Performance Analysis and Prediction in CARLA

Etai Zilberman[1], Kfir Hoftman[1], Roee Zach[1], Bar Ara[1], Roey Srugo[1]

Department of Computer Science, College of Management Academic Studies (COLMAN)
March 2025

Abstract

This project investigates vehicle performance and predictive modeling using the CARLA simulator, a high-fidelity open-source platform for autonomous driving research. We aim to analyze vehicle behavior across different routes and conditions, collecting key parameters such as lap completion time, acceleration, speed, and stability. The study leverages machine learning algorithms, including regression models, classification, clustering, and neural networks, to predict vehicle performance based on environmental and vehicular factors. Additionally, classification and clustering methods are used to group driving behaviors and assess similarities across different vehicle types. Our research contributes to a deeper understanding of self-driving vehicle dynamics, enhancing route optimization and vehicle tuning strategies.

1 Introduction

Autonomous driving technology has progressed rapidly in recent years, driven by advancements in artificial intelligence, computer vision, and vehicular simulations. Evaluating vehicle behavior under various conditions is crucial for improving self-driving capabilities, optimizing route efficiency, and ensuring safety. The CARLA simulator provides an ideal environment for studying autonomous vehicles by enabling the testing of different driving scenarios without real-world risks.

This project aims to analyze vehicle performance by extracting telemetry data from CARLA across multiple routes and vehicle configurations. We investigate the relationships between speed, acceleration, route difficulty, vehicle physics, and environmental factors, seeking to develop predictive models that estimate lap times, fuel consumption, and average speed.

To achieve these objectives, we:

- **Collect and preprocess vehicle performance data** using ROS2.
- **Analyze key performance metrics** to identify correlations between vehicle type, road conditions, and driving efficiency.
- **Develop predictive models** using machine learning techniques, including regression models (e.g., Decision Tree, Random Forest, Gradient Boosting etc.) and neural networks.
- **Implement classification and clustering methods** to categorize vehicle performance patterns.
- **Evaluate model performance** using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 scores.

By integrating data science methodologies, this study enhances the understanding of self-driving vehicle behavior.

2 Data

2.1 Data Extraction

We retrieve real-time vehicle telemetry from CARLA using ROS2, collecting data on speed, acceleration, tire pressure, and completion times across various routes and vehicle types. Simulations are run using CARLA's autopilot system to ensure standardized test conditions.

2.2 Data Preprocessing

Collected data undergoes preprocessing steps such as anomaly detection and outlier removal.

3 Regression

3.1 Objective

Regression analysis is a fundamental technique in predictive modeling, allowing us to establish relationships between independent variables (features) and dependent variables (targets). In this project, we aim to develop regression models to predict key vehicle performance metrics such as **lap time**, **fuel consumption**, and **average speed** based on vehicle characteristics.

We employ multiple regression techniques, each offering different strengths in terms of model interpretability, flexibility, and accuracy. Our goal is to assess how well various models can capture the dependencies between track complexity, vehicle parameters, and performance outcomes.

3.2 Implementation Overview

To streamline model building and ensure reproducibility, we used `Pipeline` objects in scikit-learn. Each model was wrapped in a pipeline. We did **Hyperparameter tuning** via grid search, choosing the best-performing combination of hyperparameters.

After training on the chosen hyperparameters, we evaluated each model on a held-out test set. We used metrics such as R^2, MSE (Mean Squared Error), RMSE (Root Mean Squared Error), and MAE (Mean Absolute Error) to compare performance. Below are the results for three targets: **lap_time**, **fuel_consumption**, and **average_speed**.

3.3 Linear Regression

Target	R^2	RMSE	MAE
Lap Time	0.8322	0.7387	0.4198
Fuel Consumption	0.9836	0.0832	0.0641
Average Speed	0.2229	0.4975	0.3833

Table 1: Linear Regression Performance Metrics

While Linear Regression was our baseline, it often performed surprisingly well (especially on *fuel_consumption*). This strong performance may indicate that *fuel_consumption* exhibits predominantly linear or near-linear relationships with the available features. Additionally, if the dataset is well-behaved (e.g., minimal outliers, limited range of feature values), then linear regression can perform surprisingly well despite its simplicity.

3.4 Decision Tree Regressor

Target	R^2	RMSE	MAE
Lap Time	0.8322	0.7387	0.4198
Fuel Consumption	0.9836	0.0832	0.0641
Average Speed	0.2223	0.4977	0.3835

Table 2: Decision Tree Regressor Performance Metrics

Decision Tree Regressors use a series of hierarchical splits on the input features to predict a continuous target. They can capture complex, nonlinear relationships without requiring much data preprocessing. However, they are prone to overfitting if grown too deep. In our results, the Decision Tree achieved high R^2 on *lap time* and *fuel consumption*, indicating its ability to model those targets well with tuned hyperparameters.

3.5 Random Forest Regressor

Target	R^2	RMSE	MAE
Lap Time	0.8319	0.7394	0.4198
Fuel Consumption	0.9836	0.0832	0.0641
Average Speed	0.2231	0.4975	0.3832

Table 3: Random Forest Regressor Performance Metrics

Random Forest Regressors are ensembles of multiple decision trees, each trained on a different bootstrap sample of the dataset and a random subset of features. By averaging the predictions of many weak learners, they reduce overfitting and improve stability. Our findings show consistently strong R^2 scores across targets, making Random Forests a robust choice for scenarios where individual decision trees may overfit.

3.6 Gradient Boosting Regressor

Target	R^2	RMSE	MAE
Lap Time	0.8323	0.7387	0.4199
Fuel Consumption	0.9836	0.0832	0.0641
Average Speed	0.2229	0.4975	0.3833

Table 4: Gradient Boosting Regressor Performance Metrics

Gradient Boosting builds an ensemble of weak learners sequentially, where each new learner focuses on the residual errors of the previous ones. This iterative error-correction approach often yields highly accurate models. Here, Gradient Boosting excelled in predicting *lap time* and *fuel consumption*, reflecting its strength in capturing complex interactions among features.

3.7 Support Vector Regression (SVR)

Target	R^2	RMSE	MAE
Lap Time	0.8291	0.7456	0.4215
Fuel Consumption	0.9824	0.0862	0.0693
Average Speed	0.2191	0.4987	0.3840

Table 5: SVR Performance Metrics

SVR attempts to fit the data within a margin of tolerance (the ϵ-tube) while minimizing model complexity. The choice of kernel (e.g., RBF, linear) significantly influences its performance. Although it achieved competitive results in our experiments, especially for *fuel consumption*, it did not surpass tree-based methods for *lap time* and *average speed*.

3.8 Gaussian Process Regressor (GPR)

Target	R^2	RMSE	MAE
Lap Time	0.8322	0.7387	0.4198
Fuel Consumption	0.9836	0.0832	0.0641
Average Speed	0.2229	0.4975	0.3833

Table 6: Gaussian Process Regressor Performance Metrics

GPR is a Bayesian, non-parametric approach that provides not only predictions but also uncertainty estimates. It posits that data come from a Gaussian Process, defined by a mean function and a covariance (kernel) function. In our work, GPR demonstrated strong performance similar to tree-based ensembles for *lap time* and *fuel consumption*, suggesting that a smooth kernel function can effectively model the underlying relationships.

3.9 Conclusion

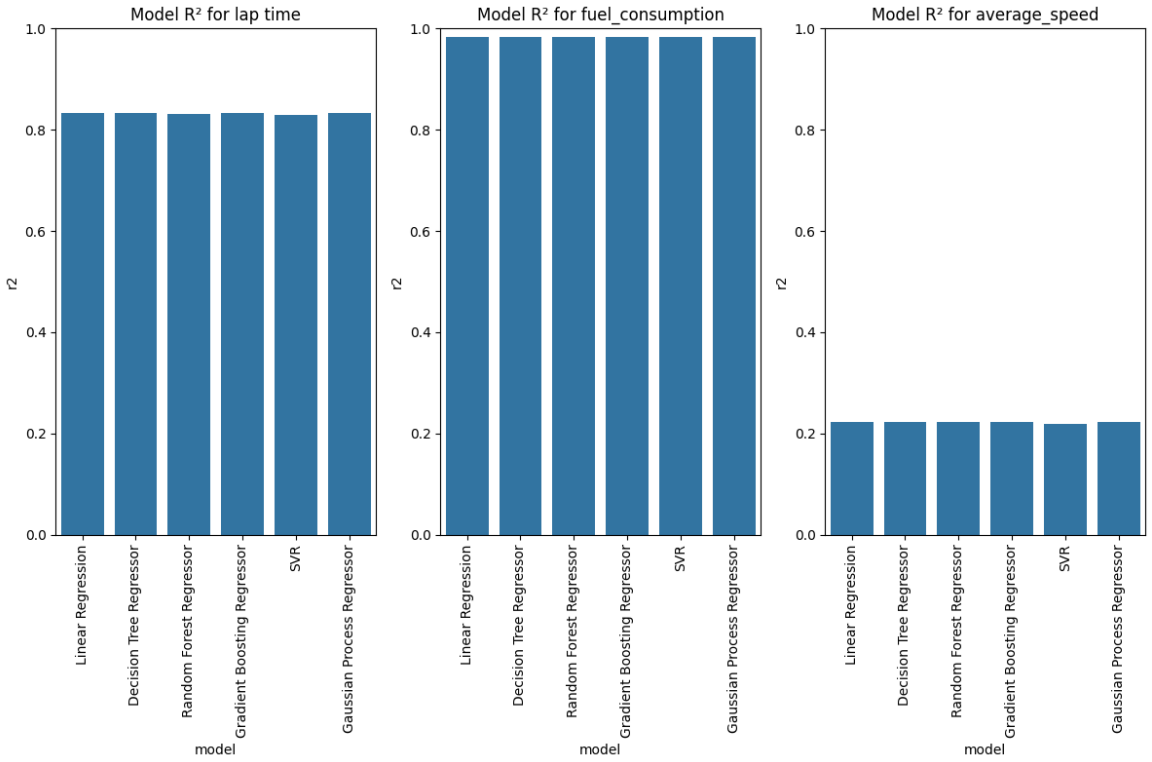

Figure 1: Regression r2 conclusion

3.10 Neural Networks (Basic)

Below are the results of a standard feedforward neural network (*Basic NN*) for each target:

Target	R^2	RMSE	MAE
Lap Time	0.7330	0.9317	0.6692
Fuel Consumption	0.9803	0.0923	0.0720
Average Speed	0.1819	0.5087	0.3960

Table 7: Basic Neural Network Performance Metrics

3.11 Neural Networks (with Embeddings)

We also experimented with a neural network that includes an *embedding layer* for categorical variables, followed by dense layers. Below are the results:

Target	R^2	RMSE	MAE
Lap Time	0.8180	0.0769	0.0448
Fuel Consumption	0.9813	0.8985	0.6953
Average Speed	-0.0208	0.5682	0.4773

Table 8: Neural Network w/ Embeddings Performance Metrics

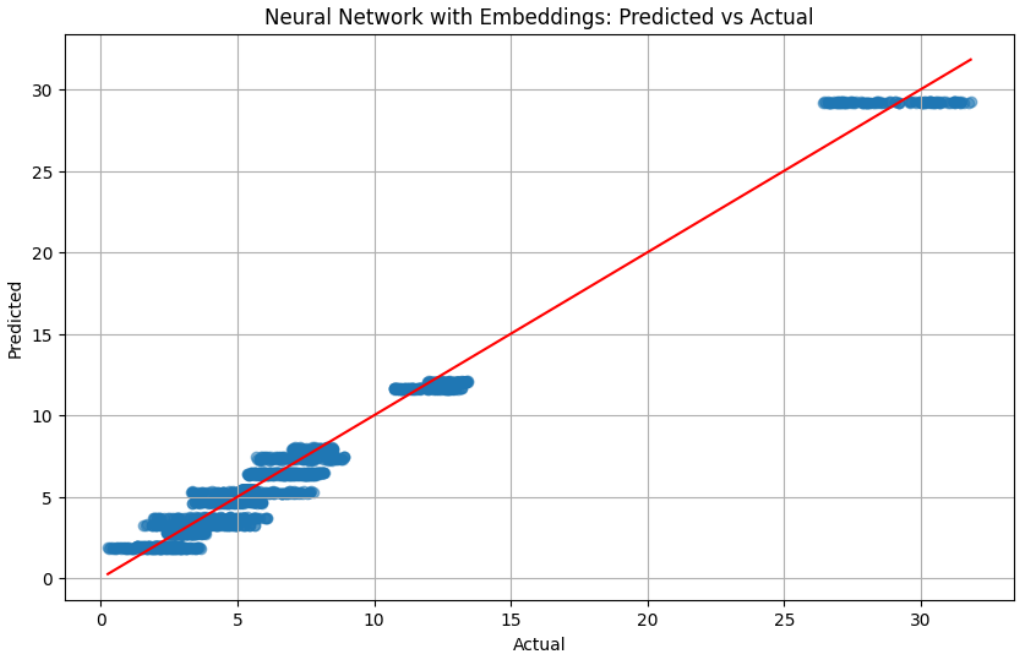

Figure 2: Embedded Neural Network plot

3.12 Comparison of Models and Conclusion

Below are the **best models** for each target according to R^2 and RMSE:

For completeness, we provided a quick side-by-side snapshot of each model's performance on each target:

From these results, several observations stand out:

- **Gradient Boosting** and **Decision Tree** are consistently strong performers for *lap time* and *fuel consumption*.
- Many models yield **very high** R^2 on *fuel consumption*, suggesting that this target is relatively easier to predict with the given features.

Model	Lap Time	Fuel Cons.	Avg. Speed
Linear Regression	0.8322	0.9836	0.2229
Decision Tree	0.8322	0.9836	0.2223
Random Forest	0.8319	0.9836	0.2231
Gradient Boosting	0.8323	0.9836	0.2229
SVR	0.8291	0.9824	0.2191
Gaussian Process	0.8322	0.9836	0.2229
NN (Basic)	0.7330	0.9803	0.1819
NN (Embeddings)	0.8180	0.9813	-0.0208

Table 9: High-Level Comparison of All Models (R^2 Scores)

- *Average speed* shows **lower** R^2 overall, implying it might be influenced by additional factors not captured in the current feature set.
- **Neural networks**—especially with embeddings—sometimes underperform on these data, possibly due to scale differences, the need for more tuning/regularization, or additional data.

Final Remarks:

- Pipelines and hyperparameter tuning helped optimize each model and maintain a clear, modular codebase.
- For each target, different algorithms can excel.
- Neural networks can outperform traditional methods if given enough data and carefully tuned architectures, but can also be more sensitive to data preprocessing.

Overall, these regression experiments provide meaningful insights into vehicle performance prediction. With additional tuning, feature selection, or more advanced architectures, certain models (especially for *average speed*) may see further improvements.

4 Classification and Clustering

4.1 Objective

In this section, we perform multiple classification tasks (both three-class and four-class labels) and cluster analyses on the vehicle data. The goal is to:

- Predict and classify **vehicle types** (Car, Motorcycle, Truck, Van).
- Discover natural groupings via **clustering**.

4.2 Dataset Overview

Dataset shape: 8500 rows, 13 features.
Vehicle Types and Counts:

- Car: 3500

- Motorcycle: 2000
- Truck: 1500
- Van: 1500

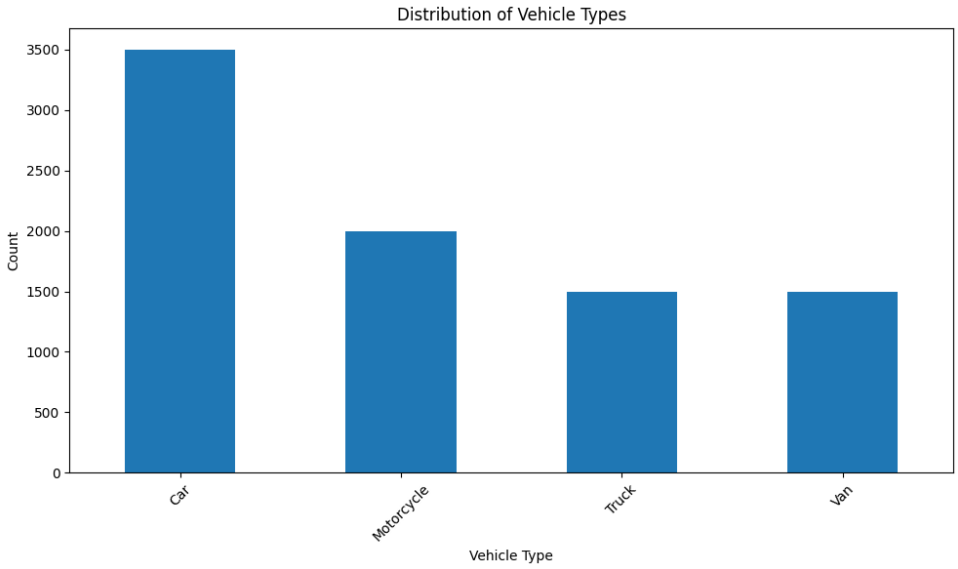

Figure 3: Embedded Neural Network plot

Train/Test Split: We split data into a training set (70%) and a test set (30%). **Pipeline and Tuning:** We used a grid search over hyperparameters for each model, selecting the best cross-validation scores.

4.3 Classification: lap time

We separated lap times into three classes: Fast, Medium and Slow.

4.3.1 Decision Tree Classifier

Performance (Test Set):
Accuracy = 0.83

Class	Precision	Recall	F1-Score	Support
0	0.81	0.83	0.82	499
1	0.84	0.86	0.85	1271
2	0.84	0.78	0.81	780
Weighted Avg	0.83	0.83	0.83	2550

Table 10: Decision Tree – Classification report Metrics

Discussion: The decision tree achieved an overall accuracy of 0.83. Hyperparameters such as `max_depth` and `min_samples_split` were tuned to mitigate overfitting.

4.3.2 Naïve Bayes Classifier

Naïve Bayes uses Bayes' theorem under an independence assumption:

$$P(C \mid X) = \frac{P(X \mid C)\, P(C)}{P(X)}. \tag{1}$$

Performance (Test Set):
Accuracy = 0.75

Class	Precision	Recall	F1-Score	Support
0	0.70	0.85	0.77	499
1	0.89	0.63	0.74	1271
2	0.67	0.89	0.76	780
Weighted Avg	0.79	0.75	0.75	2550

Table 11: Naïve Bayes – Classification report Metrics

Discussion: Although Naïve Bayes assumes independence of features, it managed a 0.75 accuracy. Hyperparameter tuning (e.g., `var_smoothing`) improved stability.

4.3.3 XGBoost Classifier

Performance (Test Set):
Accuracy = 0.84

Class	Precision	Recall	F1-Score	Support
0	0.82	0.82	0.82	499
1	0.83	0.88	0.85	1271
2	0.87	0.78	0.82	780
Weighted Avg	0.84	0.84	0.84	2550

Table 12: XGBoost – Classification report Metrics

Discussion: XGBoost yielded a higher accuracy (0.84) than Naïve Bayes or the Decision Tree. Optimal parameters included `learning_rate=0.01`, `max_depth=7`, and `n_estimators=200`.

4.3.4 Random Forest Classifier

Performance (Test Set):
Accuracy = 0.84

Class	Precision	Recall	F1-Score	Support
0	0.80	0.85	0.82	499
1	0.84	0.87	0.85	1271
2	0.86	0.78	0.82	780
Weighted Avg	0.84	0.84	0.84	2550

Table 13: Random Forest – Classification report Metrics

Discussion: The random forest also reached an 0.84 accuracy, closely matching XGBoost. Best parameters were `max_depth=7`, `min_samples_split=7`, and `n_estimators=120`.

4.3.5 Comparison and Observations

- **Naïve Bayes** (0.75 accuracy) underperformed relative to the tree-based models but is significantly faster to train.
- **XGBoost** and **Random Forest** both achieved 0.84 accuracy, outperforming Decision Tree (0.83).
- **Hyperparameter tuning and pipelines were critical in maximizing each model's performance.**

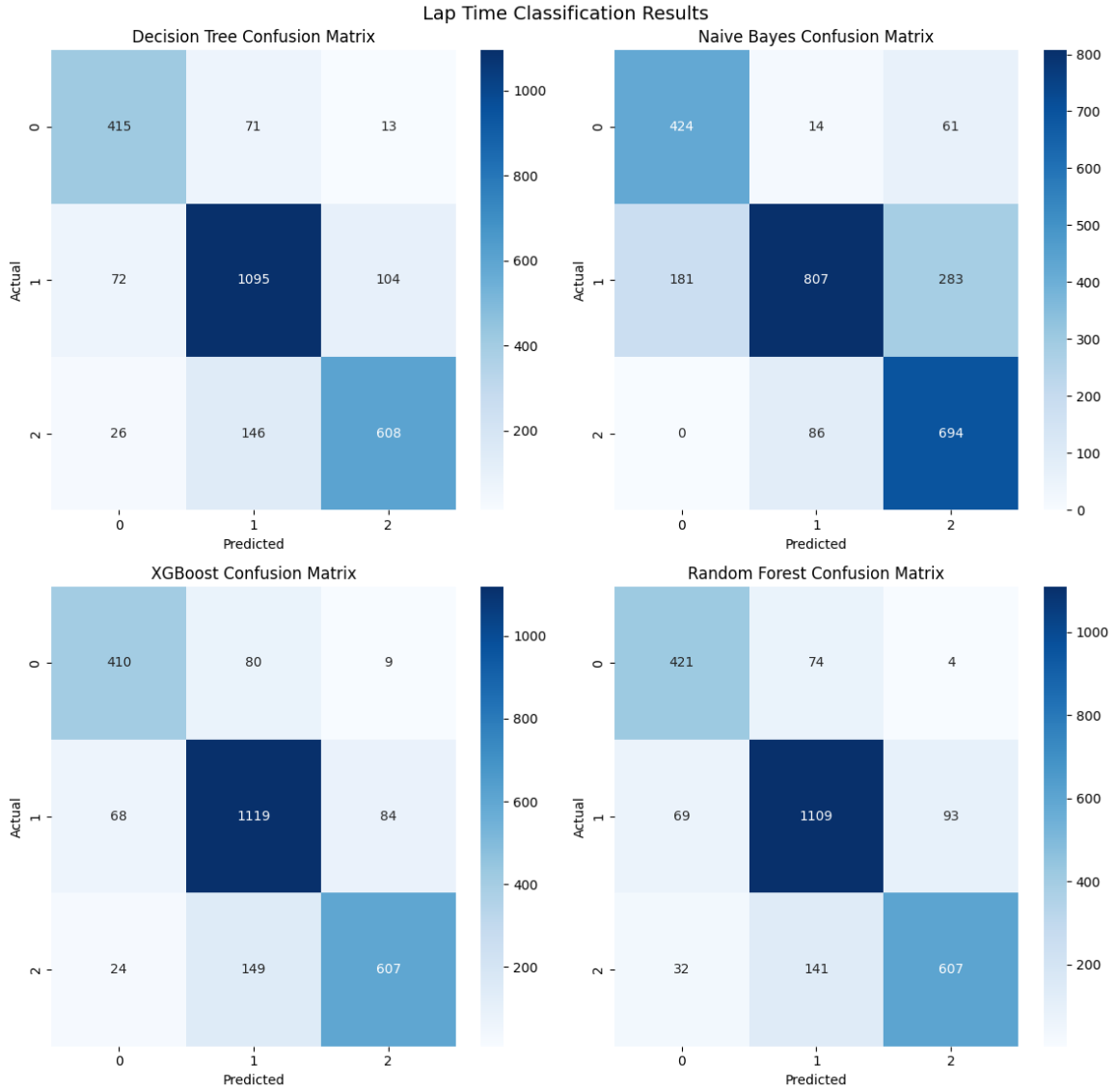

Figure 4: Comparison of three-class classification models.

4.4 Classification: Vehicle Types (Car, Motorcycle, Truck, Van)

In a second classification task, we predicted **vehicle type** (4 classes). Below are the best-performing metrics.

4.4.1 Random Forest Classifier

Class	Precision	Recall	F1-Score	Support
Car	1.00	1.00	1.00	1050
Motorcycle	0.99	0.81	0.89	600
Truck	0.98	0.89	0.93	450
Van	0.73	0.98	0.83	450
Accuracy			0.93	
Weighted Avg	0.95	0.93	0.93	2550

Table 14: Random Forest – Vehicle Type Prediction

Tuned parameters:

- `max_depth=7`,
- `min_samples_split=7`,
- `n_estimators=120`.

4.4.2 Gradient Boosting Classifier

Class	Precision	Recall	F1-Score	Support
Car	1.00	1.00	1.00	1050
Motorcycle	0.99	0.81	0.89	600
Truck	0.99	0.88	0.93	450
Van	0.73	0.99	0.84	450
Accuracy			0.93	
Weighted Avg	0.95	0.93	0.93	2550

Table 15: Gradient Boosting – Vehicle Type Prediction

Tuned parameters:

- `learning_rate=0.01`,
- `max_depth=5`,
- `n_estimators=200`.

4.4.3 XGBoost Classifier

Class	Precision	Recall	F1-Score	Support
Car	1.00	1.00	1.00	1050
Motorcycle	0.99	0.81	0.89	600
Truck	0.97	0.89	0.93	450
Van	0.73	0.97	0.83	450
Accuracy			0.93	
Macro Avg	0.92	0.92	0.91	2550
Weighted Avg	0.94	0.93	0.93	2550

Table 16: XGBoost – Vehicle Type Prediction

Tuned parameters:

- `learning_rate=0.01`,
- `max_depth=7`,
- `n_estimators=200`.

4.4.4 Naïve Bayes Classifier

Class	Precision	Recall	F1-Score	Support
Car	0.86	0.98	0.92	1050
Motorcycle	0.58	1.00	0.73	600
Truck	0.62	0.18	0.28	450
Van	0.50	0.20	0.29	450
Accuracy			0.71	
Weighted Avg	0.69	0.71	0.65	2550

Table 17: Naïve Bayes – Vehicle Type Prediction

Tuned parameters: `var_smoothing=1e-09`.

4.4.5 Comparison and Observations (Four-Class Classification)

- **Gradient Boosting** achieved the highest accuracy (0.9314), edging out Random Forest and XGBoost (both around 0.93).
- **Naïve Bayes** performed worse (0.71 accuracy) but was very fast and simple to train.
- Pipeline tuning with cross-validation improved performance across all estimators.

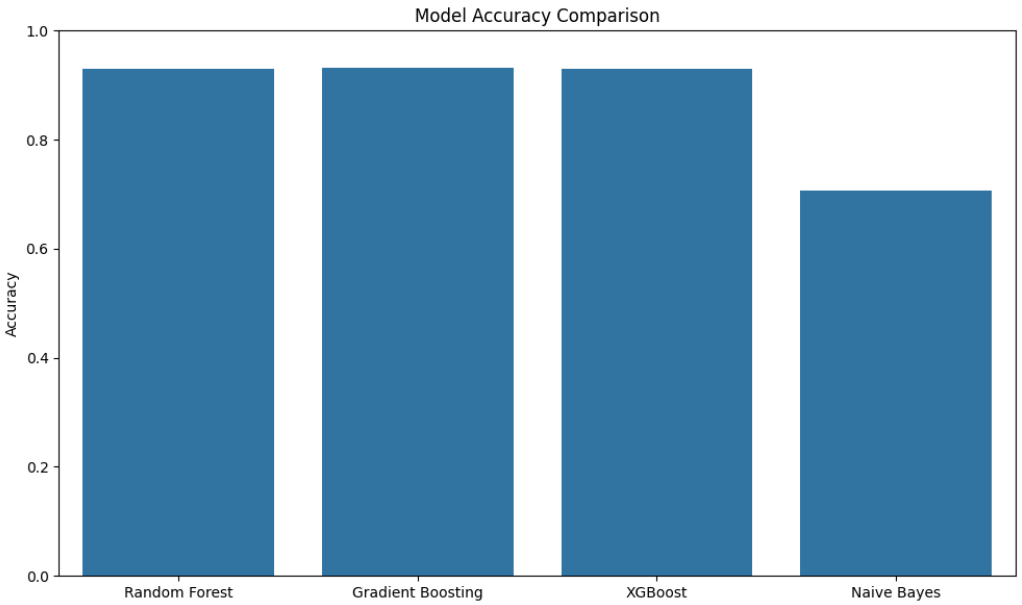

Figure 5: Comparison of vehicle-type classification models by accuracy.

4.5 Clustering Analysis

Clustering was performed to find natural groupings of vehicles based on speed, fuel consumption, lap time, etc.

4.5.1 Cluster Summary

We extracted four clusters ($k = 4$) using methods of Agglomerative Clustering. Agglomerative Clustering is a hierarchical clustering algorithm in which clusters are formed bottom-up by iteratively merging the closest pairs of clusters until a stopping criterion (like the number of clusters) is met. Below are the key statistics:

Cluster	Car	Motorcycle	Truck	Van
0	0	2000	132	1000
1	517	0	405	336
2	2983	0	463	164
3	0	0	500	0

Table 18: Vehicle Types per Cluster

Observations:

- **Cluster 0** is dominated by Motorcycles (2000) and Vans (1000), with relatively moderate speed and higher lap time.
- **Cluster 1** and **Cluster 2** mostly contain Cars (517 in cluster 1, 2983 in cluster 2).
- **Cluster 3** is purely Trucks, with slightly higher average speed than cluster 1 but a somewhat longer lap time than cluster 2.

Dominant Vehicle Type per Cluster:

- Cluster 0: Motorcycle

- Cluster 1: Car
- Cluster 2: Car
- Cluster 3: Truck

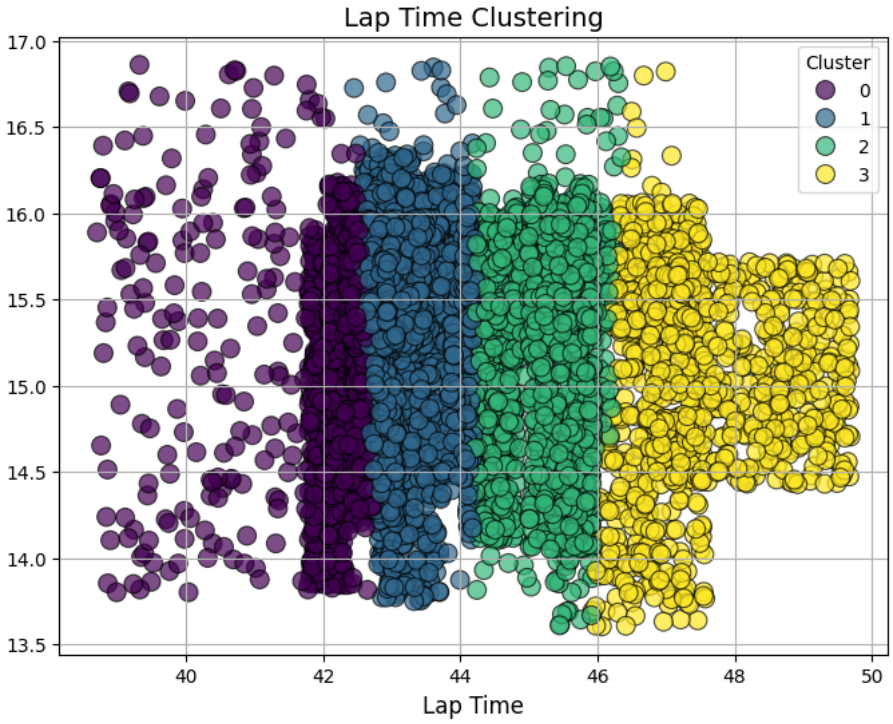

Figure 6: clustering of vehicles – lap time vs avg speed.

4.5.2 Conclusion (Clustering)

- K-Means identified four clusters distinguished by speed, fuel consumption, and lap time.
- Each cluster was dominated by a specific vehicle type or combination.
- Such clustering insights can guide further analysis of driving patterns, fuel efficiency, and performance across different vehicle classes.

4.6 Summary

Overall, the combined results illustrate that careful data preprocessing, hyperparameter tuning, and model selection can yield strong predictive models for vehicle metrics. For continuous targets, ensemble-based regression models generally shone, whereas classification tasks often favored tree-based or boosting algorithms. Clustering provided additional insights for grouping similar driving patterns or vehicles. With additional time, we could gather more data which could lead to a better performance.