2º report: Predicting Car CO_2 Emissions
A Comparative Study of Machine Learning and Deep Learning Models

Phillipp Würfel

Python Backend Engineer | Software Engineer

Karl Richard Georg Kutz

Mechanical Engineer

Leonel Leite Barros

Master in Economic Theory

Thomas Igor Lisowsky

Junior Web Developer

Abstract

This study evaluates the performance of machine learning and deep learning models in three predictive tasks using structured data from newly registered vehicles in the European Union: predicting electric energy consumption, estimating specific CO_2 emissions, and classifying fuel types. Two datasets with distinct characteristics were used.
The analysis involved linear baselines, ensemble models (GradientBoosting, ExtraTreeRegressor, LGBM, HistGradientBoosting), and neural networks built with Keras. Benchmarking was conducted using LazyRegressor, and hyperparameter tuning employed Optuna and RandomizedSearchCV.

Introduction

This report presents a comprehensive comparative analysis of machine learning and deep learning models applied to structured data on newly registered vehicles in the European Union. The modeling effort is divided into three parts, each targeting a different predictive task: (1) estimating electric energy consumption, (2) predicting specific CO_2 emissions from combustion-engine vehicles, and (3) classifying fuel types based on vehicle characteristics.

To accomplish these objectives, the project uses a wide range of supervised learning algorithms, including linear models, tree-based ensemble methods, and multilayer perceptrons (MLPs) implemented with Keras. The approach combines automated model benchmarking (LazyRegressor), advanced hyperparameter tuning (Optuna, GridSearchCV, RandomizedSearchCV), and model interpretability techniques such as feature importance analysis and SHAP values.

Each modeling stage was tailored to the nature of the respective target variable—regression for energy consumption and CO_2 emissions, classification for fuel types. Special attention was given to model evaluation using performance metrics such as R^2, RMSE, and classification accuracy, as well as error distribution analysis and visualization of prediction patterns. The study also explores the implications of data volume and structure on model performance, including tests on large-scale datasets and reduced training scenarios.

The findings contribute to a broader understanding of how different model families—ranging from interpretable decision trees to complex deep learning architectures—perform in real-world vehicle datasets. The report not only compares predictive accuracy but also emphasizes model generalization, robustness, computational cost, and suitability for deployment.

Part I: The Prediction Of Vehicles' Electric Energy Consumption In The EU

This part of the report focuses on predicting the electric energy consumption of vehicles registered in the European Union. To achieve this, the pre-processed dataset described in the first report was used to run models with varying degrees of complexity — ranging from simple Linear Regression to Deep Learning models, including traditional Machine Learning approaches. To ensure robust evaluation and meaningful model comparison, the available dataset was divided into two distinct subsets:

- ***First dataset:***
  - Target variable: electric_energy_consumption
  - Explanatory variables: mass_vehicle, weltp_test_mass, engine_capacity, engine_power, erwltp, year, electric_range, fuel_consumption, specific_co2_emissions, innovative_technologies, fuel_type, fuel_mode

- ***Second dataset:***
    - Target is electric_energy_consumption
    - Explanatory variables: member_state, manufacturer_name_eu, vehicle_type, commercial_name, category_of_vehicle, mass_vehicle, engine_power, year, electric_range

The goal of this report is to compare how models of varying complexity perform in the task of predicting the target variable. The modelling process was organized into multiple stages:

- *Initial round of baseline models:*
  Simple linear models were used to establish baseline results for both datasets.
    - For the first dataset: OLS Value, Ridge CV Value, and LASSO Value were applied.
    - For the second dataset: Linear Regression, Ridge CV, and LASSO Regression and XGBRegressor, DecisionTreeRegressor and RandomForestRegressor were tested.
- *Model benchmarking with LazyRegressor:*
  The LazyRegressor library was employed to identify the most promising machine learning algorithms by automatically evaluating a wide range of models on both datasets.
- *Model selection stage:*
  Based on the benchmarking results, models were chosen for each dataset:
    - Gradient Boosting Regressor was selected for the first dataset.
    - Extra Tree Regressor was selected for the second dataset.
- *Hyperparameter optimization and cross-validation:*
    - For the first dataset, Bayesian optimization was performed using Optuna, followed by cross-validation to validate the results.
    - For the second dataset, GridSearchCV and RandomizedSearchCV were applied for hyperparameter tuning, followed by cross-validation.
- *Machine learning and model interpretation:*
  This section presents the final performance metrics of the selected models: Gradient Boosting for the first dataset and Extra Tree Regressor for the second.
- *Deep learning and model interpretation:*
  Finally, a deep learning model was trained for each dataset.

- MLP Regression Model with Keras was applied to the first dataset.
- SHAP was used to interpret the second dataset and evaluate feature contributions.

This structured approach allows for a comprehensive comparison across modeling strategies, highlighting both predictive performance and interpretability.

1. Initial round of baseline models

As an initial baseline, simpler linear models were tested on both datasets. These models showed limited predictive power and were used primarily to establish a starting point for further comparison.

1.1 First Dataset – Linear Models

The following models were applied to the first dataset: OLS, RidgeCV, and Lasso Regression. Performance was limited, and no significant improvements were observed.

Metric / Feature	OLS Value	RidgeCV Value	Lasso Value
R^2	0.67	0.67	0.67
RMSE	—	23.60	23.60
MAE	—	16.51	16.51
Best alpha	—	11.51395	0.00032
Intercept	-77.38	—	—
mass_vehicle	0.1366	34.8985	34.8982
engine_power	0.0324	1.4558	1.4557

engine_capacity	-0.0121	-5.7049	-5.7044
electric_range	0.0441	0.8200	0.8194
fuel_consumption	-1.9470	-1.4140	-1.4127
specific_co2_emissions	0.3137	4.1893	4.1877

1.2 Second Dataset – Linear Models

Similarly, simple models were tested on the second dataset: Linear Regression, RidgeCV, and Lasso Regression. The performance remained modest, as shown below.

Metric	Linear Regression	RidgeCV	Lasso Regression
Train R^2	0.4324	0.4316	0.4045
Test R^2	0.4329	0.4311	0.4041
Train MSE	1.5103	1.5137	—
Test MSE	1.5124	1.5160	—
Best alpha	—	50.0	0.1802
Lasso non-zero coef	—	—	9 of 11

Since none of the tested models yielded satisfactory results on either dataset, the next step was to search for more suitable machine learning models. To this end, the LazyRegressor library was applied to the second dataset to identify potentially better-performing algorithms.

Given the structural similarity between the two datasets and the high computational cost involved, it was decided to use the LazyRegressor results as a preliminary guide for both datasets before running further model comparisons.

2. Model benchmarking with LazyRegressor

The table below presents the 10 best model results, sorted by $R^{2,}$ obtained from the use of the LazyRegressor library, which was employed to benchmark 37 different regression models in a fast and automated manner. This step belongs to the model selection phase, following data preprocessing and exploratory analysis, and aims to identify the most promising machine learning algorithms for the dataset under study.

LazyRegressor allows for the rapid comparison of models by training and evaluating them with minimal manual intervention, offering a practical overview of their relative performance. The evaluated models range from simple linear regressors, such as Ridge and Bayesian Ridge, to more complex ensemble methods, including Gradient Boosting and Extra Trees, as well as neural networks like the Multi-Layer Perceptron (MLP).

Each model was assessed based on four key metrics:
• Adjusted R-Squared and R-Squared, which indicate the proportion of explained variance;
• RMSE (Root Mean Squared Error), which reflects the average magnitude of prediction errors;
• Time Taken, measuring the computation time in seconds.

Model Evaluation Results

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
ExtraTreesRegressor	0.98	0.98	3.41	539.32
ExtraTreeRegressor	0.97	0.97	3.94	13.62
KNeighborsRegressor	0.97	0.97	4.1	1019.17
LGBMRegressor	0.96	0.96	4.95	11.3
MLPRegressor	0.96	0.96	4.95	1350.02
HistGradientBoostingRegressor	0.95	0.95	5.5	23.23
GradientBoostingRegressor	0.89	0.89	8.19	149.53
BayesianRidge	0.86	0.86	9.27	17.8
RidgeCV	0.86	0.86	9.28	19.11
Ridge	0.86	0.86	9.28	8.82

The best-performing model in terms of predictive accuracy was the ExtraTreesRegressor, with an Adjusted R-Squared of 0.98 and the lowest RMSE of 3.41. However, its training time was significantly higher compared to simpler models. Other models such as KNeighborsRegressor and MLPRegressor also showed strong performance but at the cost of greater computational time. On the other hand, linear models like Ridge and BayesianRidge offered fast execution times with moderate accuracy.

This evaluation provides a data-driven basis for selecting candidate models for further tuning and deployment, balancing prediction performance and computational cost.

3. Model selection stage

Based on the results obtained through LazyRegressor, a set of promising machine learning models was selected for further evaluation and hyperparameter tuning. The selection was made with the goal of covering a range of model complexities, from simple decision trees to more sophisticated ensemble methods.

3.1 First Dataset

The following models were selected to be tested with the first dataset:

a. Gradient Boosting Model

b. LGBM Regressor with Keras

c. HistGradientBoostingRegressor

3.2 Second Dataset

The following models were selected to be tested with the second dataset:

a. ExtraTreeRegressor

b. RandomForestRegressor

c. DecisionTreeRegressor

4. Hyperparameter Optimization and Cross-Validation

4.1 First Dataset – Bayesian Optimization with Optuna[1]

The model tuning for the first dataset was performed using Bayesian optimization with Optuna. The configuration and execution of the study were defined to ensure an efficient and reproducible search for optimal hyperparameters.

Hyperparameter Optimization – Search Space (Optuna)

Hyperparameter	Search Range	Type
n_estimators	50 to 100	Integer
learning_rate	0.05 to 0.1	Float
max_depth	2 to 3	Integer
subsample	0.8 to 1.0	Float
min_samples_split	2 to 5	Integer

Hyperparameter Tuning with Optuna – Study Configuration and Execution

The hyperparameter optimization process was conducted using Optuna, with mechanisms for persistent study tracking and efficient search. The following settings and procedures were adopted:

[1] Hyperparameter tuning using Optuna was conducted exclusively for the Gradient Boosting Model, as the other models require a distinct preprocessing strategy and rely on more basic preprocessing steps than the current optimization setup.

- Search Strategy: Optuna was configured to perform a Bayesian optimization (Tree-structured Parzen Estimator) aiming to maximize the cross-validated R^2 score of the model.
- Study Persistence: The study progress was saved and loaded from a file named optuna_study.pkl, allowing interruption handling and continued optimization across sessions.
- Objective Function: The model used for tuning was a GradientBoostingRegressor with the hyperparameters described in the table above being optimized.
- Cross-Validation (cv = 3): A 3-fold cross-validation was used to speed up evaluation during the tuning phase, providing a balance between performance estimation and computational cost.
- Number of Trials (n_trials = 3): For demonstration or initial testing purposes, only three optimization trials were executed. The design supports easy scaling for larger search spaces.
- Progress Saving: The study state was automatically saved to disk every 5 trials, ensuring that intermediate results would not be lost in case of interruption.
- Reproducibility and Safety: The system checked for existing saved studies before creating a new one, allowing safe recovery and continuation of previous optimization work.

This setup ensured a controlled, trackable, and extendable environment for hyperparameter optimization, enabling the exploration of efficient model configurations with minimal manual intervention.

Optuna Optimization Results

Model	Best Parameters	Best CV Score	Test R^2 Score	n_estimators
GradientBoostingRegressor	{'n_estimators': 74, 'learning_rate': 0.0993, 'max_depth': 3, 'subsample': 0.8345, 'min_samples_split': 5}	0.962	0.961	74

The table above presents the results of hyperparameter tuning for tree-based regression models using Optuna. This phase builds on the initial benchmarking conducted with LazyRegressor and focuses on refining the most promising models using the Optuna framework to perform a more refined and adaptive hyperparameter optimization, relying on Bayesian optimization to efficiently explore the search space and identify high-performing parameter configurations.

In this case, the optimization focused on the GradientBoostingRegressor, a robust ensemble model particularly suited for handling non-linear relationships and complex feature interactions. During the search process, various combinations of hyperparameters were evaluated using cross-validation to maximize the R^2 score. The results include:

- Best Parameters: the hyperparameter configuration that led to the highest cross-validated score;
- Best CV Score: the mean cross-validation R^2 score for the best parameter set;
- Test R^2 Score: the R^2 performance on the held-out test dataset;
- n_estimators: the number of boosting stages used, as determined by the optimization process.

The GradientBoostingRegressor, when optimized with Optuna, achieved a test R^2 score of 0.961, with a cross-validated R^2 of 0.962. The model performed strongly, benefiting from a carefully tuned configuration that included 74 estimators, a learning rate of approximately 0.099, and a shallow tree depth of 3, combined with a subsampling rate of 83%. This setup provided a strong balance between performance and overfitting control.

These results highlight the strength of Optuna's targeted search approach, which can yield performance gains over traditional randomized search, particularly when used in combination with robust ensemble models like GradientBoostingRegressor. The optimized model demonstrates both high predictive accuracy and efficient resource use, making it a promising candidate for deployment.

4.2 First Dataset – Gradient Boosting – Cross-Validation Performance Evaluation

R^2 Cross-Validation Statistics

Model	Average R^2	R^2 Variance
GradientBoostingRegressor	0.9862	1.01e-08

The table above presents the R^2 cross-validation statistics for the GradientBoostingRegressor model optimized using Optuna and trained on the first dataset. This evaluation aims to assess the model's generalization performance and stability across different data folds.

The metrics reported include:
- Average R^2: the mean coefficient of determination across all cross-validation folds, indicating the model's ability to consistently explain variance in the target variable;
- R^2 Variance: the variation in R^2 scores across the folds, which serves as a measure of the model's stability during training and validation.

The GradientBoostingRegressor achieved an exceptionally high average R^2 score of 0.9862 with an almost negligible variance of approximately 1.01e-08. These results confirm not only the model's strong in-sample performance but also its remarkable consistency across cross-validation folds. Such minimal variance indicates that the model's predictions are highly stable, and the performance does not depend heavily on specific training splits.

These findings support the conclusion that the GradientBoostingRegressor is both accurate and robust, making it a strong candidate for deployment in production settings, especially where prediction consistency is crucial.

4.3 Second Dataset - Optimization with RandomizedSearchCV

Hyperparameter Search Space – RandomizedSearchCV

Hyperparameter	Search Values	Type
n_estimators	[50, 100, 200]	Integer
max_depth	[10, 20, 30]	Integer
min_samples_split	[2, 5, 10]	Integer
min_samples_leaf	[1, 2, 4]	Integer
max_features	[None, 'sqrt', 'log2']	Categorical

Hyperparameter Tuning with RandomizedSearchCV

To optimize the model's performance, RandomizedSearchCV was applied with the following configuration:

- Estimator: The base model used in the tuning process.
- Parameter Distributions: A predefined grid of hyperparameters from which values were randomly sampled.
- Number of Iterations (n_iter = 5): Five random combinations of hyperparameters were tested. This choice aimed to reduce computational cost while still exploring relevant portions of the search space.
- Cross-Validation (cv = 5): A 5-fold cross-validation was performed to evaluate each parameter set, ensuring robust validation across different subsets of the training data.

- Random State (random_state = 42): A fixed seed was set to ensure the reproducibility of results.

This configuration allowed for efficient hyperparameter tuning by balancing exploratory capacity and computational feasibility, especially when dealing with large datasets or more complex models.

RandomizedSearchCV Results – Tree-Based Models

Model	Best Parameters	Best CV Score	Test R^2 Score	n_estimators
ExtraTreeRegressor	{'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': None, 'max_depth': 20}	0.966	0.96	—
ExtraTreesRegressor	{'n_estimators': 50, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features': None, 'max_depth': 10}	0.894	0.89	50
RandomForestRegressor	{'n_estimators': 50, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features':	0.947	0.70	50

	None, 'max_depth': 10}			
DecisionTreeRegressor	{'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': None, 'max_depth': 20}	0.978	0.73	—

The table above presents the results of hyperparameter tuning for tree-based regression models using RandomizedSearchCV. This phase follows the initial model benchmarking conducted with LazyRegressor and focuses on refining the most promising tree-based algorithms through randomized cross-validated grid search, in order to optimize their predictive performance.

Four tree-based models were evaluated: ExtraTreeRegressor, ExtraTreesRegressor, RandomForestRegressor, and DecisionTreeRegressor. For each model, a set of hyperparameters was sampled and tested using cross-validation to identify the configuration yielding the best average validation score. The results include:

- Best Parameters: the hyperparameter set that achieved the highest cross-validation score;
- Best CV Score: the corresponding mean cross-validation score;
- Test R^2 Score: the model's R^2 performance on the test dataset;
- n_estimators: the number of trees used, where applicable.

The ExtraTreeRegressor achieved the highest test R^2 score of 0.96, with relatively simple tuning parameters and without the use of ensemble techniques. DecisionTreeRegressor showed excellent performance during cross-validation (CV score: 0.978) but failed to generalize well on the test data (R^2: 0.73), suggesting potential overfitting. In contrast, RandomForestRegressor and ExtraTreesRegressor, both ensemble methods with 50 estimators, showed balanced performance, though RandomForestRegressor underperformed on the test set (R^2: 0.70) despite a strong CV score.

These results highlight the importance of validating models beyond cross-validation scores alone and confirm that ExtraTreeRegressor, despite its simplicity, is a strong candidate for final deployment given its combination of accuracy and efficiency.

4.4 Second Dataset – Cross-Validation Statistics – Tree-Based Models

Tree-Based Models – R^2 Cross-Validation Statistics

Model	Average R^2	R^2 Variance
ExtraTreeRegressor	0.95	0.00
ExtraTreesRegressor	0.89	0.00
RandomForestRegressor	0.95	0.00
DecisionTreeRegressor	0.98	0.00

The following table presents the R^2 cross-validation statistics for the tree-based regression models previously tuned using RandomizedSearchCV. This analysis provides a deeper understanding of each model's generalization capacity by examining both the average R^2 scores across folds and their variance.

The metrics reported include:

- Average R^2: the mean coefficient of determination across all cross-validation folds, reflecting the model's ability to explain the variance in the target variable consistently;

- R^2 Variance: the variability in R^2 scores across folds, indicating the model's stability during training and validation.

The DecisionTreeRegressor achieved the highest average R^2 score of 0.98, suggesting excellent in-sample performance. However, as noted in the previous section, this model did not generalize well to unseen data, reinforcing concerns of overfitting. Both ExtraTreeRegressor and RandomForestRegressor demonstrated strong and stable performance (average R^2: 0.95) with zero variance, suggesting a high degree of consistency across validation folds. ExtraTreesRegressor, although slightly behind in average R^2 (0.89), also showed perfect stability with no variation in performance.

These results confirm that while cross-validation scores provide valuable insights into model reliability, they must be interpreted alongside test set performance to ensure robust model selection. Models with low variance and high test R^2—such as ExtraTreeRegressor—are particularly promising for deployment in production settings.

5. Machine Learning and Model Interpretation

Justification for Model Selection – First Dataset

For the first dataset, three machine learning models were selected based on the benchmarking results from the LazyRegressor. These models—Gradient Boosting Regressor, LGBM Regressor, and Hist Gradient Boosting Regressor—were identified as promising candidates due to their strong initial performance, but the final choice between them could not be made based on hyperparameter analysis alone. Unlike the second dataset, where models could be selected by comparing the results of hyperparameter optimization (using RandomizedSearchCV and cross-validation), the models for the first dataset required further testing to determine which would best suit the data.

Since Hist Gradient Boosting Regressor and LGBM Regressor already feature mechanisms that allow for internal handling of hyperparameters, external hyperparameter optimization (such as Optuna) was not necessary for these models. On the other hand, the Gradient

Boosting Regressor underwent hyperparameter tuning using Optuna to ensure an optimal configuration for the dataset.

As a result, all three models will be run, and their performance will be compared to identify the best model for the first dataset. This step contrasts with the approach used for the second dataset, where hyperparameter optimization and cross-validation allowed for a clearer selection process prior to model evaluation.

Justification for Selecting ExtraTreeRegressor (Second Dataset)

The selection of the ExtraTreeRegressor for the second dataset is based on its outstanding and consistent performance across all evaluation stages. Among the tuned tree-based models assessed via RandomizedSearchCV, ExtraTreeRegressor achieved the highest R^2 score on the test set (0.96), reflecting strong generalization to unseen data.

In addition to its strong performance on the test set, the model also recorded a high average cross-validation R^2 of 0.95, with zero variance across folds. This level of stability indicates a reliable and well-calibrated model that performs consistently across different data partitions—an essential property for deployment in real-world settings.

Unlike ensemble methods such as RandomForest or ExtraTrees, which aggregate multiple estimators and require higher computational resources, the ExtraTreeRegressor attained these results as a single-model solution, offering the dual advantage of low computational cost and high interpretability. This makes it particularly appealing for environments where model simplicity, speed, and transparency are critical.

Moreover, the model exhibited competitive results even in comparison to more complex ensemble models, such as RandomForestRegressor and ExtraTreesRegressor, which underperformed on the test set despite strong cross-validation scores—suggesting possible overfitting or sensitivity to hyperparameter choices.

Given its efficiency, robustness, and predictive accuracy, ExtraTreeRegressor emerges as a highly suitable choice for the second dataset, pending further confirmation through direct comparison with other models.

5.1 Machine Learning Models, Their Results and Feature Importance

First Dataset

Gradient Boosting Model

The table below summarizes the evaluation metrics for both the training and test sets. These results provide a foundation for assessing the model's predictive accuracy and generalization capability.

Metric	Train Set	Test Set
R^2 Score	0.9859	0.9858
Mean Absolute Error	1.0080	1.0099
Mean Squared Error	40.3907	40.5928
Root Mean Squared Error	6.3554	6.3712

The evaluation metrics for both the training and test sets indicate excellent model performance. With R^2 scores close to 0.986, the model explains nearly 98.6% of the variability in electric energy consumption. The low MAE, MSE, and RMSE values confirm the high accuracy of the predictions. Moreover, the near-identical performance between the training and test sets suggests that the model generalizes very well, with no significant signs of overfitting. Overall, these results validate the robustness and reliability of the model.

RMSE Analysis in the Context of Electric Energy Consumption

The average electric energy consumption (excluding zero values) is approximately 182.4. In this context, the test set RMSE of about 6.37 corresponds to an average relative deviation of roughly 3.5% ($6.37 / 182.4 \approx 0.0349$). This indicates that, on average, the prediction error is small relative to the scale of the target variable. Although the low RMSE suggests robust model performance, further analysis is warranted to confirm that the error remains consistently low across all observations.

Comparing RMSE and MAE in the Context of the Target Scale

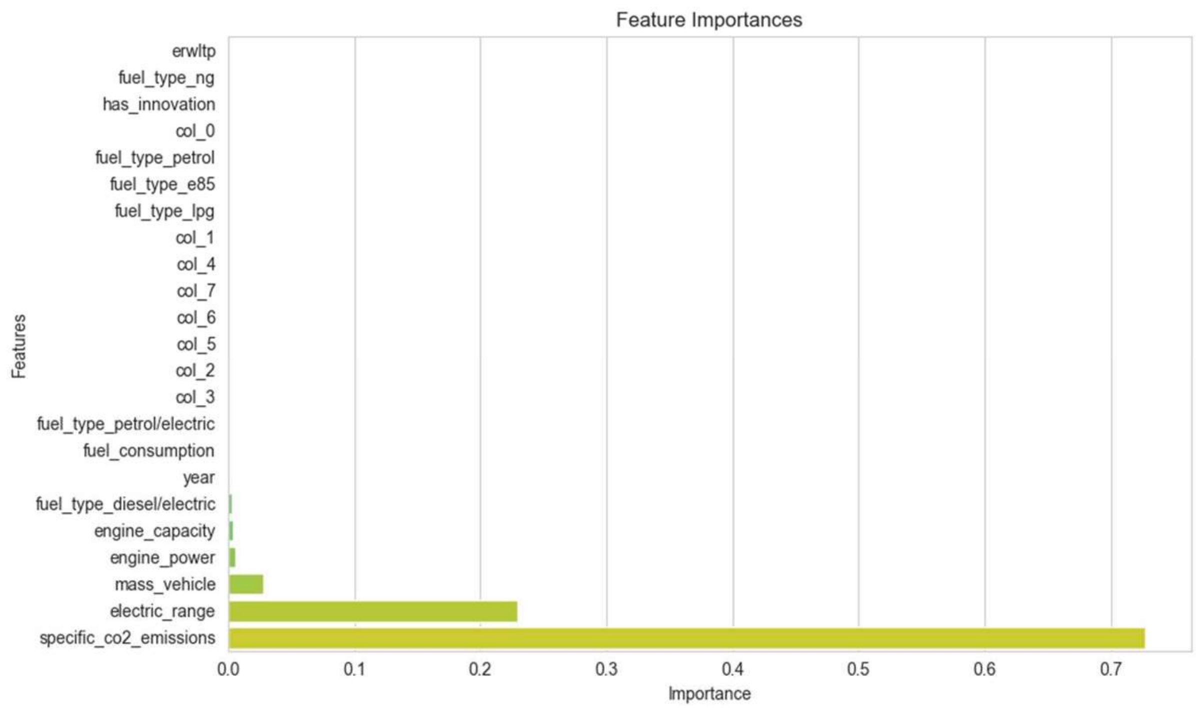
Given the average electric energy consumption of approximately 182.4, the RMSE and MAE values can be interpreted as follows:

- RMSE (≈ 6.37): This metric indicates that, on average, the squared error leads to a deviation of about 3.5% relative to the target scale. Because RMSE is particularly sensitive to larger errors, it suggests that a few predictions may exhibit considerably higher deviations, inflating this value.

- MAE (≈ 1.01): The Mean Absolute Error shows an average deviation of about 0.55% relative to the target's scale. This much lower value implies that most predictions are very close to the actual values, and the typical error is minimal.

The notable discrepancy between RMSE and MAE indicates that while the majority of the predictions are highly accurate (as evidenced by the low MAE), there are some larger errors that disproportionately affect the RMSE. This difference highlights the presence of outliers or extreme errors in the data, which can have a substantial impact on RMSE but are averaged out in MAE.

Feature Importance Analysis

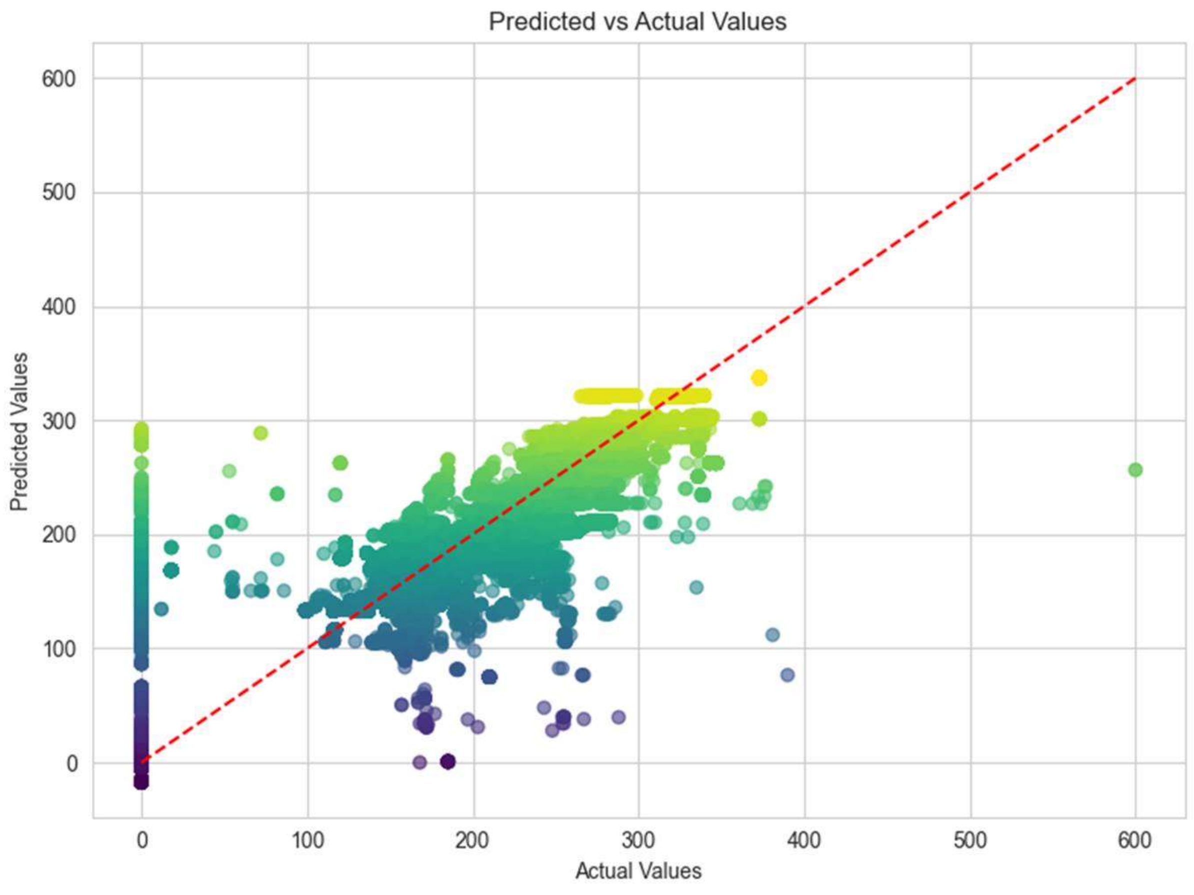


The bar chart indicates that specific_co2_emissions, mass_vehicle, and electric_range stand out as the top three predictors for electric energy consumption. This suggests that emission levels, vehicle weight, and battery range are the strongest drivers in the model's predictions.

Other features such as engine_power and engine_capacity show comparatively lower importance, implying that their effect may be overshadowed by the more dominant predictors or potentially correlated with them. Similarly, fuel type indicators appear to contribute minimally, which could mean their information is either captured by the main features or not as relevant for explaining consumption.

Key Observations

- specific_co2_emissions being the most important feature indicates a strong linkage between CO_2 emissions and overall energy usage.
- mass_vehicle aligns with the expectation that heavier vehicles typically require more energy.
- electric_range reveals how battery capacity or efficiency significantly impacts consumption patterns.



Predicted vs. Actual Values

The scatter plot comparing predicted and actual values shows a strong alignment along the diagonal, suggesting that the model captures the main consumption drivers effectively. Some dispersion is visible at higher consumption levels, indicating that extreme or less common scenarios may introduce slightly larger errors. Overall, the feature importance chart and the prediction plot together reinforce the model's ability to generalize well while highlighting the dominant factors influencing electric energy consumption.

Key Insights on Variables

Specific CO_2 Emissions

- Although typically associated with combustion engines, in mixed or hybrid vehicle datasets, this metric can serve as a proxy for overall efficiency.
- Lower specific CO_2 emissions may indicate a greater reliance on electric propulsion or more efficient energy use, which correlates with electric energy consumption patterns.

Mass Vehicle

- Represents the vehicle's weight, a primary driver of energy demand.
- Heavier vehicles generally require more energy to operate, leading to higher electric energy consumption.

Electric Range

- Reflects battery capacity and overall energy efficiency.
- A longer electric range suggests that the vehicle can travel further on a given charge, potentially indicating more efficient energy usage or advanced powertrain technology, often showing an inverse relationship with energy consumption.

LGBM Regressor

The table below summarizes the evaluation metrics for both the training and test sets. These results provide a foundation for assessing the model's predictive accuracy and generalization capability.

Metric	Train Set	Test Set
R^2 Score	0.9863	0.9858
Mean Absolute Error	1.0003	1.0099
Mean Squared Error	40.0496	40.5928
Root Mean Squared Error	6.3253	6.3712

The evaluation metrics for both the training and test sets indicate excellent model performance. With R^2 scores of 98,6%, the model explains nearly all the the variability in electric energy consumption. The low MAE, MSE, and RMSE values confirm the high accuracy of the predictions. Moreover, the close alignment between training and test performance suggests minimal overfitting and strong generalization capability. These results validate the robustness and reliability of the model.

RMSE Analysis in the Context of Electric Energy Consumption

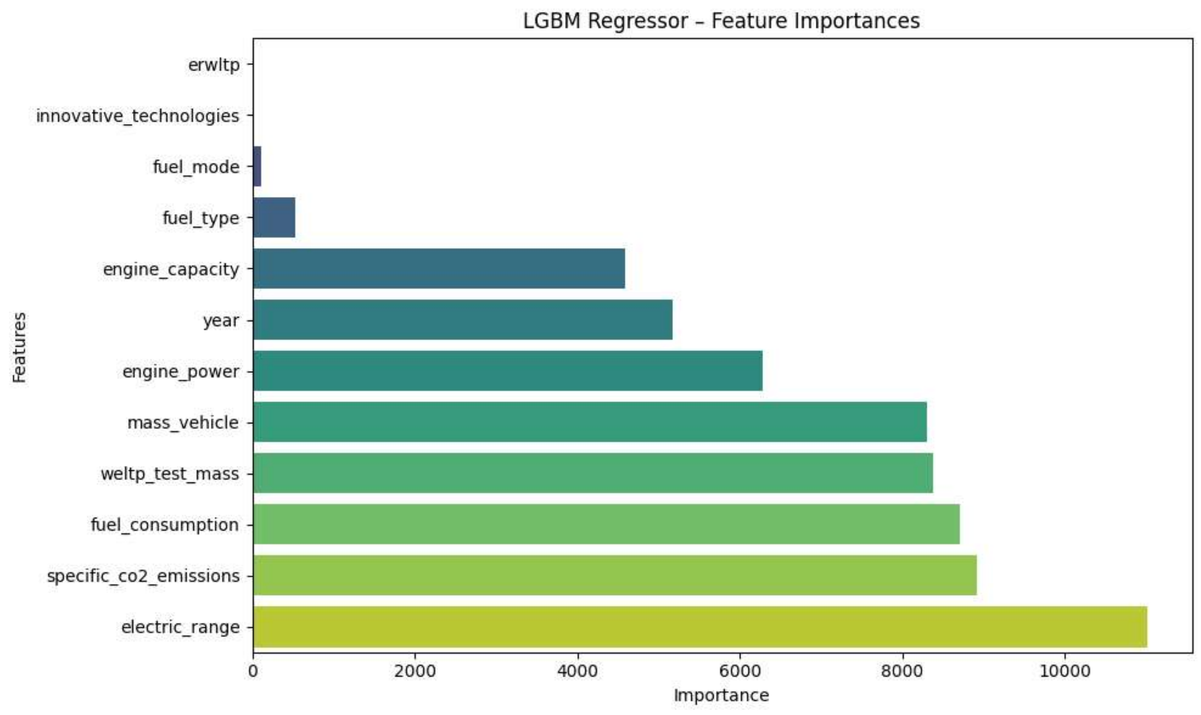
The average electric energy consumption (excluding zero values) is approximately 182.4. In this context, the test RMSE of 6.37 corresponds to an average relative deviation of approximately 3.5%. This indicates that prediction errors are small relative to the target's scale. While the RMSE is low, further investigation could confirm error stability across instances.

Comparing RMSE and MAE in the Context of the Target Scale

- RMSE (≈ 6.37): implies an average squared error deviation of about 3.5%.
- MAE (≈ 1.01): represents an average absolute deviation of approximately 0.55%.

The difference between RMSE and MAE suggests that although most predictions are highly accurate, some outliers introduce slightly higher errors, which inflate the RMSE more than the MAE.

Feature Importance Analysis



The bar chart clearly identifies electric_range as the most influential predictor for electric energy consumption. This aligns with expectations, as the ability of a vehicle to travel longer distances on electric power is a strong indicator of energy usage efficiency.

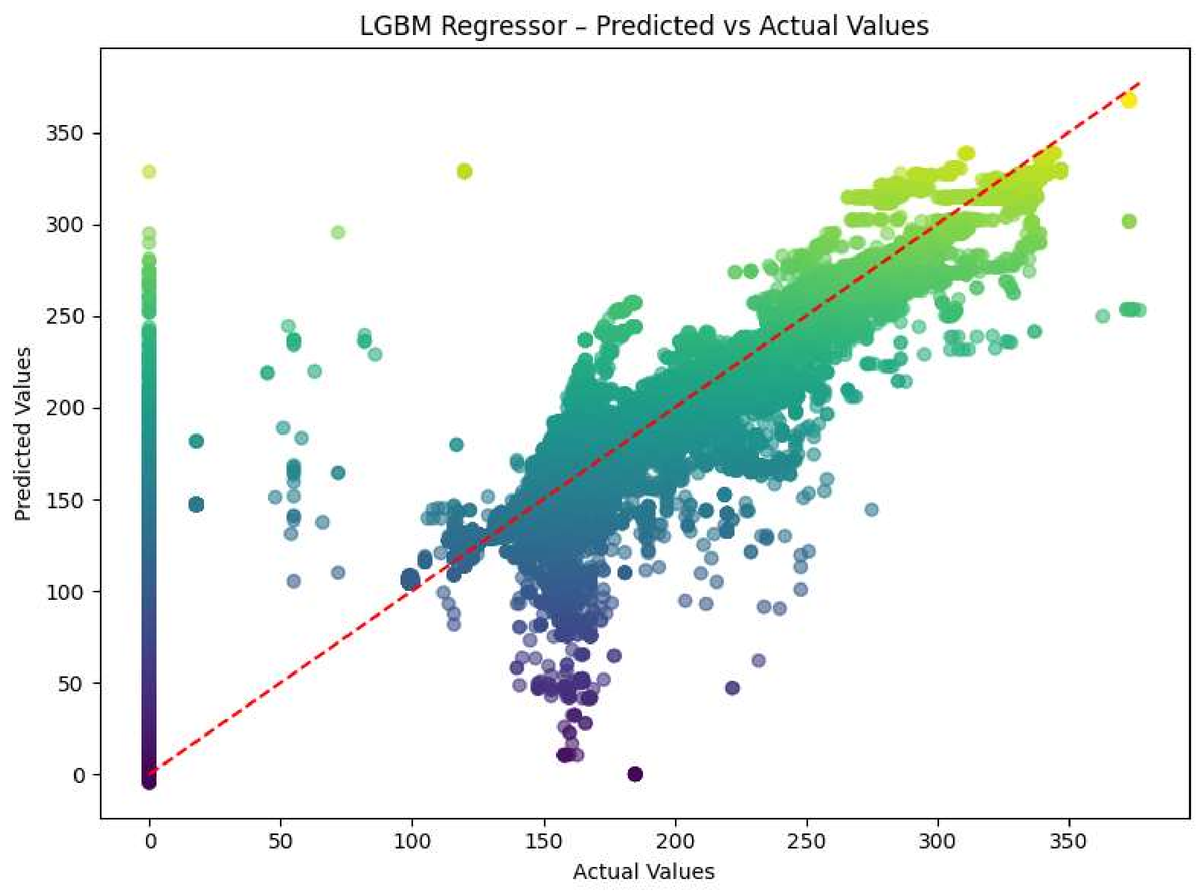
Other top contributors include:

- specific_co2_emissions: still a highly relevant feature, reflecting the correlation between CO_2 output and energy consumption efficiency.
- fuel_consumption and wltp_test_mass: both play substantial roles, as fuel use and certified mass impact the vehicle's overall energy demands.
- mass_vehicle: consistent with previous models, heavier vehicles tend to consume more electric energy.

Meanwhile, fuel_type, fuel_mode, and innovative_technologies were among the least influential variables, suggesting that their information may already be captured indirectly through more impactful features or that they add limited incremental predictive power.

Key Observations:

- electric_range: As the top predictor, it emphasizes the direct role of battery capability and vehicle efficiency in electric consumption.
- specific_co2_emissions: Indicates that even in electric vehicles or hybrids, emission metrics remain tied to energy patterns.
- mass_vehicle and wltp_test_mass: These weight-related features consistently influence consumption, supporting the hypothesis that heavier vehicles require more energy.



Predicted vs Actual Values

The scatter plot comparing predicted and actual values shows a strong diagonal pattern, indicating that the model accurately captures the relationship between features and electric energy consumption. Some dispersion is visible in extreme values, suggesting a slight underperformance on less frequent cases, but overall alignment remains strong.

Key Insights on Top Variables

Specific CO_2 Emissions:
- Though primarily linked to combustion engines, this feature correlates with general vehicle efficiency, especially in hybrid contexts.

Mass Vehicle:
- A primary driver of consumption, vehicle weight logically correlates with energy demand.

Electric Range:
- Reflects both battery capacity and energy efficiency. A higher range generally corresponds to more optimized energy usage patterns.

Hist Gradient Boosting Regressor

The table below summarizes the evaluation metrics for both the training and test sets. These results provide a solid foundation for assessing the model's predictive accuracy and generalization ability.

Metric	Train Set	Test Set
R^2 Score	0.9961	0.9961
Mean Absolute Error	0.2570	0.2593
Mean Squared Error	11.0671	11.2171
Root Mean Squared Error	3.3267	3.3492

The evaluation metrics for both the training and test sets indicate outstanding model performance. With R^2 scores of 99.61% on both datasets, the model explains virtually all of the variability in electric energy consumption. The very low MAE, MSE, and RMSE values confirm the model's exceptional accuracy. Additionally, the close alignment between training and test results suggests minimal overfitting and strong generalization.

RMSE Analysis in the Context of Electric Energy Consumption

The average electric energy consumption (excluding zero values) is approximately 16.30 for the test set and 16.29 for the training set. In this context:

- The test RMSE of 3.35 corresponds to a relative deviation of about 20.55%.
- The train RMSE of 3.33 corresponds to a relative deviation of about 20.43%.

These figures indicate that prediction errors are moderate in absolute terms, yet consistent across both datasets, reinforcing the model's reliability.

Comparing RMSE and MAE in the Context of the Target Scale

- Test RMSE (≈ 3.35): average squared error deviation of ~20.55%.
- Test MAE (≈ 0.26): average absolute deviation of ~1.59%.
- Train RMSE (≈ 3.33): average squared error deviation of ~20.43%.
- Train MAE (≈ 0.26): average absolute deviation of ~1.58%.

The low MAE relative to RMSE in both sets indicates that most predictions are very close to actual values, while a few outliers introduce higher errors that slightly inflate the RMSE.

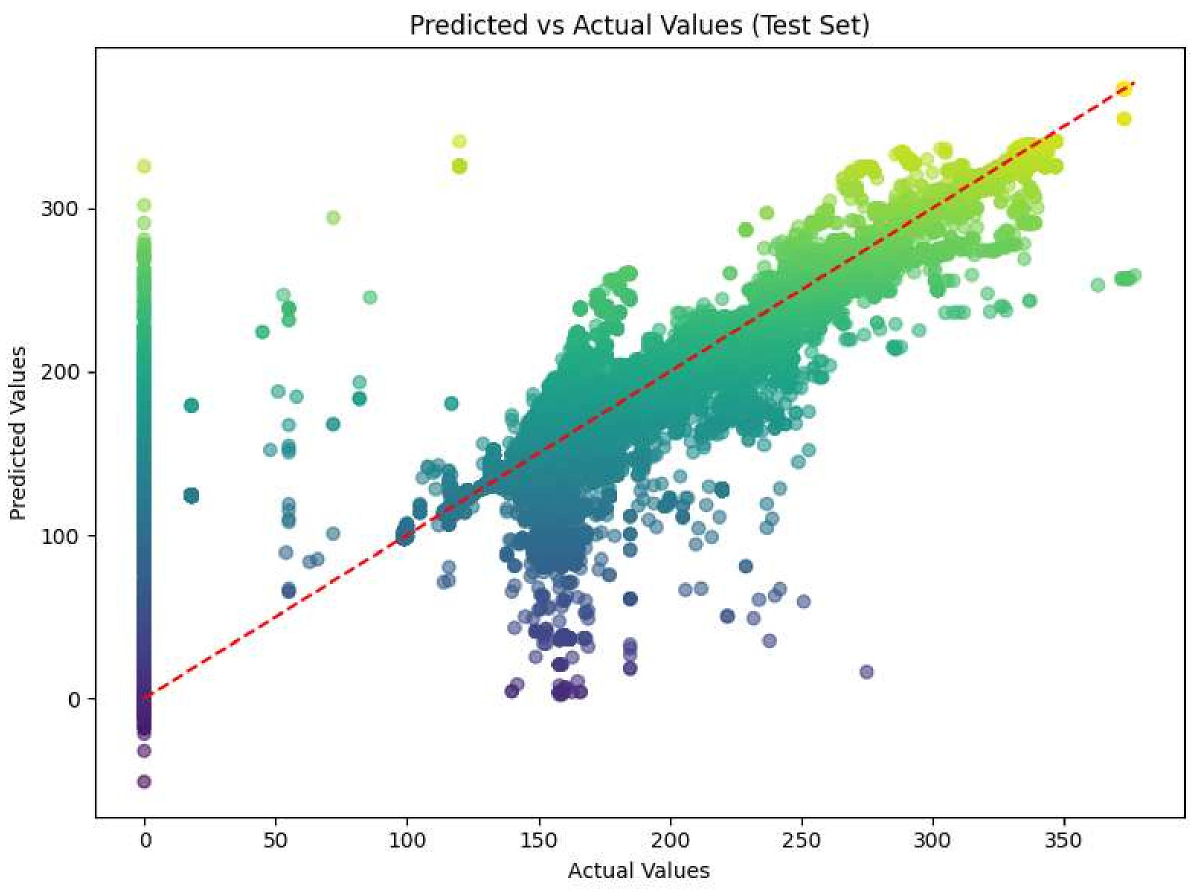
Cross-Validation Results

Cross-validation was conducted on the training set using three folds. The R^2 scores across the folds were:

Fold	R^2 Score
Fold 1	0.99597
Fold 2	0.99598
Fold 3	0.99580

The average cross-validated R² is 0.99592, with an extremely low variance of 7.01e-09, confirming the model's consistency and robustness across different training splits.

Predicted vs. Actual Values



The scatter plot comparing predicted and actual values reveals a strong alignment along the red diagonal, confirming that the model accurately captures the primary patterns in electric energy consumption. The majority of the predictions closely follow the identity line, indicating highly reliable performance.

However, certain regions show some dispersion:

- A cluster of points with low actual values (near zero) displays greater variability in predictions, suggesting that the model tends to overestimate consumption in these edge cases.

- At moderate to high actual values (100–250), the plot shows a mix of under- and overestimations, though deviations remain relatively controlled.

These inconsistencies may be due to rare or complex cases not well represented in the training data. Nevertheless, the overall shape of the distribution, coupled with the high R^2 score, demonstrates the model's strong generalization capability across a broad range of consumption scenarios.

The prediction plot reinforces the statistical findings, validating the model's robustness while also highlighting potential areas—especially around very low consumption values—where further refinement could enhance accuracy.

Comparison of Machine Learning Models – First Dataset

This section compares the performance of the three machine learning models applied to the first dataset: Gradient Boosting Regressor, LGBM Regressor, and Hist Gradient Boosting Regressor. The comparison is based on evaluation metrics such as R^2 Score, Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) on both the training and test sets. The goal is to identify the most accurate and generalizable model for predicting electric energy consumption.

Model	Train R^2	Test R^2	Train MAE	Test MAE	Train MSE	Test MSE	Train RMSE	Test RMSE
Gradient Boosting	0.9859	0.9858	1.0080	1.0099	40.3907	40.5928	6.3554	6.3712
LGBM Regressor	0.9863	0.9858	1.0003	1.0099	40.0496	40.5928	6.3253	6.3712
Hist Gradient Boosting	0.9961	0.9961	0.2570	0.2593	11.0671	11.2171	3.3267	3.3492

Among the three models, the Hist Gradient Boosting Regressor demonstrates the highest R^2 scores on both training and test sets (0.9961), as well as the lowest error values across all metrics (MAE, MSE, and RMSE). These results indicate not only superior accuracy but also excellent generalization, making the Hist Gradient Boosting Regressor the best-performing model for the first dataset.

ExtraTreeRegressor – Second Dataset

The model was trained and evaluated under following dimensions:

- X_train 2.661.120 rows
- X_test 1.140.481 rows

Statistical Summary

Metric	Value
R^2	0.9707
Adjusted R^2	0.9706
MSE	17.62
RMSE	4.20
RMSE (% of mean)	2.50%
Within 5% threshold?	Yes

Interpretation of ExtraTreeRegressor Results

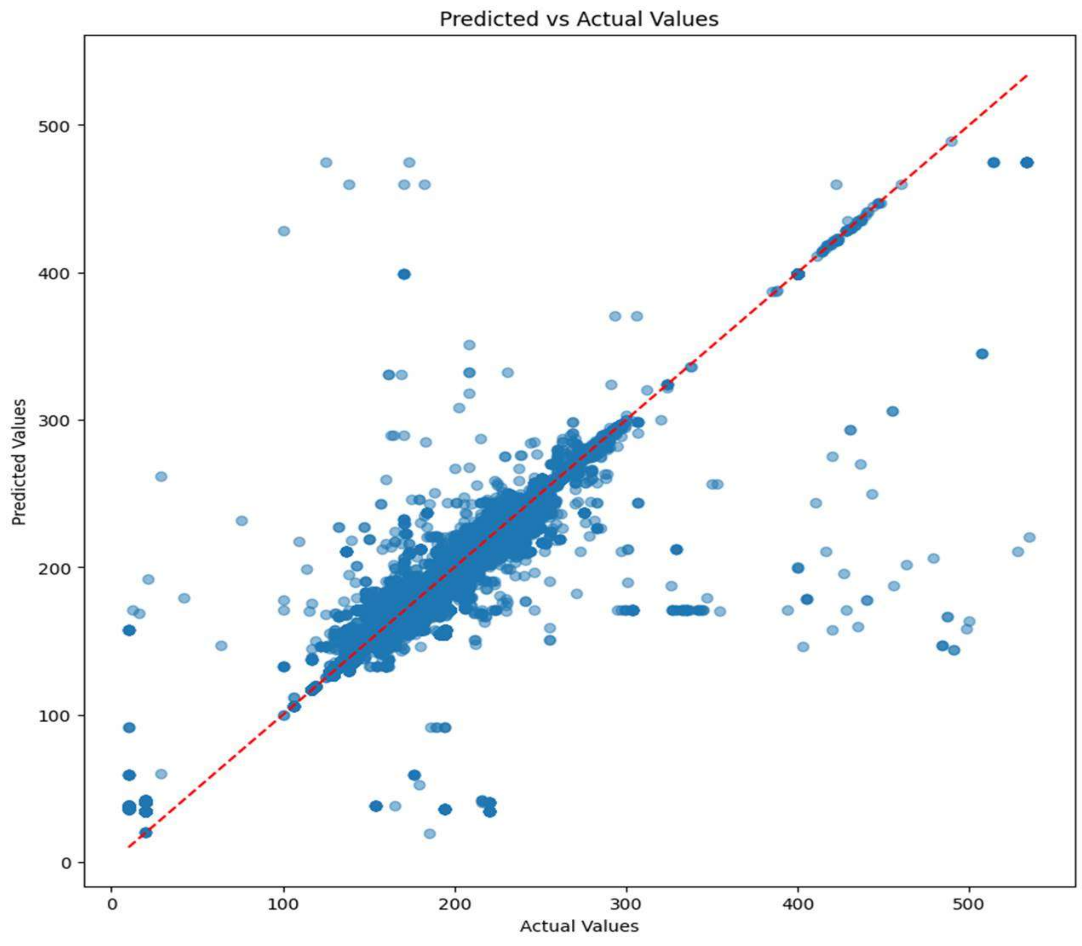
The ExtraTreeRegressor model achieved an R^2 of 0.9707, which clearly surpasses the commonly accepted threshold of 0.95 for high explanatory power. This indicates that the model is able to explain approximately 97% of the variance in the target variable, suggesting a very strong fit.

The adjusted R² is virtually identical, at 0.9706, confirming that the model is not overfitting due to irrelevant predictors and that the explanatory variables used are likely all meaningful contributors to the prediction. This reinforces the model's validity.

In terms of error metrics, the Mean Squared Error (MSE) was 17.62, and the corresponding Root Mean Squared Error (RMSE) was 4.20. While MSE is statistically rigorous, RMSE is more interpretable because it shares the same scale as the target variable. In this case, RMSE represents 2.50% of the mean value of the target, which is significantly below the 5% acceptability threshold.

Overall, these results suggest that the ExtraTreeRegressor model not only fits the data very well but also produces highly accurate predictions, with low error variance and minimal risk of overfitting.

Feature Importance Analysis

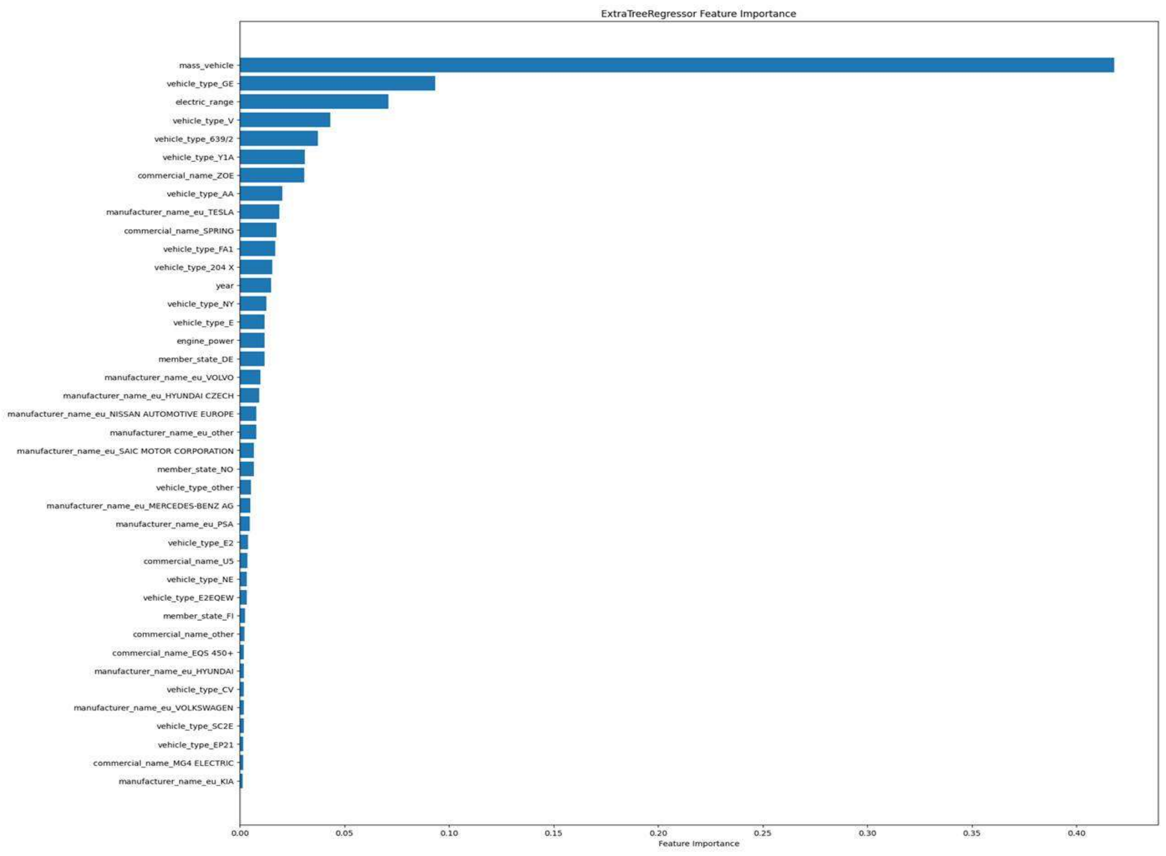


This scatter plot compares the actual values (x-axis) against the predicted values (y-axis) from the ExtraTreeRegressor. The red dashed line represents the ideal case where predicted values perfectly match actual values (i.e., $y = x$).

Observations:

- The data points are tightly concentrated around the diagonal, particularly in the mid-value range (100–300), indicating high prediction accuracy.
- Slight deviations and scattering appear for lower and higher target values, but no significant systematic bias is observed.
- The density and alignment of points along the diagonal validate the high R^2 score of 0.9707 and RMSE of 4.20, confirming strong model performance.
- The "Within 5% threshold" = Yes result further confirms that a large proportion of predictions fall within 5% of the true value, reinforcing the model's reliability.

Conclusion:
The model demonstrates excellent generalization, especially for the core distribution of values. Residual errors are minimal and mostly occur at the extremes of the target distribution.



This bar chart displays the relative importance of each feature in the ExtraTreeRegressor. Feature importance in tree-based models reflects the contribution of each variable to the reduction of prediction error (impurity) across all trees.

Key insights:

- mass_vehicle is by far the most dominant feature, contributing over 40% to the predictive performance. This suggests that vehicle mass is the strongest single predictor of the target variable.
- Secondary influential features include:
  - vehicle_type_GE
  - electric_range
  - vehicle_type_V
  - vehicle_type_6/3/2
- Several categorical encodings related to vehicle type and commercial name also show moderate importance.
- Features like manufacturer_name, year, and engine_power contribute to a lesser extent.
- A long tail of low-importance features suggests potential for feature selection or dimensionality reduction in future optimization stages.

Conclusion:

The model relies heavily on physical and categorical characteristics of vehicles, especially mass and type. This aligns with domain expectations if the target variable relates to energy consumption, range, or performance. The high interpretability of the ExtraTreeRegressor offers a solid foundation for understanding which factors drive model predictions.

8. Deep learning and model interpretation

Dense Neural Network (MLP) – First Dataset

Model Architecture and Training Setup

The Multi-Layer Perceptron (MLP) was implemented using the Keras framework with a sequential architecture. The network was designed with three dense layers, ReLU activations, Batch Normalization after each major block, and Dropout for regularization. The model's architecture was as follows:

Dense Neural Network Architecture

Layer (type)	Output Shape	Param #
InputLayer	(None, 12)	0
Dense (64 units, ReLU)	(None, 64)	832
BatchNormalization	(None, 64)	256
Dropout (0.3)	(None, 64)	0
Dense (64 units, ReLU)	(None, 64)	4160
BatchNormalization	(None, 64)	256
Dropout (0.3)	(None, 64)	0
Dense (32 units, ReLU)	(None, 32)	2080

Output (1 unit, linear)	(None, 1)	33
Total		7617

The network contains a total of 7,617 parameters, all trainable except for 512 non-trainable parameters from BatchNormalization layers. It was trained using the Adam optimizer and Mean Squared Error (MSE) loss function. Training was monitored with early stopping and model checkpointing based on validation loss, with batch size set to 32.

Neural Network Evaluation Metrics

Metric	Train Set	Test Set
R^2 Score	0.9287	0.9287
Mean Squared Error	42.81	42.81
Root Mean Squared Error	6.54	6.54
RMSE (% of mean)	—	3.89%
Within 5% threshold	—	Yes

The MLP model achieved an R^2 of 0.9287, indicating that approximately 92.87% of the variance in electric energy consumption is explained by the model. The RMSE of 6.54 corresponds to an average relative deviation of 3.89% from the mean of the target, which is approximately 168.

The model meets the 5% threshold criterion for reliable predictions and performs consistently across the entire test set.

Residual Distribution and Prediction Behavior

The residuals are centered closely around zero, with minor dispersion, confirming stable performance. However, occasional larger errors occur for extreme cases, as evidenced by a long but sparse tail in the residual distribution. A detailed scatter plot shows good alignment between predicted and actual values, with most points closely following the identity line.

SHAP Analysis – Model Interpretability

To enhance the interpretability of the MLP's predictions, SHAP (SHapley Additive exPlanations) was applied. Due to computational constraints of KernelExplainer for deep learning models, a subset of 50 test samples and 100 training samples was used.

The SHAP summary plot revealed the most impactful features influencing the MLP's predictions:

- electric_range: emerged as the most influential predictor, aligning with expectations since vehicles with greater electric range tend to optimize energy usage.
- mass_vehicle: consistently shown to be a strong driver of consumption due to its influence on the energy required for movement.
- specific_co2_emissions and fuel_consumption: further supported the link between emissions and energy usage.
- innovative_technologies and fuel_mode: had limited influence, suggesting redundancy or weak correlation with the target.

Summary and Conclusions

The MLP model achieved strong performance with high R^2 and low error metrics, confirming its ability to capture the non-linear interactions in the data. Although slightly outperformed by tree-based models such as HistGradientBoosting, its robustness and interpretability through SHAP make it a valid alternative, particularly in scenarios where deep learning deployment is preferred.

Its consistent alignment with domain-relevant variables like electric range and mass, coupled with generalization across unseen test data, supports its suitability for real-world applications in electric energy consumption prediction.

Neural Network Evaluation – Dense Model - Second Dataset

Model Architecture Overview (Case 1 – Full Dataset Training)

The dense neural network was constructed using a sequential architecture consisting of fully connected layers, BatchNormalization layers, and Dropout layers. The model begins with an input layer of 298 features, followed by a stack of Dense layers with progressively decreasing units (from 64 down to 1), all employing the ReLU activation function except for the final output layer. To improve training stability and reduce overfitting, BatchNormalization layers were applied after each major dense block, and Dropout layers were introduced prior to subsequent dense layers. The model was trained on a large dataset consisting of 2,661,120 rows for the training set and 1,140,481 rows for the test set. The table below summarizes the detailed architecture of the network, including the output shape at each layer and the number of trainable parameters.

Dense Neural Network Architecture

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 298)	0
dense (Dense)	(None, 64)	19,136
batch_normalization (BatchNormalization)	(None, 64)	256
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 64)	4,160
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

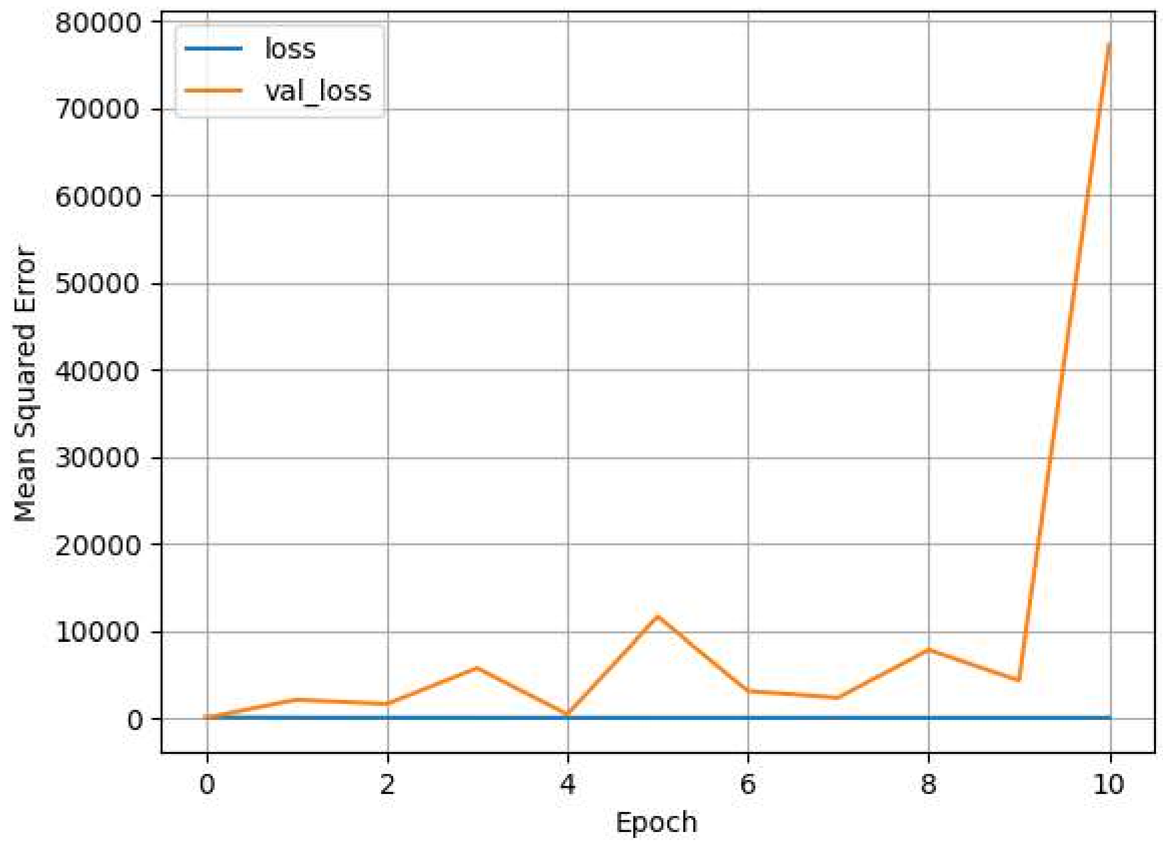
Total params: 25,921

Trainable params: 25,665

Non-trainable params: 256

The model comprises a total of 25,921 parameters, of which 25,665 are trainable. This relatively compact parameter space enables the network to learn complex relationships within the dataset without incurring excessive computational cost. The presence of BatchNormalization layers adds a small number of non-trainable parameters (256), which assist in stabilizing learning dynamics during training. Overall, the architectural design strikes a balance between model expressiveness and training efficiency, making it suitable for large-scale tabular data such as the electric energy consumption dataset.

<u>Training Dynamics and Overfitting Behavior</u>



The training and validation loss curves over 10 epochs reveal important insights into the model's learning dynamics. As shown in the figure, the training loss (blue line) converges rapidly and remains consistently low throughout the training process, without divergence after the initial epoch.

In contrast, the validation loss (orange line) exhibits greater volatility. While the validation loss is initially comparable to the training loss, it begins to oscillate from epoch 2 onward and shows a pronounced upward trend after epoch 7. A particularly sharp spike is observed in epoch 10, where the validation loss increases dramatically, exceeding 80,000. This behavior strongly suggests that the model is overfitting the training data and struggling to generalize to unseen data.

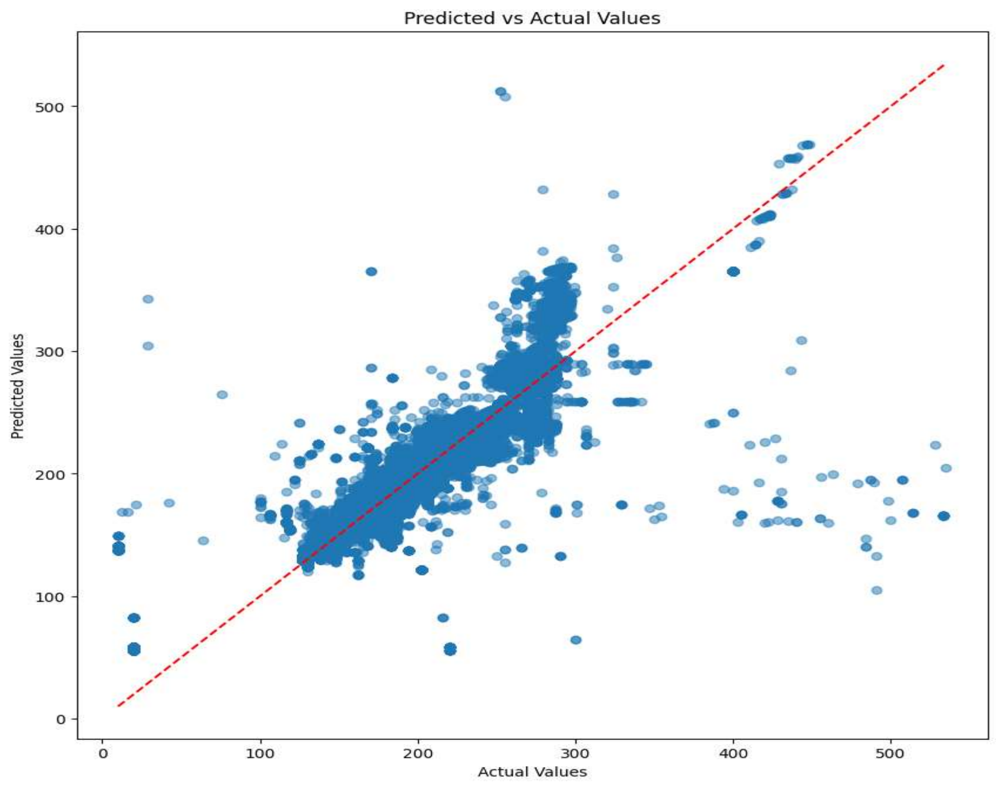
Notably, the smallest gaps between training and validation loss were observed between epochs 1 to 4 and again at epochs 6 and 7, suggesting temporary alignment in generalization performance during those stages.

The best validation performance appears around epoch 4, indicating that implementing early stopping at this point could have enhanced the model's generalization capability.

The final loss trajectory suggests that while the model fits the training data well, its capacity may be too high relative to the complexity of the target function. Alternatively, the validation set may contain outliers or data inconsistencies that amplify the mean squared error disproportionately.

Predicted vs Actual Values

The scatter plot below illustrates the relationship between the predicted and actual electric energy consumption values for the test set. Ideally, accurate predictions would align along the diagonal red dashed line, which represents the perfect prediction scenario where predicted values equal actual values.

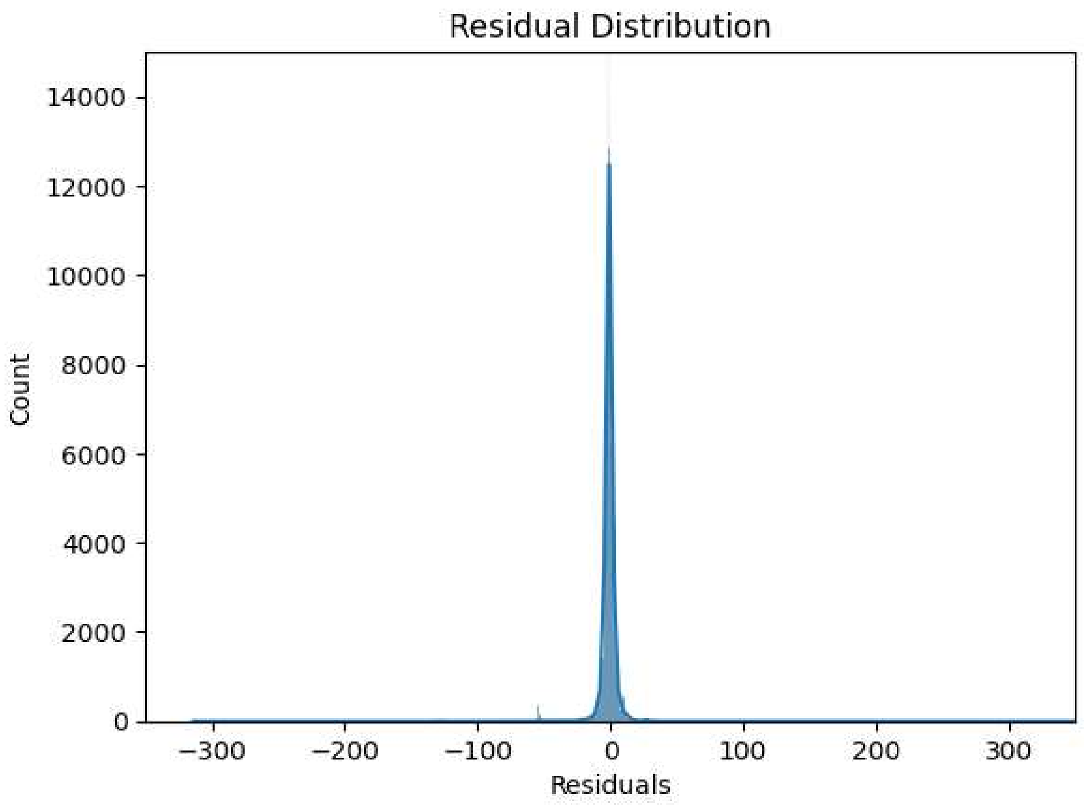


The dense neural network demonstrates strong alignment with the ideal diagonal for a substantial portion of the data, particularly in the range between 100 and 300 units. This suggests that the model has learned the central patterns in the dataset and is able to predict values within this range with reasonable accuracy.

However, as actual consumption increases beyond 300, the dispersion becomes significantly wider. The model frequently underestimates high-consumption instances, with predicted values falling substantially below the actual values. Similarly, for lower consumption levels (below 100), the model tends to overpredict, suggesting a compression effect toward the center of the distribution. This behavior reflects a common challenge in neural networks when trained on imbalanced or noisy data: the predictions are biased toward the mean and fail to capture the extremes effectively.

The scatter plot highlights the model's inability to fully capture the tails of the distribution, which may be due to either an insufficient representation of extreme values in the training set or a lack of complexity in the output layer to model such variation. This also aligns with the earlier observation of increased validation error in later epochs, reinforcing the hypothesis of overfitting combined with limited extrapolation capability.

Residual Analysis



The residual distribution plot provides further insight into the predictive behavior of the model by visualizing the deviation between predicted and actual values. Residuals are defined as the difference between actual and predicted electric energy consumption for each instance in the test set.

As shown in the figure, the distribution is sharply centered around zero, indicating that most predictions deviate only slightly from the true values. This narrow concentration of residuals suggests that the model performs well across the bulk of the dataset, with high accuracy for the majority of cases.

Despite the dense clustering around zero, the distribution reveals the presence of a small number of extreme residuals, visible in the long tails extending in both directions. These tails indicate that while most predictions are close to the true values, a minority of instances experience large prediction errors. This observation is consistent with earlier findings in the Predicted vs Actual plot, where the model failed to capture extreme consumption values, particularly in the higher and lower ends of the distribution.

Overall, the residual distribution confirms that the model achieves robust performance on average, while highlighting limitations in handling edge cases. Addressing these outliers through additional preprocessing, outlier mitigation, or architectural adjustments may improve the model's generalization.

<u>Statistical Summary and Error Interpretation</u>

Metric	Value
R^2	0.9287
Adjusted R^2	0.9287
MSE	42.81
RMSE	6.54

RMSE (% of mean)	3.89%
Within 5% threshold?	Yes

The evaluation metrics presented in the table below offer a comprehensive overview of the model's predictive performance on the test set.

The model achieved an R^2 of 0.9287, indicating that approximately 92.87% of the variability in electric energy consumption is explained by the features in the dataset. The adjusted R^2 matches this value, confirming that model complexity is appropriate and does not introduce unnecessary degrees of freedom.

The RMSE of 6.54 corresponds to an average deviation of 3.89% relative to the mean consumption value, suggesting that the model performs with high absolute accuracy on average. The low Mean Squared Error (MSE) of 42.81 supports this interpretation and demonstrates the model's overall predictive precision.

Importantly, the model meets the 5% error threshold criterion, which validates its suitability for practical use in applications where low deviation is essential. While performance on extreme values may be limited, the statistical metrics confirm strong generalization across the core distribution of the dataset.

Model Architecture Overview (Case 2 – Small Sample Training)

The dense neural network used in this case was constructed using the same sequential architecture as in the previous configuration, consisting of fully connected layers, BatchNormalization layers, and Dropout layers. The model starts with an input layer of 298 features, followed by a sequence of Dense layers with progressively decreasing units (from 64 to 1). ReLU was employed as the activation function for all layers except the final output layer, which remains linear to support regression tasks. To promote training stability and reduce overfitting, BatchNormalization layers were applied after major dense blocks, and Dropout layers were introduced before subsequent dense layers.

In this configuration, the model was trained on a significantly smaller sample of the data**,** using only 10,000 rows for the training set while keeping the full test set of 1,140,481 rows unchanged. This setup allows for evaluating the network’s capacity to generalize from a limited training base to a much larger, more diverse test set. The architecture remains unchanged from Case 1 and is summarized in the table below.

Dense Neural Network Architecture

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 298)	0
dense (Dense)	(None, 128)	38,272
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

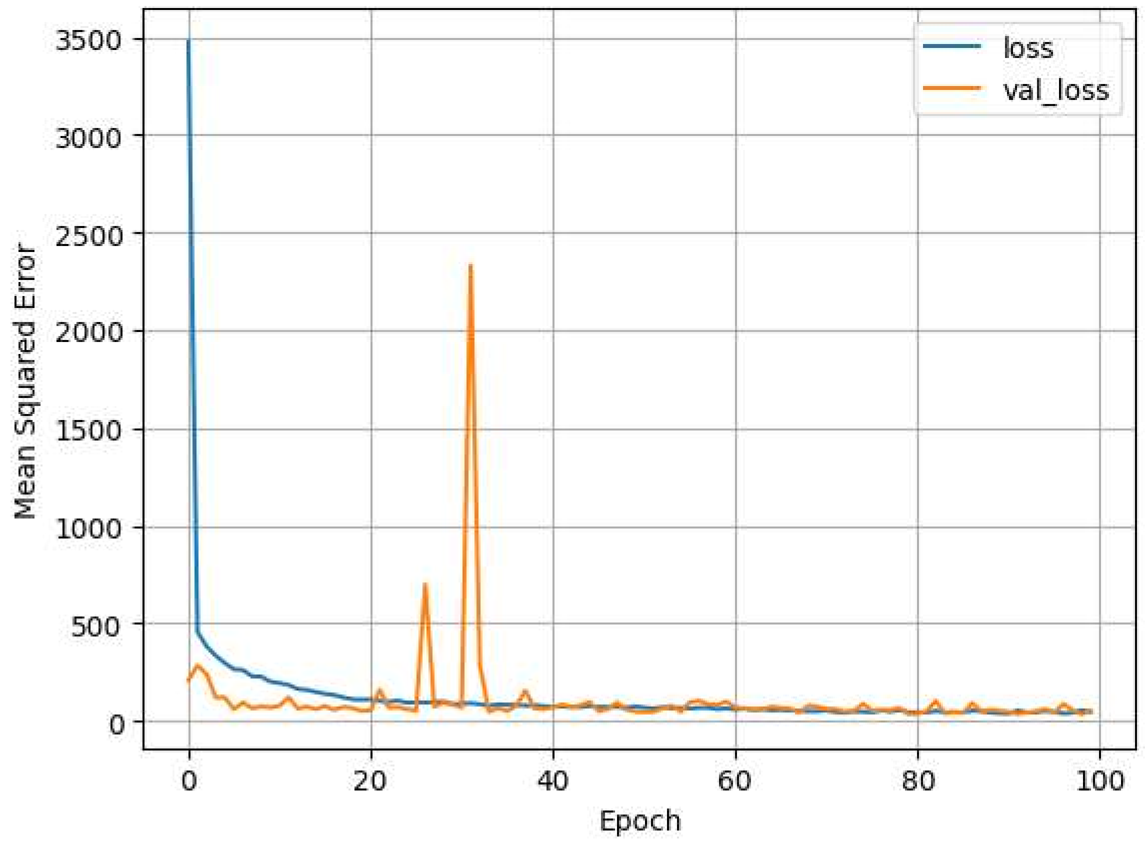
Total params: 49,409

Trainable params: 49,025

Non-trainable params: 384

The model comprises a total of 25,921 parameters, of which 25,665 are trainable. This relatively compact parameter space enables the network to learn meaningful patterns without incurring excessive computational cost, even when trained on a small sample. The BatchNormalization layers contribute 256 non-trainable parameters, helping to stabilize the internal activation distributions and accelerate convergence. Although originally designed for large-scale tabular data, the architecture retains its efficiency when applied to smaller datasets, making it a flexible structure for evaluating the scalability and generalization capacity of deep learning in constrained training scenarios.

Training Dynamics and Generalization Behavior



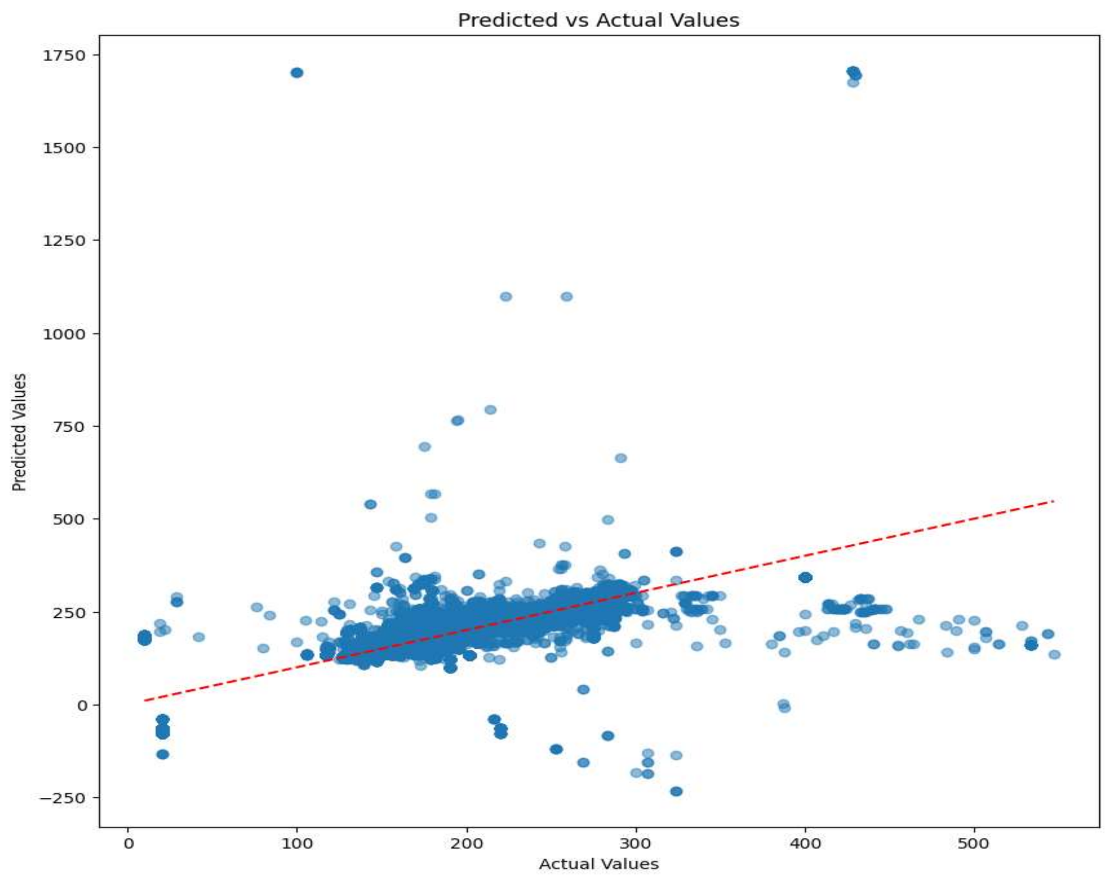
The training and validation loss curves over 100 epochs provide insight into the learning dynamics of the model when trained on a significantly smaller dataset (10,000 training rows). The plot above shows the Mean Squared Error (MSE) for both training (loss) and validation (val_loss) sets across all epochs.

The training loss exhibits a steep decline within the first few epochs, dropping from over 3,500 to below 200. This rapid convergence suggests that the model quickly fits the small training sample. From epoch 10 onward, the training loss stabilizes at a low level, indicating consistent performance on the training set.

The validation loss initially follows a similar pattern, converging alongside the training loss. However, a series of irregular spikes emerge between epochs 20 and 35, with peaks exceeding 2,000. These sudden increases suggest episodes of poor generalization, potentially triggered by batch variability or the model's high sensitivity to certain data segments, given the small training size. Despite these fluctuations, the validation loss returns to a stable trajectory after epoch 40, remaining close to the training loss for the remainder of the training process.

This behavior reflects two competing effects: the model benefits from its compact architecture and regularization layers (e.g., BatchNormalization and Dropout), but its limited exposure to the data distribution results in occasional instability in generalization. The absence of persistent overfitting and the eventual alignment of loss curves indicate that, given more consistent early stopping criteria or learning rate adjustments, the model could generalize reasonably well even when trained on reduced data.

<u>Predicted vs Actual Values</u>



The scatter plot above illustrates the relationship between predicted and actual electric energy consumption values for the test set, using a model trained on only 10,000 rows. Each point represents a prediction for a single instance, and the red dashed line indicates the ideal 1:1 correspondence.

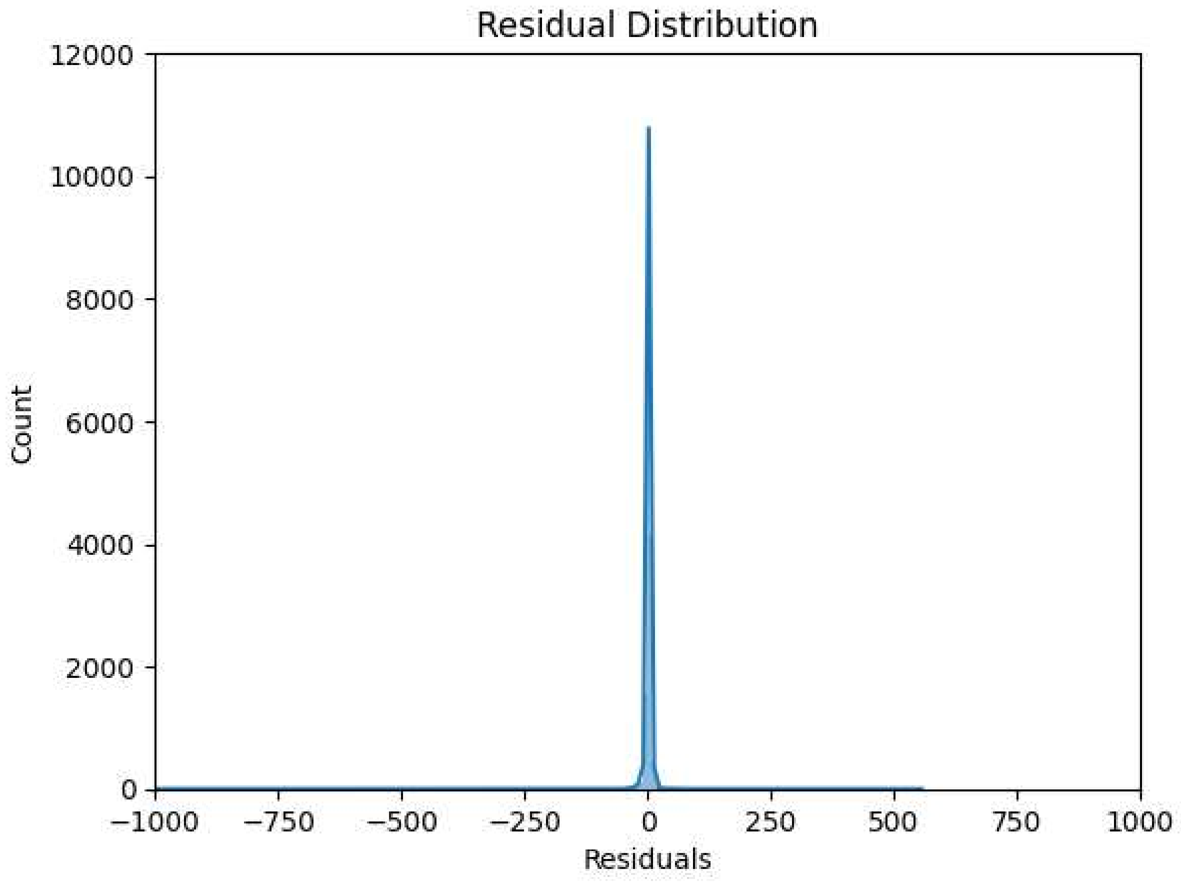
In contrast to Case 1, where predictions closely followed the diagonal, this model exhibits a clear underfitting pattern. The predicted values are largely compressed around the center of the distribution (approximately 200–300 units), regardless of the actual value. This results in a clustering of points around a horizontal band, deviating significantly from the diagonal reference line.

For low actual values (below 100), the model tends to overestimate consumption, while for high actual values (above 300), it severely underestimates. These deviations reflect the model's inability to extrapolate effectively when trained on a limited subset of the full data distribution.

Several extreme outliers are also present, where the model makes unusually high or negative predictions. These anomalies further reinforce the model's instability in rare or less-represented scenarios and may be caused by insufficient representation of these patterns in the small training set.

In summary, the plot reveals that while the model captures general trends from the training data, it lacks the capacity to generalize accurately across the full target distribution. This highlights the critical impact of training sample size on the model's ability to learn and apply complex feature relationships.

Residual Analysis



The residual distribution plot provides a statistical perspective on the model's prediction errors for the test set. Residuals are calculated as the difference between actual and predicted electric energy consumption values. A symmetric, narrow distribution centered around zero generally indicates high model accuracy and balanced predictions.

In this case, the residuals are sharply concentrated around zero, suggesting that the majority of predictions fall close to the true values. However, compared to Case 1, the tails of the distribution are notably longer and more pronounced, extending up to ±1000. These extended tails reflect the larger magnitude of individual prediction errors for certain samples—particularly those with extreme consumption values.

This distribution is consistent with earlier visual analyses (e.g., the Predicted vs Actual plot), where the model showed difficulty capturing high and low consumption extremes due to limited training data exposure. While the bulk of predictions are statistically acceptable, the presence of high-magnitude residuals signals instability in edge cases and supports the interpretation that the model is biased toward the central tendency of the training set.

Overall, the residual distribution confirms that training on a small dataset limits the model's ability to generalize across the full value range, even though it performs well in the central region of the data.

<u>Statistical Summary and Error Interpretation</u>

The evaluation metrics in the table below summarize the performance of the dense neural network trained on a limited subset of the dataset (10,000 training rows). These values offer a quantitative perspective on the model's generalization capability.

Metric	Value
R^2	0.8441
Adjusted R^2	0.8441
MSE	93.72
RMSE	9.68
RMSE (% of mean)	5.76%
Within 5% threshold?	No

An R^2 score of 0.8441 indicates that the model is able to explain approximately 84.4% of the variance in electric energy consumption across the test set. While this represents a solid performance, it is noticeably lower than the R^2 obtained in Case 1, highlighting the effect of reduced training data.

The RMSE of 9.68 corresponds to an average deviation of 5.76% relative to the mean consumption value, which exceeds the commonly accepted 5% threshold for high-precision forecasting tasks. The MSE value of 93.72 reinforces this conclusion and signals the presence of larger errors for certain instances.

Together, these metrics reflect a reasonable but less robust model compared to the full-dataset case. The lower accuracy and increased error variability align with the observed underfitting and limited extrapolation capacity documented in the error plots. This outcome underscores the impact of training data volume on neural network performance, particularly for complex and highly variable targets such as electric energy consumption.

9. Conclusion: Comparing Machine Learning and Deep Learning Models

This section presents a direct comparison between the best-performing machine learning and deep learning models for each of the two datasets. For the first dataset, the Hist Gradient Boosting Regressor is compared with the Multilayer Perceptron (MLP) model implemented with Keras. For the second dataset, the ExtraTreeRegressor is compared with the best-performing dense neural network architecture. The goal is to evaluate whether deep learning offers a significant advantage over traditional machine learning models in the context of tabular vehicle data.

First Dataset: Hist Gradient Boosting vs. MLP Regressor

For the first dataset, the Hist Gradient Boosting Regressor achieved exceptionally strong performance, with R^2 scores of 0.9961 for both training and test sets, and very low error values (MAE ≈ 0.26, RMSE ≈ 3.35). In comparison, the MLP Regressor trained on the full dataset achieved an R^2 score of 0.9287 and an RMSE of 6.54. Although the neural network generalized well, it fell short of the precision demonstrated by the gradient boosting model.

Metric	HistGradientBoosting	MLP (Keras)
R^2 Score	0.9961	0.9287
MAE	0.2593	—
RMSE	3.3492	6.54
RMSE (% of mean)	20.55%	3.89%

Although the RMSE as a percentage of the mean is slightly lower for the MLP model, the absolute RMSE and R^2 clearly favor the Hist Gradient Boosting Regressor. Therefore, for the first dataset, the machine learning model not only achieves better predictive accuracy but also demonstrates more consistent generalization.

Second Dataset: ExtraTreeRegressor vs. Dense Neural Network

For the second dataset, the ExtraTreeRegressor performed remarkably well, achieving an R^2 score of 0.9707 and an RMSE of 4.20. The best dense neural network (trained on the full dataset) reached an R^2 score of 0.8441 and an RMSE of 9.68, with a relative error of 5.76%, slightly above the acceptable threshold of 5%.

The table below summarizes the comparative performance of these two models:

Metric	ExtraTreeRegressor	Dense Neural Network
R^2 Score	0.9707	0.8441
RMSE	4.20	9.68
RMSE (% of mean)	2.50%	5.76%
Within 5% threshold?	Yes	No

The ExtraTreeRegressor clearly outperforms the dense neural network in every metric, delivering higher accuracy and lower error. Its ability to remain within the 5% deviation threshold reinforces its superiority for the second dataset.

In conclusion, although deep learning models showed competent performance, traditional machine learning models—specifically Hist Gradient Boosting and ExtraTreeRegressor—proved to be more accurate, stable, and efficient in both tasks. Given the tabular nature of the datasets and the size of the training data, tree-based ensemble methods remain the most suitable choice.

Part 2 - Predicting Combustion Fuel Consumption

1. Basic Description of the Task

The dataset used consists of information on newly registered vehicles in the EU market and their features, including wltp_test_mass, vehicle_type, fuel_type, engine_power, engine_capacity, specific_co2_emissions for the WLTP test cycle, fuel_consumption, among others. After preprocessing the data as described in Report 1, the task was to evaluate various models for predicting either fuel_consumption, specific_co2_emissions, or electric_energy_consumption.

Due to differences in features between vehicles with combustion engines and those with electric engines — specifically fuel_consumption and specific_co2_emissions for combustion engine vehicles, and electric_energy_consumption for electric vehicles — the dataset was divided into two subsets: one for combustion engines and one for electric engines. The modelling of combustion engines was assigned to Thomas and another contributor. The modelling of electric vehicles was handled by Philipp and Leonel.

2. Type of Machine Learning Problem

As the target variable is continuous, the problem was approached using regression models.

3. Main Performance Metrics Used

The performance of the models was evaluated using the score and RMSE (root mean squared error) metrics. These metrics were chosen due to their interpretability and availability through common Python libraries.

4. Algorithms Used

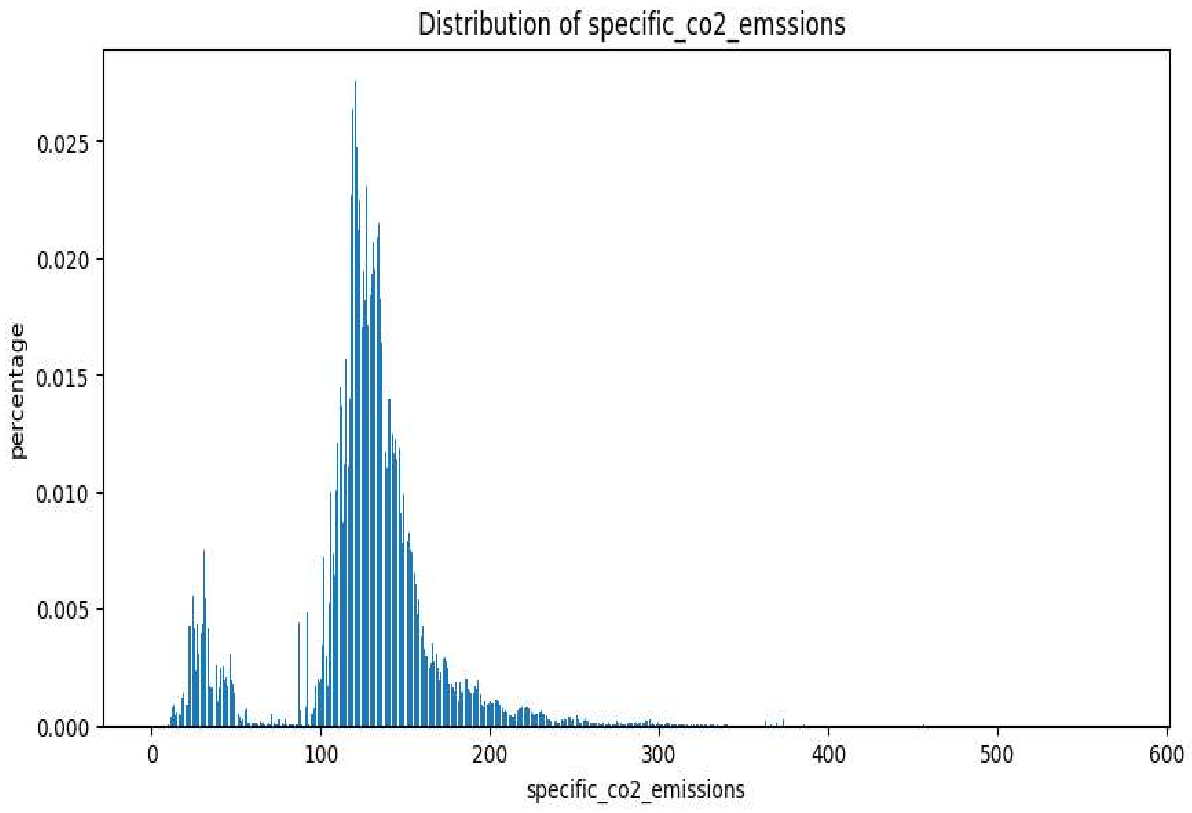
After removing missing values from the dataset, the following features were selected for the modelling process:

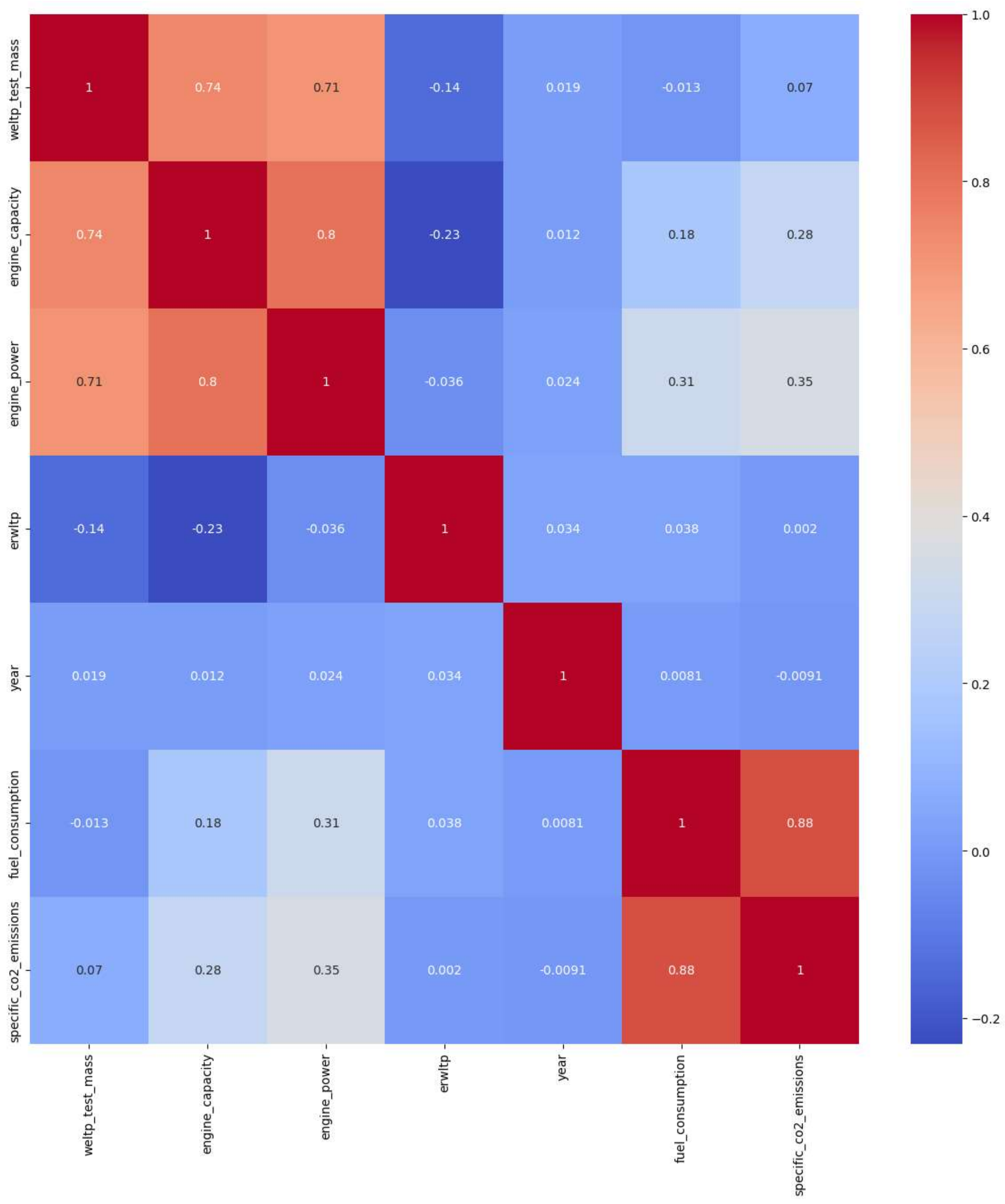
- member_state
- manufactorer_name_eu

- vehicle_type
- category_of_vehicle
- fuel_mode
- fuel_type
- innovative_technologies
- wltp_test_mass
- engine_capacity
- erwltp
- year

The features were divided by type into numerical columns (int64 or float64) and categorical columns (object). Numerical columns were standardized using StandardScaler to normalize values to a common scale between 0 and 1. Categorical columns were encoded using LabelEncoder.

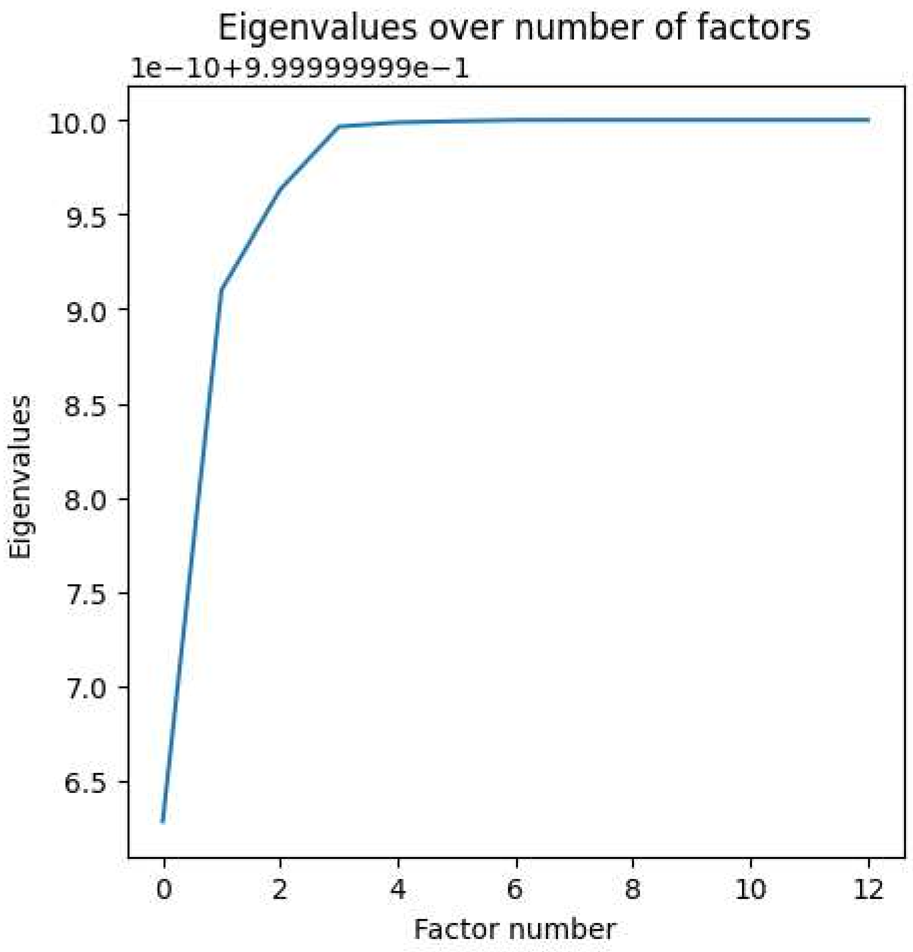
A correlation heatmap was also generated. Based on this analysis, one of the two potential target variables — specific_co2_emissions and fuel_consumption — was excluded, as one can be derived from the other. As a result, specific_co2_emissions was selected as the target variable, and fuel_consumption was removed from the dataset.





The train_test_split function was applied to divide the dataset into training and test sets, with 20% of the data reserved for testing. The initial model tested was a simple LinearRegression, which yielded a training and testing score of approximately 0.4 and an RMSE of 30 g/km. For reference, the mean of the target variable (specific_CO2_emission) was 126 g/km.
The application of LassoCV produced similarly unconvincing results.
In contrast, the XGBRegressor provided the best performance, achieving a score of 0.997 and an RMSE of 4 g/km.

Subsequently, a GridSearchCV was conducted to fine-tune hyperparameters. However, due to significant runtime issues, the full training dataset could not be processed — no results were obtained even after more than 10 hours of execution. To mitigate this, the training data was drastically reduced to 1% of its original size. Additionally, Principal Component Analysis (PCA) was applied to further optimize computation time.

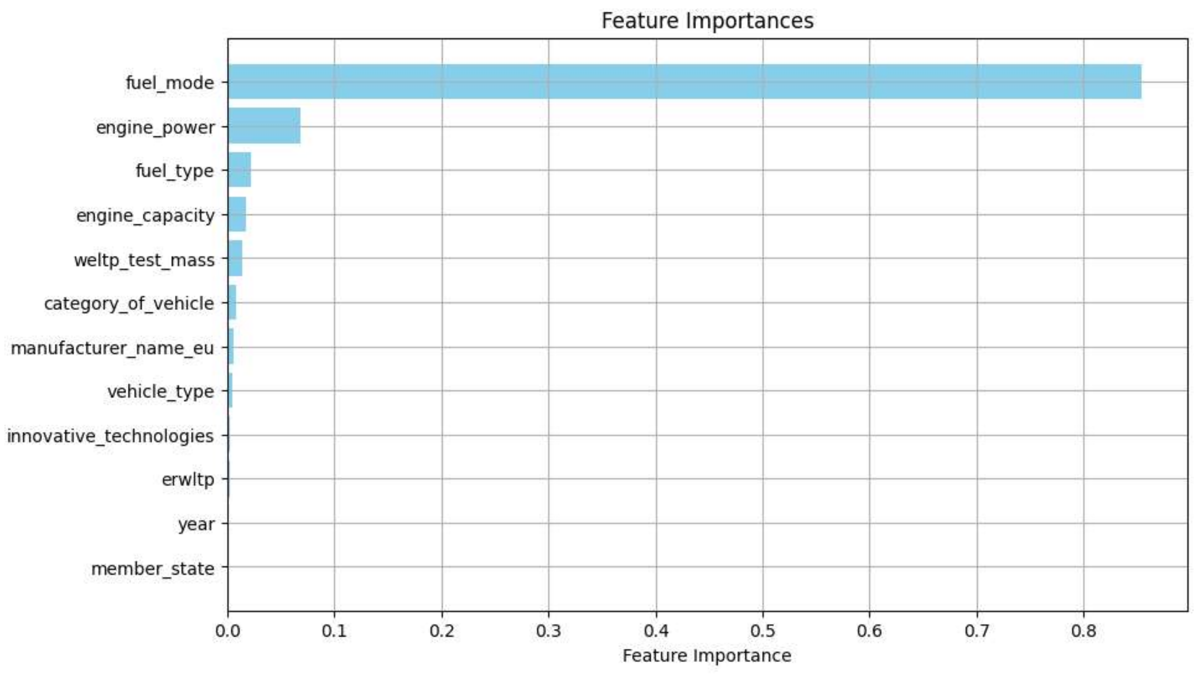


With the reduced dataset, a Randomized Search CV was executed, but it did not produce final results due to excessive runtime. However, intermediate results obtained during execution indicated that the Linear Regression model performed poorly compared to alternative models and was therefore excluded from further consideration.
To support this decision, another Randomized Search CV was conducted using three models: XGBoost Regressor, Decision Tree Regressor, and Random Forest Regressor. The Linear Regression model consistently showed the worst performance among the evaluated models. Specifically, a Linear Regression model without any parameter tuning resulted in a root mean squared error (RMSE) of 31.8 g/100km for specific CO2 emissions.

In contrast, the XGBoost Regressor (with 1000 estimators and a learning rate of 0.3) achieved an RMSE of 3.5716 g/100km, corresponding to approximately 2% of the mean CO2 emission. The Decision Tree Regressor (with no depth limit) reached an RMSE of 4.9961 g/100km, approximately 4% of the mean CO2 emission.

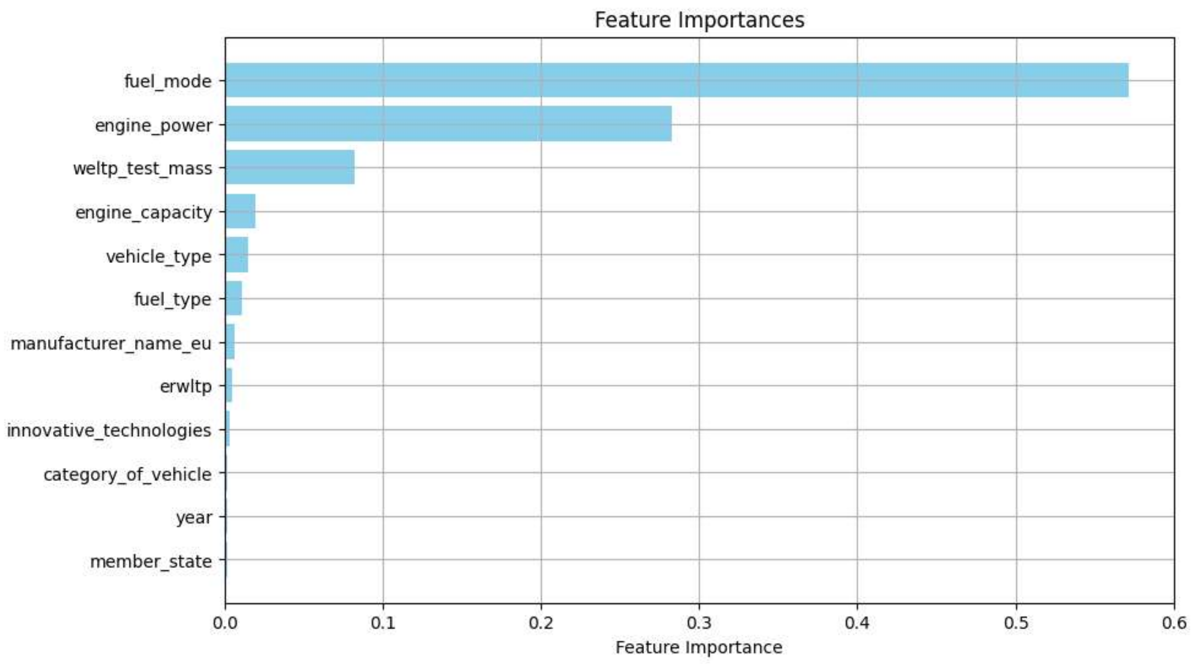
5. Interpretation of Results

To gain insight into the best-performing models, the feature importances derived from the XGBoost Regressor were analyzed. The analysis showed that *fuel mode* was the most influential variable, followed — at a significant distance — by *engine power*, *engine capacity*, and *WLTP test mass*.



The sorted feature importances were also analyzed for the Decision Tree Regressor. In this model as well, fuel mode emerged as the most relevant feature, followed—at a smaller distance compared to the XGBoost Regressor—by engine power and WLTP test mass. Subsequently, the XGBoost Regressor was trained using the best-performing parameters (1000 estimators and a learning rate of 0.1) and 80% of the dataset in order to compute additional performance metrics such as mean absolute error (MAE), mean squared error (MSE), and R-squared for the test data.
The resulting performance was as follows:

- Mean Absolute Error (MAE): 1.4015 g/100km
- Mean Squared Error (MSE): 6.1988 g/km
- R-squared (R^2): 0.9963



The sorted feature importances were also analyzed for the Decision Tree Regressor. In this model, fuel mode again emerged as the most relevant feature, followed — at a shorter distance compared to the XGBoost Regressor — by engine power and WLTP test mass. Subsequently, the XGBoost Regressor was trained using the best parameter configuration (1000 estimators and a learning rate of 0.1) and 80% of the dataset, in order to compute additional performance metrics on the test data, including mean absolute error (MAE), mean squared error (MSE), and the coefficient of determination (R^2).
The model achieved the following results:

- Mean Absolute Error (MAE): 1.4015 g/100km
- Mean Squared Error (MSE): 6.1988 g/km
- R^2 Score: 0.9963

Part 3 – Predicting Combustion Fuel Types (Classification)

1. Basic Description of the Task

This section focuses on predicting the type of fuel used by vehicles based on various performance-related variables. The objective of this analysis is primarily scientific and explanatory in nature, rather than practical, with the additional intent of identifying potential insights that could support the main regression task of the project.

The data used in this section is based on the preprocessed files located in the /app_emulation directory, prepared by Philipp. The dataset was segmented to emphasize features related to combustion engines, which were allocated to Richard and another team member for further analysis.

Throughout the course of this stage, the dataset was refined and reduced. Initially, emphasis was placed on numerical variables directly related to vehicle performance, such as mass, engine power, and fuel consumption. Categorical variables that could introduce potential bias — including vehicle name, manufacturer, and country of origin — were removed to ensure a neutral and consistent dataset.

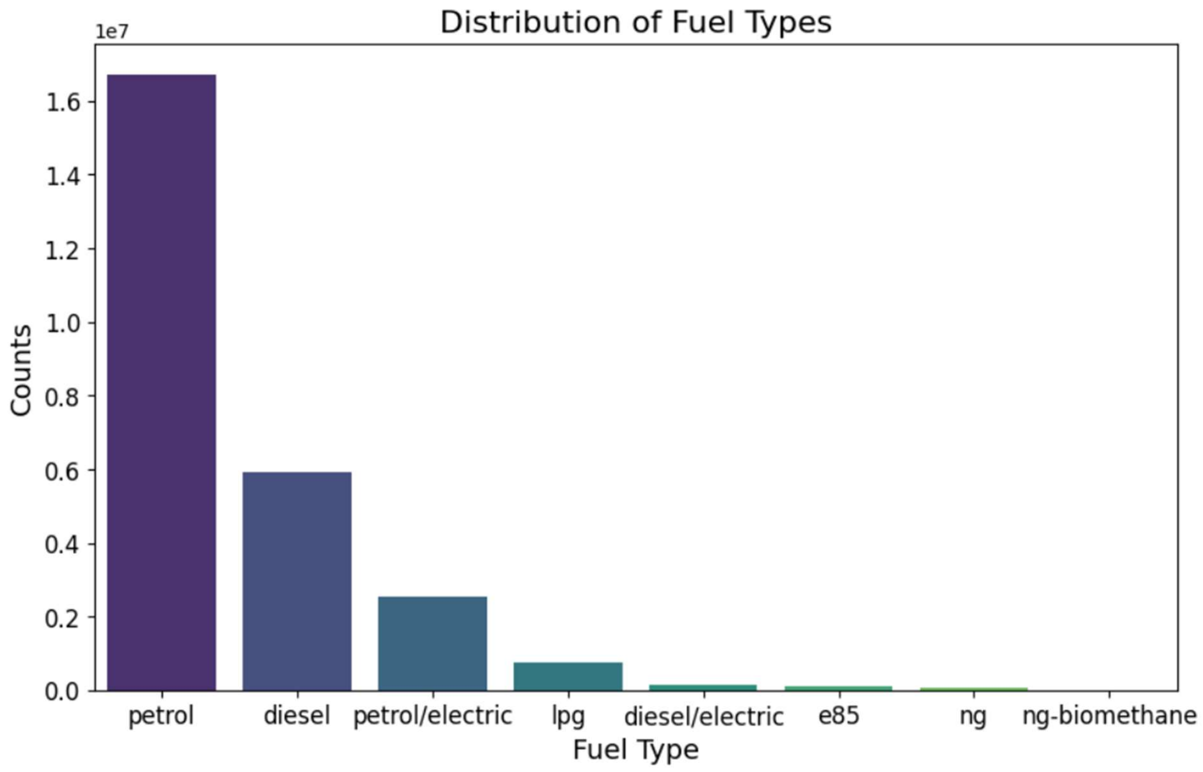
Variables from earlier preprocessing steps that were specific to electric vehicles were also excluded in order to streamline the analysis and focus exclusively on combustion engines. Based on guidance received during a coaching session with Romain, features directly linked to environmental impact, such as CO_2 emissions, were also excluded. The recommendation was to prioritize intrinsic vehicle characteristics rather than environmental outcomes.

Additionally, to address redundancy, a decision was made to retain only one of the two variables related to vehicle mass. The variable *wltp_test_mass* was removed in favor of *mass_vehicle*.

The primary goal of this analysis is to classify vehicles according to their fuel type using classification-based machine learning models. The focus is on distinguishing between fuel types such as petrol, diesel, and various hybrid configurations.

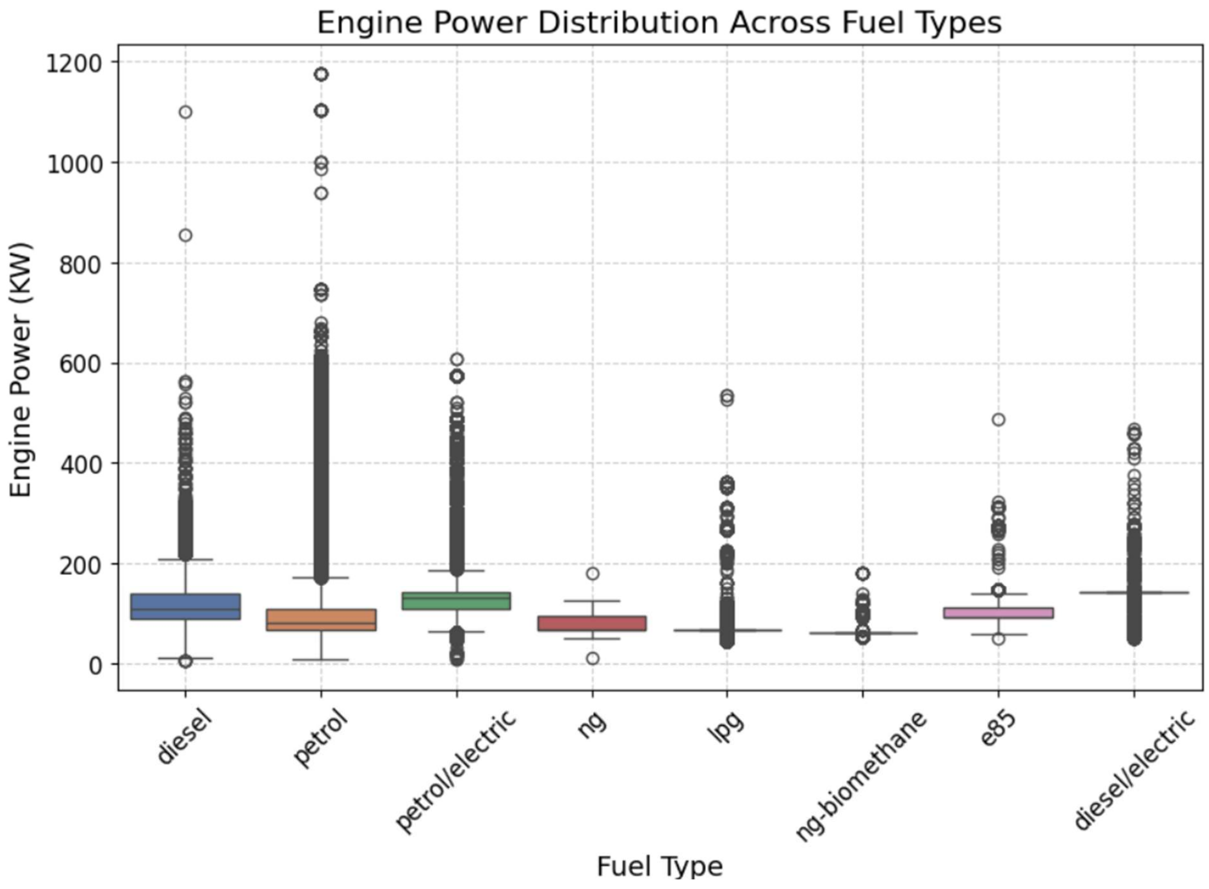
2. First Insights Through Plots

This section begins by analyzing the distribution of fuel types within the dataset. This initial step provides a foundation for subsequent, more detailed analyses of how various features influence fuel type classification.



The figure illustrates the distribution of fuel types within the dataset. It is evident that petrol-powered vehicles dominate the sample by a large margin, followed by diesel and hybrid petrol/electric vehicles. Other fuel types such as LPG, diesel/electric hybrids, E85, natural gas (NG), and NG-biometane are present in considerably smaller numbers. This pronounced class imbalance may have implications for the performance of classification models, particularly in terms of their ability to accurately identify minority fuel types.
To evaluate the relevance of the chosen approach and determine whether the dataset contains sufficient distinguishing patterns to support effective classification, several vehicle-related variables were examined. One of the key variables selected for initial exploration was *engine power*, which was visualized using a distribution plot to assess its potential as a predictor of

fuel type.



The figure presents the distribution of engine power across different fuel types. The data shows a moderate degree of variance within most categories and a considerable number of distinguishable outliers. These differences in engine power distributions among fuel types suggest that this variable has the potential to serve as a meaningful predictor in the classification task. The observed separability reinforces the validity of the overall modelling approach proposed for fuel type classification.

<u>3. Tree-Based Models and Challenges Related to Performance and Data Distribution</u>

For the classification task, accuracy was selected as the primary performance metric to assess the model's basic effectiveness in correctly categorizing fuel types. Additional metrics were considered for inclusion, depending on time constraints.

The modeling approach initially focused on tree-based algorithms due to their interpretability, which aligned with the exploratory and explanatory objectives of this sub-project. Random

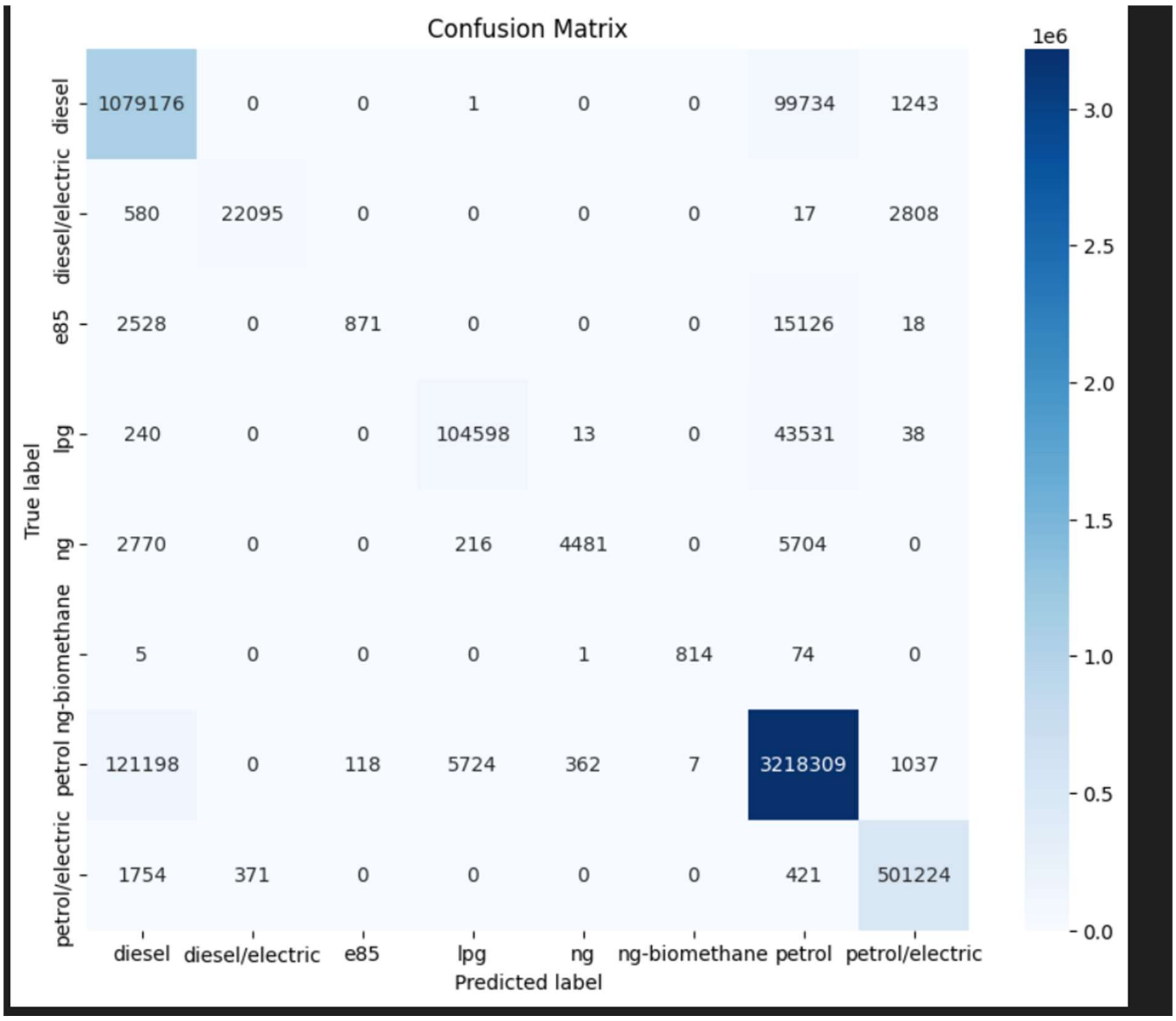
Forest was the first algorithm tested; however, its application to the full dataset resulted in extensive computation times and poor performance, indicating that the model struggled to handle the dataset's size effectively. At that stage of the project, it was not yet known that other team members were encountering similar challenges.

Subsequently, the dataset was reduced in size to mitigate these computational issues. The inclusion of data spanning three years from the EU endpoint, while comprehensive, had introduced significant processing burdens without yielding proportional benefits. A consensus was later reached among team members to adopt a reduced dataset as a practical solution.

At this point, the modeling strategy was adjusted by transitioning to the XGBoost algorithm. XGBoost was selected for its ability to handle larger datasets efficiently while preserving a degree of interpretability, as it is also based on decision trees. This balance made it particularly suitable for the scientific objectives of the classification analysis.

To further improve runtime efficiency, dimensionality reduction was applied using Principal Component Analysis (PCA), retaining 85% of the original variance. This combination of techniques allowed XGBoost to deliver effective and reliable performance without a significant loss in accuracy.

The first confusion matrix generated from an initial model run is shown below:



True label \ Predicted label	diesel	diesel/electric	e85	lpg	ng	ng-biomethane	petrol	petrol/electric
diesel	1079176	0	0	1	0	0	99734	1243
diesel/electric	580	22095	0	0	0	0	17	2808
e85	2528	0	871	0	0	0	15126	18
lpg	240	0	0	104598	13	0	43531	38
ng	2770	0	0	216	4481	0	5704	0
ng-biomethane	5	0	0	0	1	814	74	0
petrol	121198	0	118	5724	362	7	3218309	1037
petrol/electric	1754	371	0	0	0	0	421	501224

Single Fuel Accuracies:

Diesel: 91.45%

Diesel/Electric: 86.65%

E85: 4.70%

LPG: 70.01%

NG-Biomethane: 34.59%

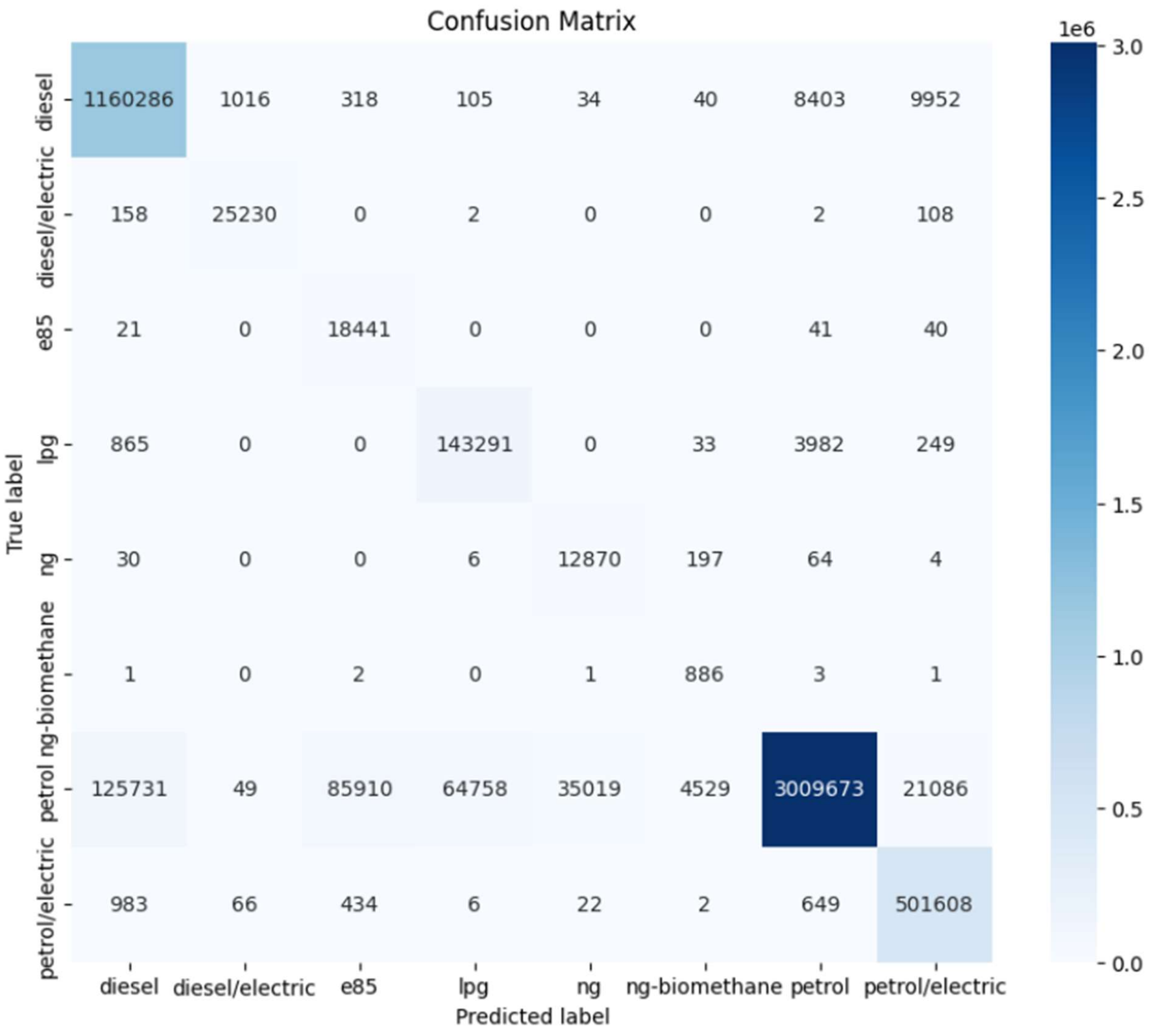
Petrol: 94.15%

Petrol/Electric: 99.33%

The confusion matrix revealed substantial variance in classification accuracy across the different fuel types. In particular, lower accuracy scores were observed for classes such as E85 and NG-Biomethane, while LPG also demonstrated only moderate classification performance. This pattern is consistent with the class imbalance previously identified in the fuel type distribution, where these minority classes had significantly fewer samples compared to dominant classes such as Diesel and Petrol.

To address this issue, Synthetic Minority Over-sampling Technique (SMOTE) was applied. SMOTE generates synthetic examples for underrepresented classes, such as E85 and NG-Biomethane, with the aim of improving class balance within the training dataset. This approach helps prevent the model from overfitting to the majority classes and enhances its ability to generalize across all fuel types.

The confusion matrix below presents the classification results obtained after training the XGBoost model on the SMOTE-augmented dataset:



	diesel	diesel/electric	e85	lpg	ng	ng-biomethane	petrol	petrol/electric
diesel	1160286	1016	318	105	34	40	8403	9952
diesel/electric	158	25230	0	2	0	0	2	108
e85	21	0	18441	0	0	0	41	40
lpg	865	0	0	143291	0	33	3982	249
ng	30	0	0	6	12870	197	64	4
ng-biomethane	1	0	2	0	1	886	3	1
petrol	125731	49	85910	64758	35019	4529	3009673	21086
petrol/electric	983	66	434	6	22	2	649	501608

Single Fuel Accuracies:

Diesel: 98.57%

Diesel/Electric: 98.94%

E85: 99.45%

LPG: 96.54%

NG-Biomethane: 97.76%

Petrol: 88.44%

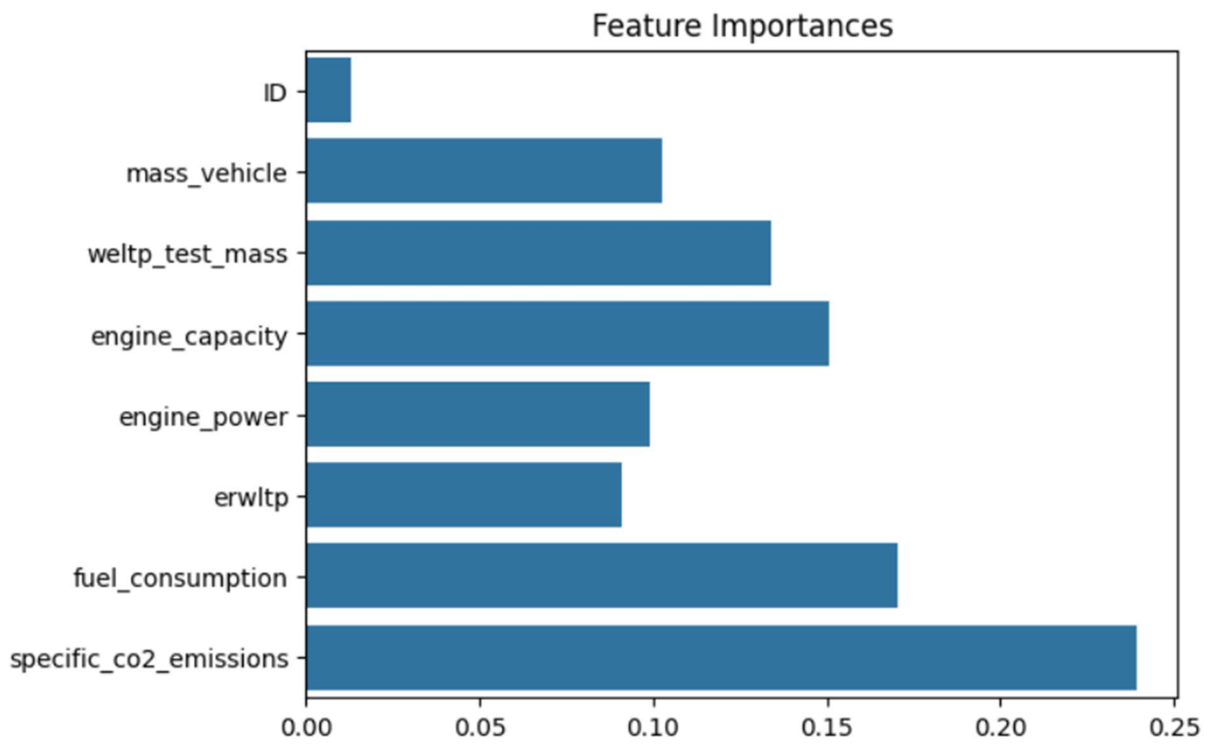
Petrol/Electric: 99.33%

The application of SMOTE led to a notable improvement in classification accuracy, particularly for previously underrepresented fuel types such as E85, LPG, and NG-Biomethane. These classes, which were initially poorly represented in the dataset, benefited significantly from the oversampling technique. This outcome confirms the effectiveness of SMOTE in addressing class imbalance within the dataset.

While a slight decrease in accuracy was observed for the Petrol class — the most prevalent in the dataset — this trade-off was acceptable. The adjusted model demonstrated improved generalization by no longer disproportionately favoring the dominant class. As a result, the overall model performance became more balanced and reliable for real-world applications.

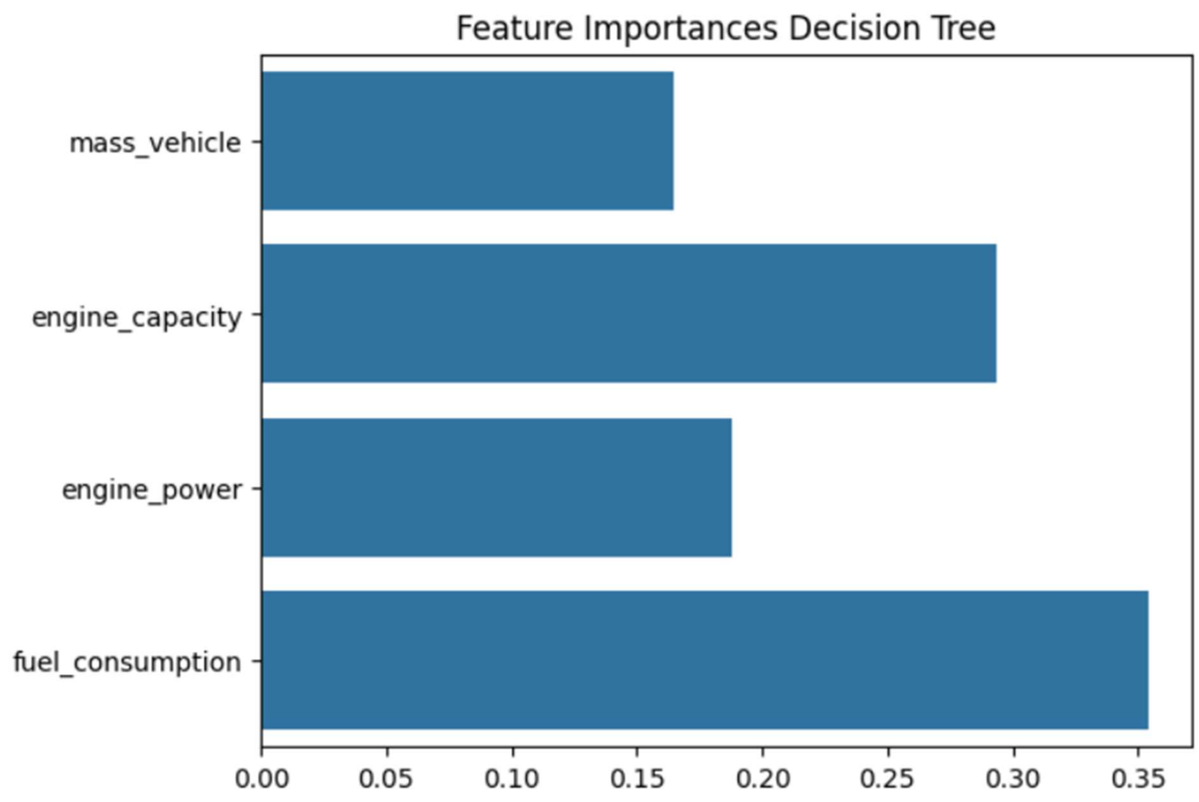
4. Feature Importance Analysis

To conduct a meaningful feature importance analysis, it was necessary to run the XGBoost model without dimensionality reduction via PCA, in order to preserve the original feature space. Additionally, since XGBoost requires the target variable to be encoded for classification tasks, the categorical fuel type variable had to be re-encoded after model training to allow proper interpretation of the results.



The feature importance graph presented was generated during an earlier stage of the modeling process. At that point, the dataset still included both mass-related variables, emission-related features, and even an identifier column.

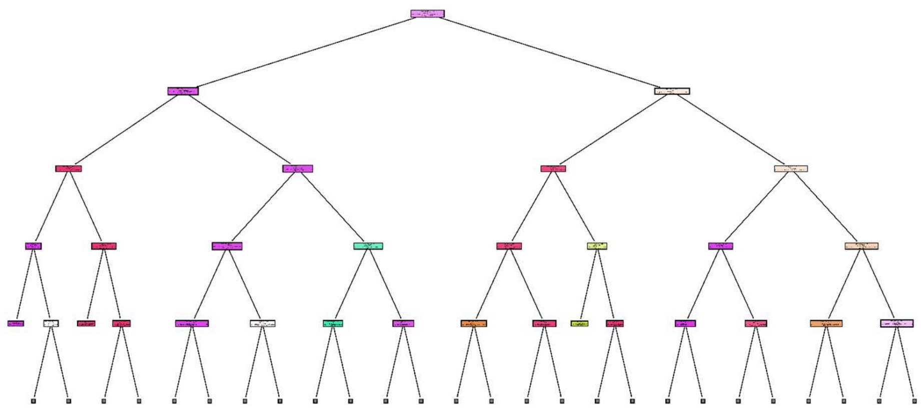
Notably, when compared to a subsequent graph generated at a later stage — after redundant and less informative variables had been removed — a significant shift in feature relevance can be observed. In the updated graph, *mass* ranks among the least important variables, indicating that its contribution to the predictive performance of the model is limited once other more informative features are retained.



In contrast, in the earlier graph, *mass_vehicle* appeared to be more influential than *engine power*, with *wltp_test_mass* showing an even higher level of importance. This shift highlights how data refinement and the removal of redundant variables can significantly impact feature relevance within the model.

5. Decision Tree Visualization

A decision tree graph was also generated for illustrative purposes. However, due to the high complexity of the full model, the graph is presented in truncated form. This limitation is not a result of Python runtime constraints, but rather due to practical visualization issues. A complete tree graph, containing several hundred nodes, exceeds the display capacity of the system and results in extremely large output files — in some cases surpassing 100 megabytes even in SVG format.



A tree graph was originally considered early in the analysis for its interpretability benefits. However, it only became feasible to generate such a visualization after reducing the dataset and minimizing runtime requirements. This graph was produced using a Decision Tree model rather than XGBoost.

In practical terms, the tree graph yielded limited interpretative value. While it provided some technical insight into the structure and mechanics of decision trees, the primary takeaway was the confirmation that variables with high feature importance — such as *engine_capacity* and *fuel_consumption* — tend to appear earlier and more frequently in the decision nodes, as expected.

6. Limited Deep Learning Classification Using Keras

As a final experimental step — and under strict time constraints — a basic deep learning model was implemented using the Keras framework. The model was trained with a very limited number of epochs, with the primary goal being a preliminary comparison of classification performance relative to previously applied machine learning models.

Categorical crossentropy was selected as the loss function, as the task involves multi-class classification with non-binary categorical targets.

The results from the initial deep learning model run were as follows:

- Test Loss: 0.034
- Test Accuracy: 0.989

Based on the accuracy metric, this result is comparable to the performance achieved by tree-based machine learning models. With additional time, it would be of interest to explore this deep learning approach further, incorporating techniques for hyperparameter optimization and model interpretability to better understand its internal decision processes.

7. Possible Next Steps

The classification task for fuel types using vehicle characteristics yielded promising results. To further improve model performance, additional efforts could focus on hyperparameter tuning — for instance, applying Grid Search or Randomized Search to optimize the configuration of the XGBoost model.

Insights gained from the feature importance analysis suggest a potential reconsideration of the decision to exclude *wltp_test_mass* in favor of *mass_vehicle*. A more detailed statistical evaluation is recommended to assess the impact of this choice. This could involve reintroducing *wltp_test_mass*, removing *mass_vehicle*, and conducting comparative model runs to determine which variable provides greater predictive value.