Project Report: Machine Learning Model for Car-Like Robot

Introduction:

The goal of this project is to develop a regression-based model to predict the angular velocity command **(`Cmd_vel_w`)** and linear velocity command **(`Cmd_vel_v`)** for a car-like robot. The prediction is based on laser scan data and positional goals of the robot, such as final and local goal positions. A comprehensive machine learning pipeline has been implemented, which includes data ingestion, data cleaning, feature engineering, data preprocessing, model training, hyperparameter tuning, and evaluation. The pipeline ensures a systematic and reproducible approach to solving the problem.

Machine Learning Pipeline:

Data Ingestion:

The data used in this project was divided into training, validation, and testing datasets. The training data was loaded from multiple CSV files, resulting in 378,384 samples and 1,094 features. Each data point contained 1080 laser readings alongside positional features describing the final goal, local goal, and the robot's current position. The target variables were Cmd_vel_v and Cmd_vel_w, which represent linear and angular velocity commands, respectively.
The validation dataset consisted of 94,597 samples and had the same structure as the training data. This dataset was used for intermediate model evaluation during hyperparameter tuning. The test dataset contained 88,296 samples and was used to evaluate the final model's performance.

Data Cleaning and Preprocessing

The data underwent rigorous cleaning and preprocessing steps to ensure its quality and readiness for modeling. Duplicate rows were removed, and any rows containing missing values were dropped to maintain consistency. Following this, laser sensor data, consisting of 1080 features, was standardized using StandardScaler to ensure uniform scaling. Standardization reduced the impact of varying magnitudes among features, ensuring that models were not biased toward specific dimensions of the input data.

Given the high dimensionality of the laser sensor data, Principal Component Analysis (PCA) was applied for dimensionality reduction. PCA successfully retained 99% of the variance in the data while reducing the number of laser features from 1080 to approximately 100 components. This step was essential to minimize computational complexity without sacrificing critical information, allowing models to train more efficiently.

Feature Engineering:

Feature engineering was performed to create new features that could provide additional insights for the model. Specifically, two types of distance-based and angle-based features were engineered:

1. The distance to the final goal and distance to the local goal were calculated using the Euclidean distance formula. These features capture how far the robot is from its respective goals.

2. The angle to the final goal and angle to the local goal were calculated using the arctangent formula (`arctan2`), which captures the orientation of the robot relative to its goals.

The newly engineered features, namely `distance_to_final_goal`, `distance_to_local_goal`, `angle_to_final_goal`, and `angle_to_local_goal`, added meaningful context for the machine learning models to improve predictions.

Models Used:

We explored multiple regression models to predict the velocity commands. These models included linear approaches such as Linear Regression, Ridge Regression, and Lasso Regression, as well as more complex tree-based and boosting methods like Decision Tree Regressor, Random Forest Regressor, XGBoost, LightGBM, and Gradient Boosting Regressor. Each model was evaluated using Root Mean Squared Error (RMSE), which served as the primary performance metric. RMSE effectively measured the prediction errors for both the training and test datasets.

Pros and Cons of each Model used:

- **Linear Regression**:

    - *Pros*: Simple to implement, computationally efficient, and interpretable.
    - *Cons*: Assumes a linear relationship between features and targets, struggles with high-dimensional or non-linear data.
- **Ridge Regression (L2 Regularization)**:
    - *Pros*: Reduces overfitting by penalizing large coefficients, works better than Linear Regression for multicollinearity.
    - *Cons*: Still assumes linearity, and regularization strength must be carefully tuned.
- **Lasso Regression (L1 Regularization)**:
    - *Pros*: Performs feature selection by shrinking less important coefficients to zero.
    - *Cons*: May eliminate relevant features if the regularization parameter is too strong.
- **Decision Tree Regressor**:
    - *Pros*: Captures non-linear relationships and is easy to visualize.
    - *Cons*: Prone to overfitting without proper depth and pruning, sensitive to small data variations.
- **XGBoost**:
    - *Pros*: Highly efficient, handles non-linear relationships well, and performs robustly on large datasets.
    - *Cons*: Requires careful hyperparameter tuning and can be computationally expensive for large trees.
- **LightGBM**:
    - *Pros*: Faster training compared to XGBoost, efficient for large-scale datasets.
    - *Cons*: May not perform as well on small datasets and requires careful handling of hyperparameters.

Training and Validation

All models were trained using the cleaned and processed training dataset. After training all models, the top three performing models were selected for further fine-tuning of hyperparameters. The validation dataset was then used to evaluate the performance of these tuned models to ensure generalization before testing on unseen data.
During training, it was observed that simpler linear models struggled to capture the complexity of the relationships within the data, leading to relatively higher RMSE values. Tree-based and boosting models, however, performed significantly better, demonstrating their ability to handle non-linear relationships and complex interactions between features.

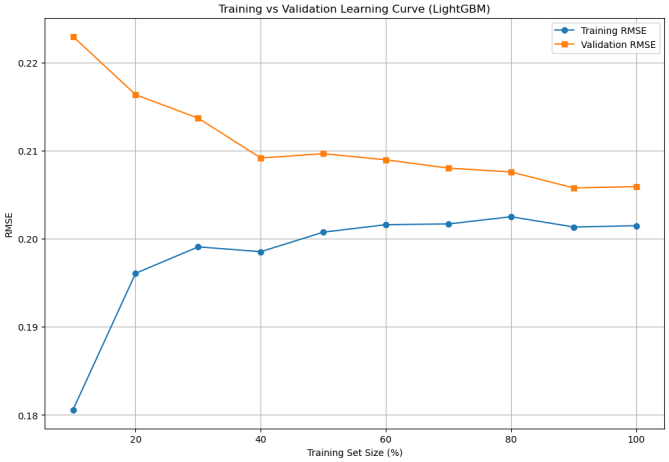
Model	Training RMSE	Test RMSE
Linear Regression	0.2235	0.2341
Ridge Regression	0.2235	0.2341
Lasso Regression	0.2246	0.2345
Decision Trees	0.1222	0.1382
XGBoost	0.0845	0.1193
LightGBM	0.1218	0.1276

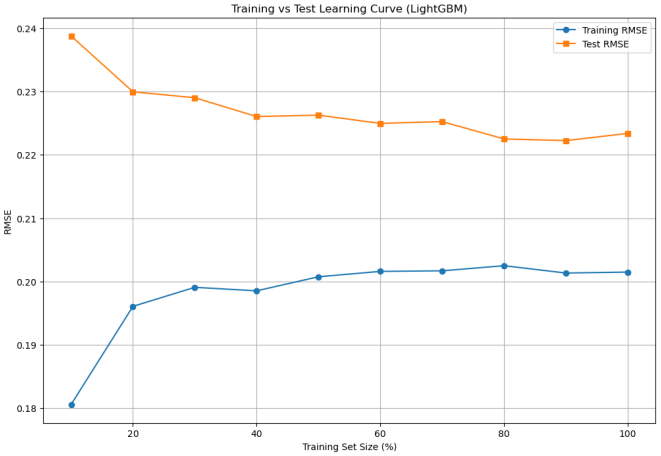
Hyperparameter Tuning
Hyperparameter tuning was carried out using RandomizedSearchCV to identify the optimal configuration for each model. For boosting models such as XGBoost key hyperparameters like n_estimators, max_depth, learning_rate, and subsample were fine-tuned. . During hyperparameter tuning, **3-fold cross-validation** was performed for all models to identify their optimal configurations.
The hyperparameter tuning process demonstrated that optimized configurations could considerably enhance model accuracy. This was particularly true for tree-based models, where an appropriate combination of depth and tree count helped reduce overfitting while maintaining strong predictive capabilities.

Model Evaluation and Selection:

Among all models, **LightGBM** emerged as the final selected model. Although XGBoost achieved slightly better performance (Train RMSE: **0.0441**, Test RMSE: **0.0901**) compared to LightGBM (Train RMSE: **0.1124**, Test RMSE: **0.1136**), LightGBM was chosen due to its significantly faster training time and lower computational requirements. Additionally, LightGBM demonstrated more stable and dependable generalization performance across validation and test datasets. This balance of performance, efficiency, and reliability made LightGBM the ideal choice for this project

Learning Curves:
Learning curves were generated for the LightGBM model to analyze performance on the training, validation, and test datasets as the training size increased. The plots revealed that both models demonstrated stable performance, with decreasing RMSE as the training size increased. The validation and test RMSE values plateaued, indicating that the models did not overfit the training data.





Conclusion:
This project successfully implemented a machine learning pipeline to predict the linear and angular velocity commands for a car-like robot. The process included rigorous data cleaning and preprocessing, feature engineering, model training, and hyperparameter tuning. LightGBM emerged as the best-performing model, achieving a low RMSE on the test data. The inclusion of engineered features and dimensionality reduction through PCA significantly contributed to the model's performance. Overall, the pipeline was designed to be systematic, reproducible, and efficient in solving the given regression task.

Name: Jonathan Deepak Kannan
UID: 120233593