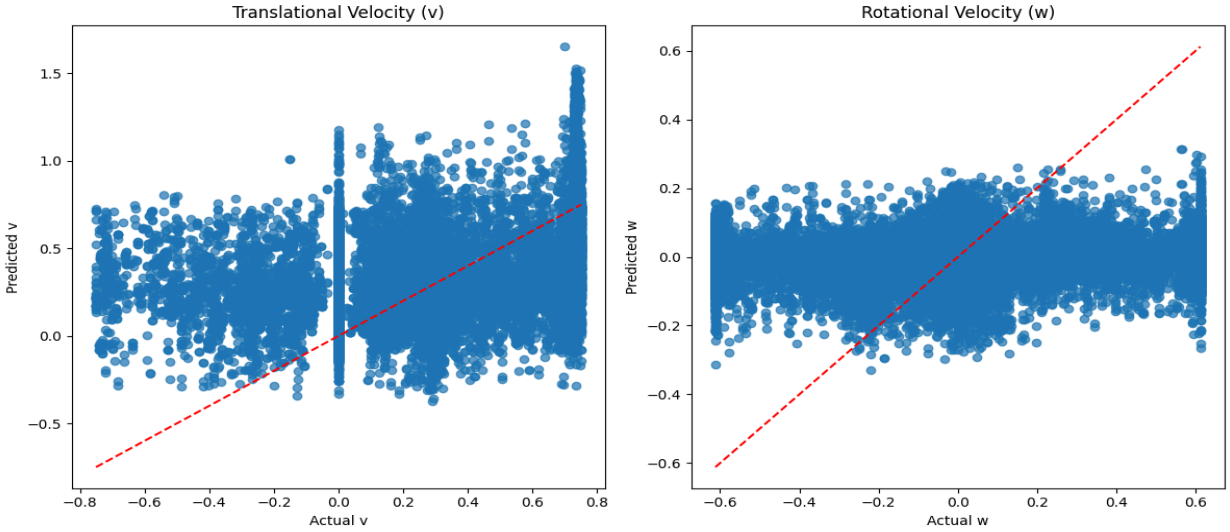
3.5. Final Hypothesis *g*

The final hypothesis g is the learned approximation of the target function f obtained after training the model. It is used to make predictions on new, unseen data. The final hypothesis is defined by the learned weights of the neural network after training. It models a non-linear decision boundary that best fits the training data. From the results we can summarize that the best performing model is the neural network model, generalizing the non-linear behavior of the data well. After the tuning of the hyperparameters on the validation set, we can say that the neural network performs best out of the three models compared, generalizing the non-linear behavior of the dataset to some extent.

4. Results

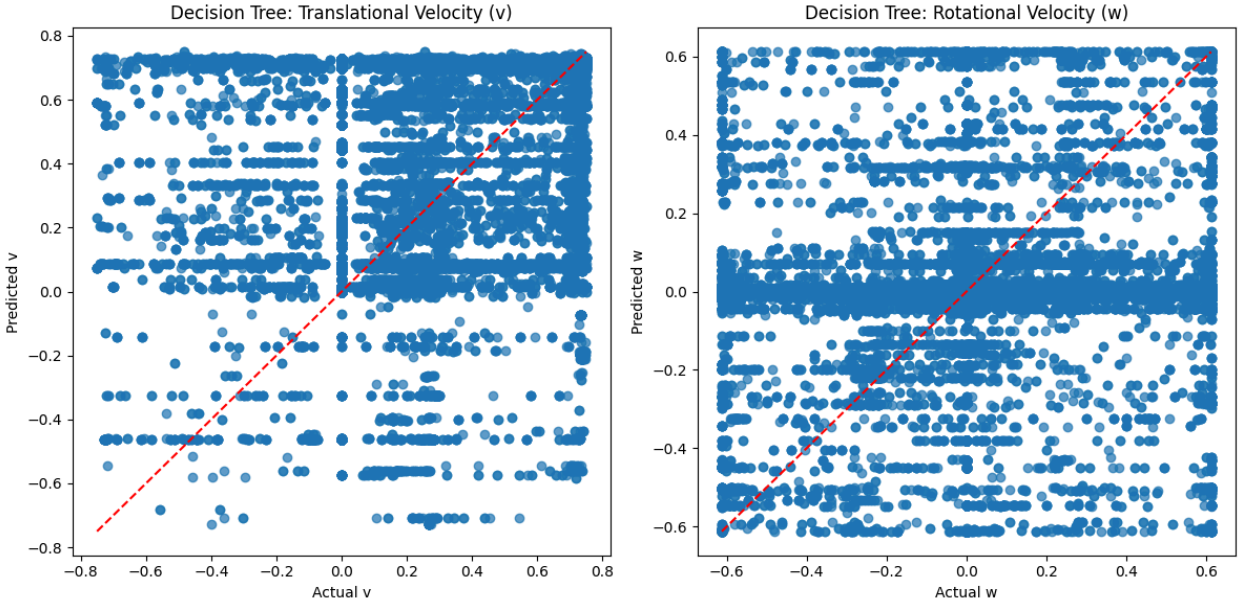
4.1. Linear Regression

Name: Onkar P. Kher
UID: 120407062
Directory ID: okher



Example of a high bias in the linear regression model where the red line indicates the predicted values and the scatter plot has the actual (true) values of the outputs. Graph of actual velocities v/s the predicted velocities.

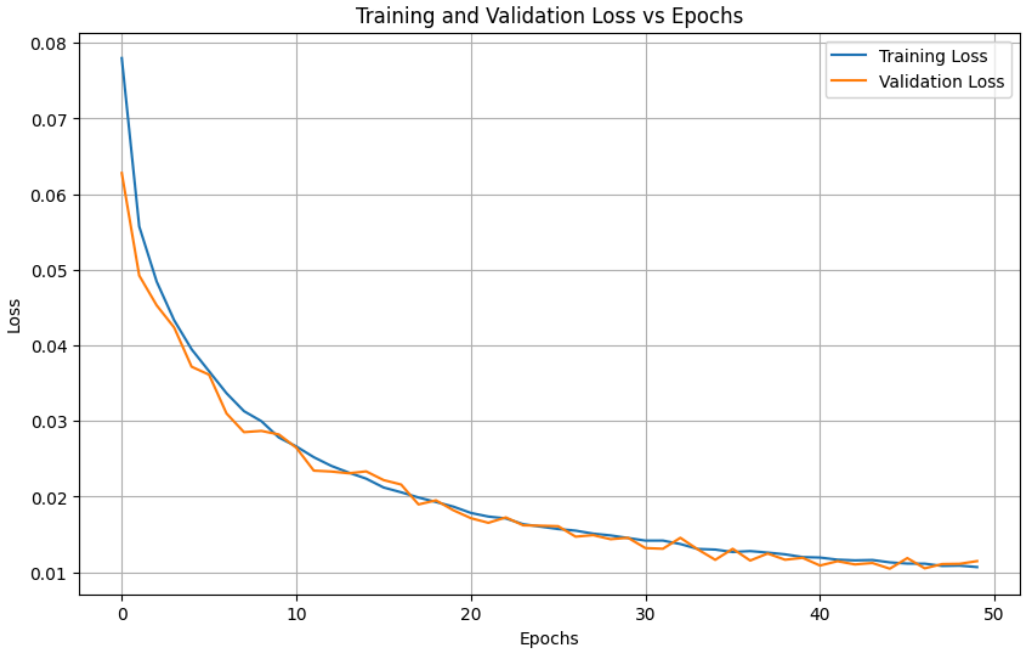
4.2. Decision Tree



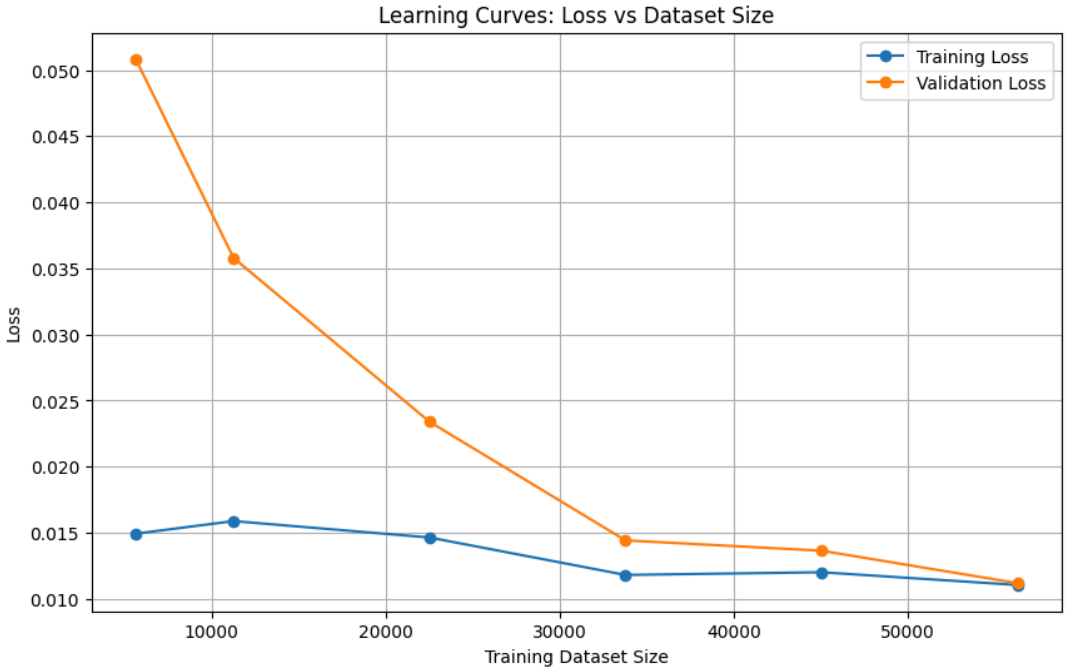
Example of poor generalization by the decision tree regression model, where the red lines indicate the predicted values and the scatter plot indicates the actual (true) values of the outputs. Graph of the actual v/s the predicted velocities

4.3. Neural Network

Name: Onkar P. Kher
UID: 120407062
Directory ID: okher



The plot of the training and validation loss during the final training of the model based on the tuned hyperparameters (namely batch size and learning rate)



The learning curves plotted by varying the dataset size in steps, the slight increase in the training loss indicates that since more training data is being added the model is trying to generalize better and getting more complex leading to rise in the training loss, also the decrease in the validation loss indicates that the model is performing well on previously unseen data even though the training size is increasing indicating learning of the model.

Name: Onkar P. Kher
UID: 120407062
Directory ID: okher

5. Insights

The goal of this project was to predict translational velocity (v) and rotational velocity (w) of a car based on sensor inputs. Given the nature of the problem and the relationships between input features and outputs, three models were selected for implementation:

5.1. Neural Networks

Neural Networks were chosen as the final model because of their ability to approximate complex, non-linear relationships. With multiple layers and non-linear activations, they are well-suited for learning patterns in high-dimensional data.

The flexibility of a feedforward architecture allows the model to learn intricate mappings from input features to outputs (v, w).

5.2. Linear Regression

Linear Regression served as a baseline model to benchmark performance. It assumes a linear relationship between inputs and outputs, which makes it computationally efficient and interpretable. While it cannot capture the non-linear dependencies, it provided insights into the limitations of simpler models. Though the MSE, on unseen (evaluation data) was better than the neural network model we can visualize the prediction results, of the linear model and confidently not select this as our final hypothesis, as the plots indicate towards a high bias, which is confirmed by the R^2 error. Thus, this model is not selected as our final hypothesis.

5.3. Decision Tree Regression

Decision Trees were compared because of their ability to model non-linear relationships and provide interpretability. They can split data into regions and predict piecewise constant outputs, making them suitable for regression tasks.

The diversity of these models allowed a comprehensive evaluation of the problem, balancing complexity, performance, and computational cost. The learning pipeline was followed as given in the **Section 3** of this report, utilizing the various attributes of the models.

5.4. Steps Taken in the Learning Process

The following steps were systematically followed for model selection and evaluation:

5.4.1. Data Preprocessing

Features were standardized (mean = 0, standard deviation = 1) for models like Neural Networks and Linear Regression to ensure numerical stability. The dataset was shuffled before splitting to ensure randomness and eliminate any sequential bias.

- The training data was split into:
  - Training Set (used to train the models), and
  - Validation Set (used for hyperparameter tuning)

5.4.2. Model Training

- Neural Networks were trained using the Adam optimizer with Mean Squared Error (MSE) as the loss function.
- Linear Regression used the least squares optimization method to minimize MSE.

- Decision Tree Regression iteratively split data to minimize variance in the target outputs.

5.4.3. Hyperparameter Tuning

Neural Networks: Batch size and learning rate were tuned using grid search.

- Learning rates of 0.001, 0.01, 0.0005 were evaluated.
- Batch sizes of 16, 32, 64 were tested.

5.4.4. Validation Set Usage

The validation set played a critical role in hyperparameter selection and model monitoring. Models were evaluated on the validation set to identify overfitting or underfitting. Hyperparameters that minimized validation loss were chosen for final training.

5.4.5. Final Model Training

Once optimal hyperparameters were identified, models were retrained on the combined training and validation sets to maximize learning from available data. The test set was reserved for final evaluation.

5.4.6. Evaluation Metrics

All models were evaluated on the test set using:

- Mean Squared Error (MSE),
- Mean Absolute Error (MAE), and
- R^2 Score (to measure explained variance).

5.5. Regularization

Regularization, such as L2 regularization or dropout, is typically used to prevent overfitting by penalizing model complexity. In this project:

- Regularization techniques were not implemented in the final workflow due to their computational expense.
- Preliminary results indicated that including regularization led to jagged learning curves and unstable results. Given the dataset size, overfitting was mitigated by:
  - Early stopping (monitoring validation loss to halt training when overfitting was detected),
  - Careful selection of hyperparameters such as batch size and learning rate,
  - Validation-based performance monitoring.

While regularization techniques could have further controlled model complexity, the above methods provided a practical balance between computational cost and performance.

5.6. Hyperparameter Selection Process

The process for selecting hyperparameters can be summarized as follows:

1. Grid Search: Batch size and learning rate were systematically tested over a predefined range.
2. Validation Loss: Each combination was evaluated using the validation set, and the combination with the lowest validation loss was chosen.

Name: Onkar P. Kher
UID: 120407062
Directory ID: okher

3. Early Stopping: Used to monitor validation loss and prevent overfitting by halting training when no further improvement was observed.

Conclusion

In conclusion, three concatenated training datasets one from each case (open_box, corridor and special) have been used to train all the models to avoid any biases towards a particular dataset. I was unable to add any further datasets due to a huge computational overhead to concatenate another three datasets in this project. In the evaluation phase, Two concatenated test datasets have been used in the evaluation phase, to avoid crashing of the virtual environment on which the project was implemented. The neural network model has been selected as the final hypothesis for mapping the inputs to the outputs, due to their ability to incorporate and successfully map nonlinear relationships between the input and output data. Furthermore, they provide an estimate of the upper bound which is calculated using the V.C dimensions which give us a worst-case scenario which will always improve with the increase in the size of the dataset. The performance of the neural network model should improve with the inclusion of more datasets and incorporation of methods such as batch normalization . The tuning of the hyperparameters have helped in improving the performance of the model, making it more robust and effective on unseen data. The linear model and the decision tree model have not been chosen as the final models (explained in the **Insights** section) because of their high bias and low R^2 scores which can be seen in the results and the visualization plots.