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

Part 1 - Predicting Electric Energy Consumption

Abstract

This report presents the development and evaluation of machine learning models for predicting electric energy consumption based on European vehicle data. Two datasets were used to train and compare models of varying complexity, including Gradient Boosting and Extra Tree Regressors. The process involved initial benchmarking with LazyRegressor and further refinement through hyperparameter tuning using Optuna and RandomizedSearchCV, with model selection based on R^2 and RMSE metrics.

In the final stage, a deep learning model — MLP Regressor implemented with Keras — was applied to evaluate the performance of a neural network architecture on the same datasets. Additionally, SHAP analysis was conducted to interpret feature contributions. The results demonstrate that all selected models generalize well, with low prediction error and consistent performance across datasets. This project was carried out as part of the Data Science program, aiming to assess how models of different levels of complexity behave when applied to structured vehicle data for target variable prediction.

1. Introduction to the Project

This project focuses on predicting electric energy consumption in vehicles registered in the European Union using machine learning and deep learning techniques. To ensure robust evaluation and 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.

2. 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.

2.1 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

2.2 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.

3. 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.

4. 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.

4.1 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

4.2 Second Dataset

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

a. ExtraTreeRegressor

b. RandomForestRegressor

c. DecisionTreeRegressor

5. Hyperparameter Optimization and Cross-Validation

5.1 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.

---

[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.

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:

- 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': 89, 'learning_rate': 0.0821, 'max_depth': 3, 'subsample': 0.993, 'min_samples_split': 5}	0.962	0.961	89

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.

5.2 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.

5.3 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': None, 'max_depth': 10}	0.947	0.70	50
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.

5.4 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.

6. 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, Hist Gradient Boosting

Regressor, and LGBM 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.

6.1 Machine Learning Models, Their Results and Feature Importance

Gradient Boosting Model – First Dataset

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 ≈ 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

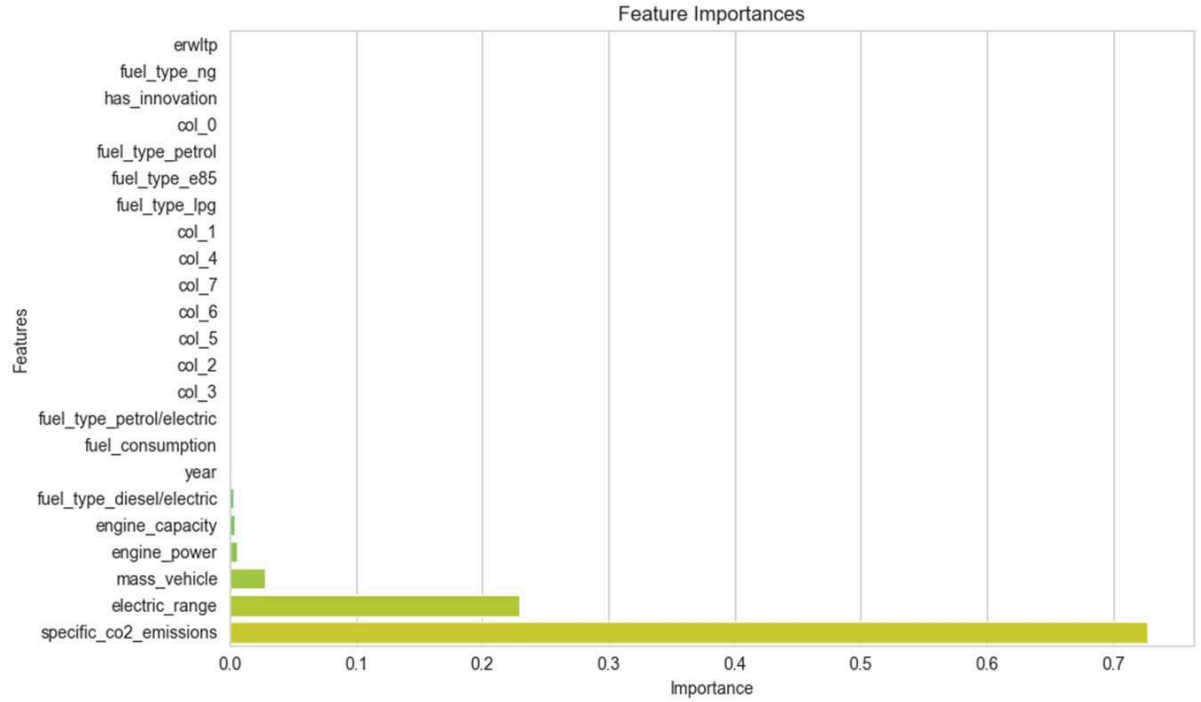
Given the average electric energy consumption of approximately 182.4, the RMSE and MAE values can be interpreted as follows:

- 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



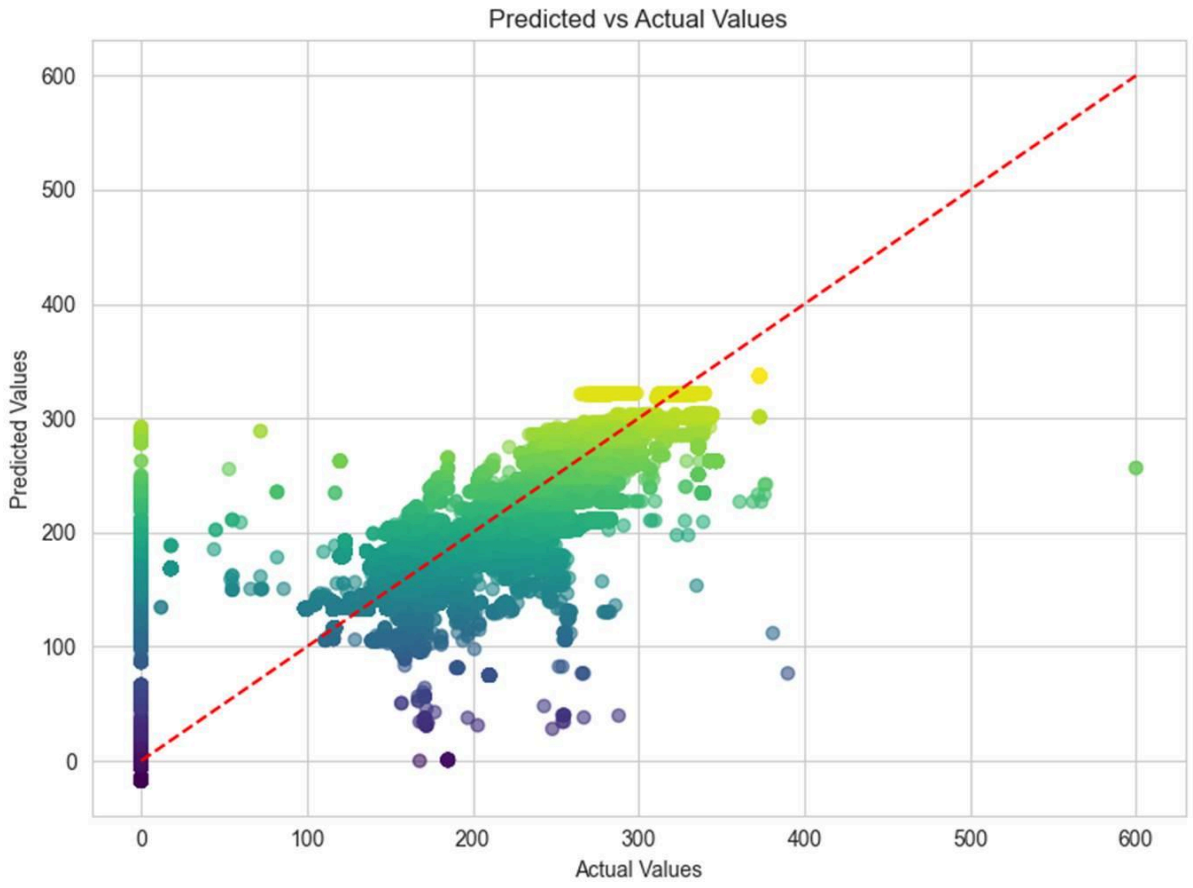
The bar chart indicates that specific_co2_emissions, mass_vehicle, and electric_range stand out as the top three predictors for electric energy consumption. This suggests that emission levels, vehicle weight, and battery range are the strongest drivers in the model's predictions.

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.

ExtraTreeRegressor – Second Dataset

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

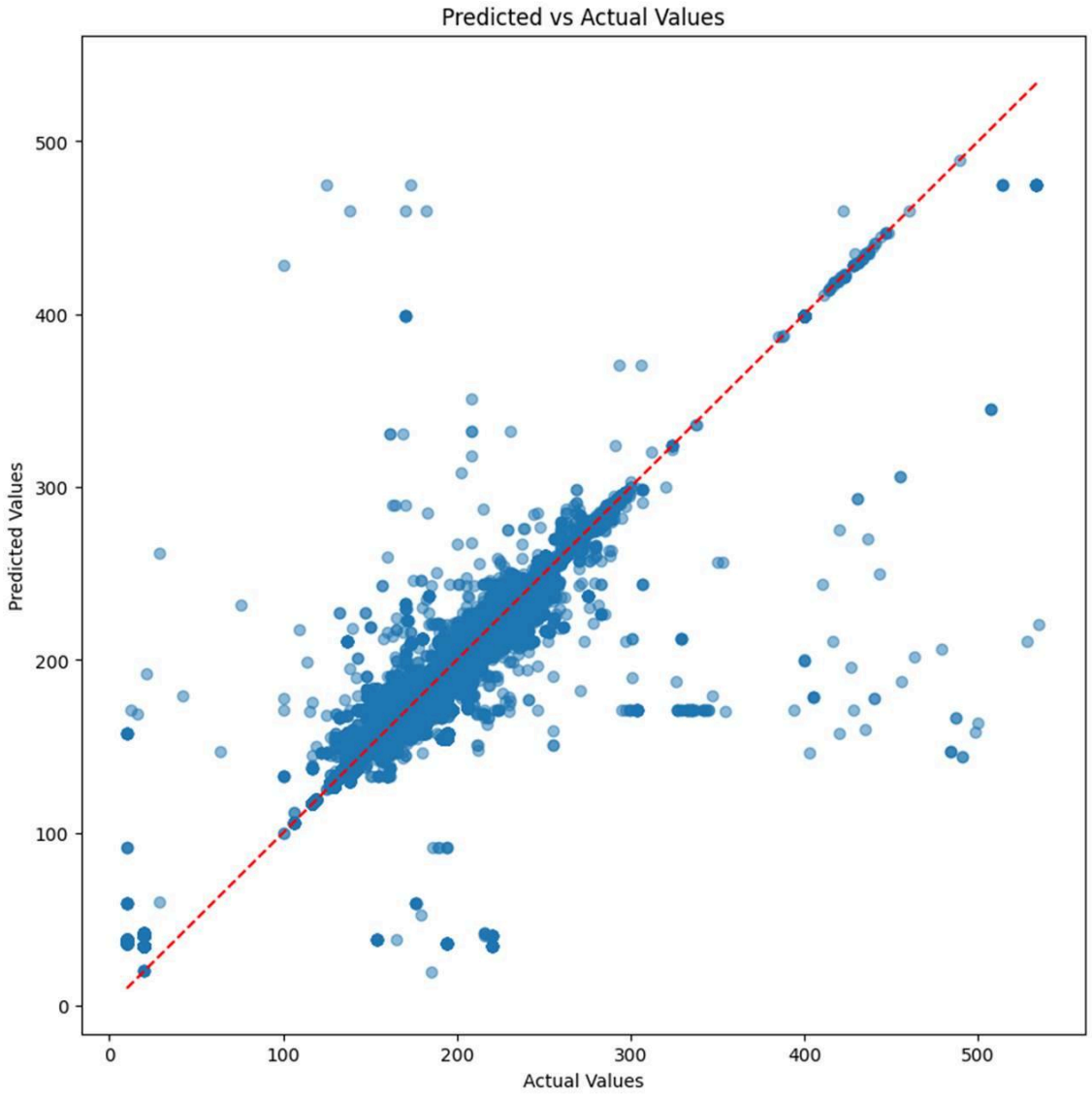
The ExtraTreeRegressor model achieved an R^2 of 0.9707, which clearly surpasses the commonly accepted threshold of 0.95 for high explanatory power. This indicates that the model is able to explain approximately 97% of the variance in the target variable, suggesting a very strong fit.

The adjusted R^2 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

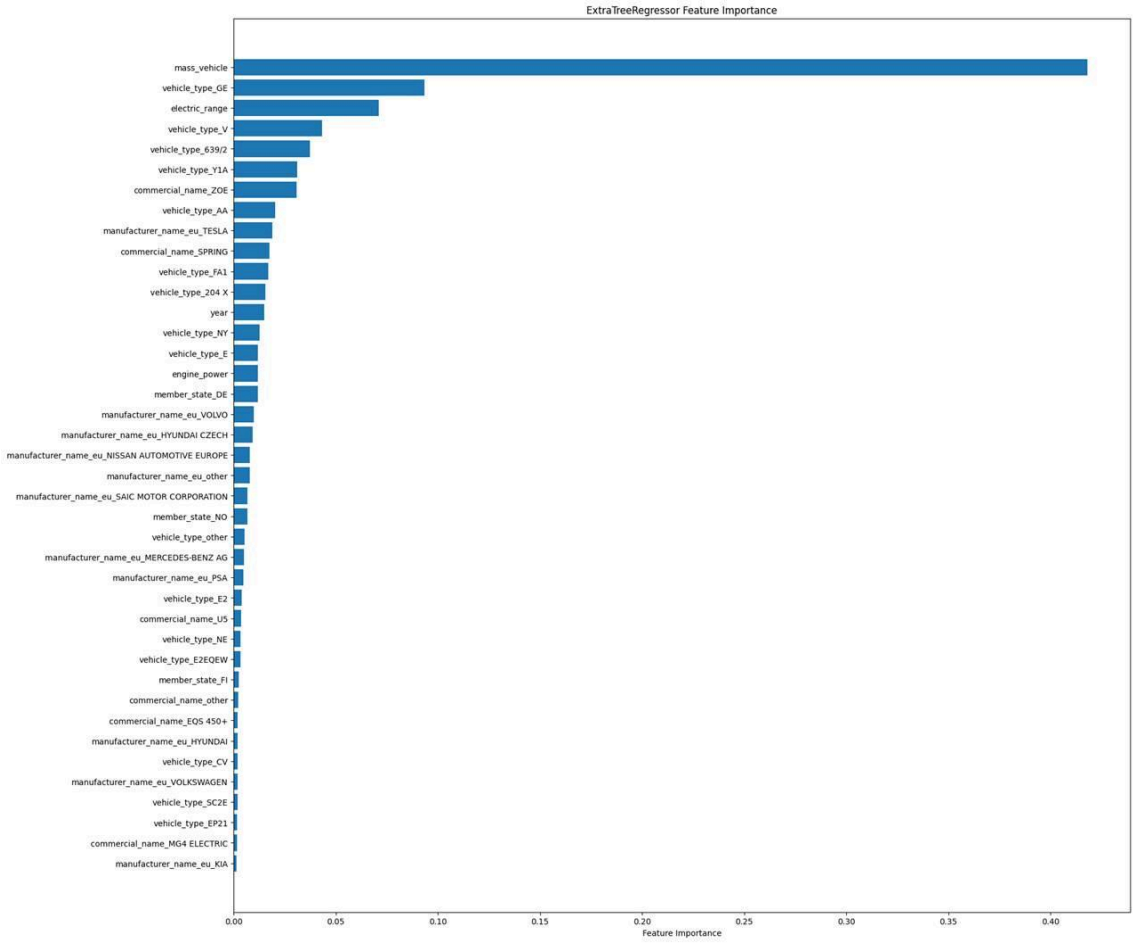


This scatter plot compares the actual values (x-axis) against the predicted values (y-axis) from the ExtraTreeRegressor. The red dashed line represents the ideal case where predicted values perfectly match actual values (i.e., $y = x$).

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

MPL Regression Model with Keras para o 1º conjunto de dados

SHAP Model para o 2º conjunto de dados

—- DRAFT

Case 1

- constructed simple dense neural network on entire dataset
- X_train shape 2.661.120 rows
- X_test shape 1.140.481 rows

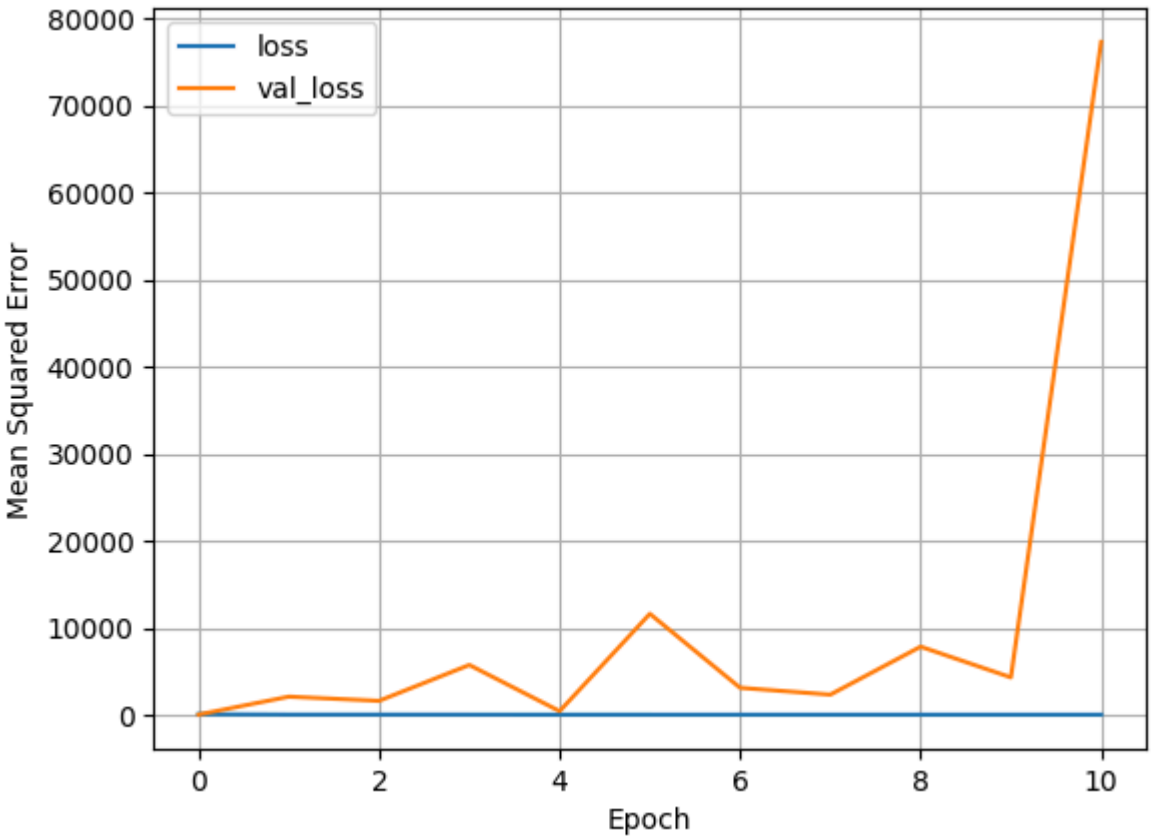
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 298)	0
dense (Dense)	(None, 64)	19,136
batch_normalization (BatchNormalization)	(None, 64)	256
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 64)	4,160
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)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

Total params: 25,921 (101.25 KB)

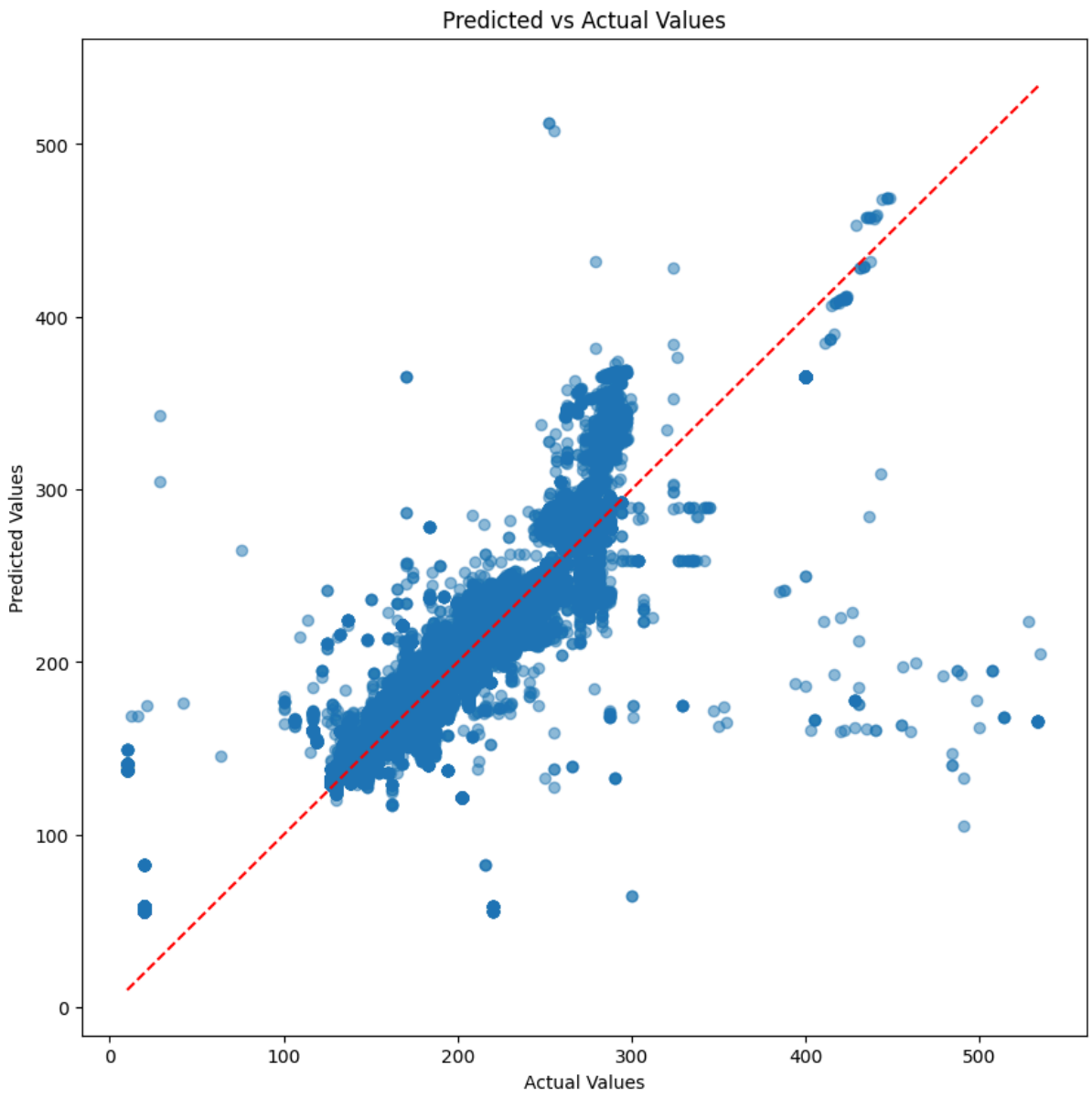
Trainable params: 25,665 (100.25 KB)

Non-trainable params: 256 (1.00 KB)

- Dense with relu as activation function for regression problem -> Multiple layers but sequentially reducing number of units
- BatchNormalization: Normalizes activations to stabilize training.
- Dropout: Prevents overfitting by randomly disabling neurons.
- loss: MSE (Mean Squared Error)
- evaluation metric: MAE (Mean Absolute Error)



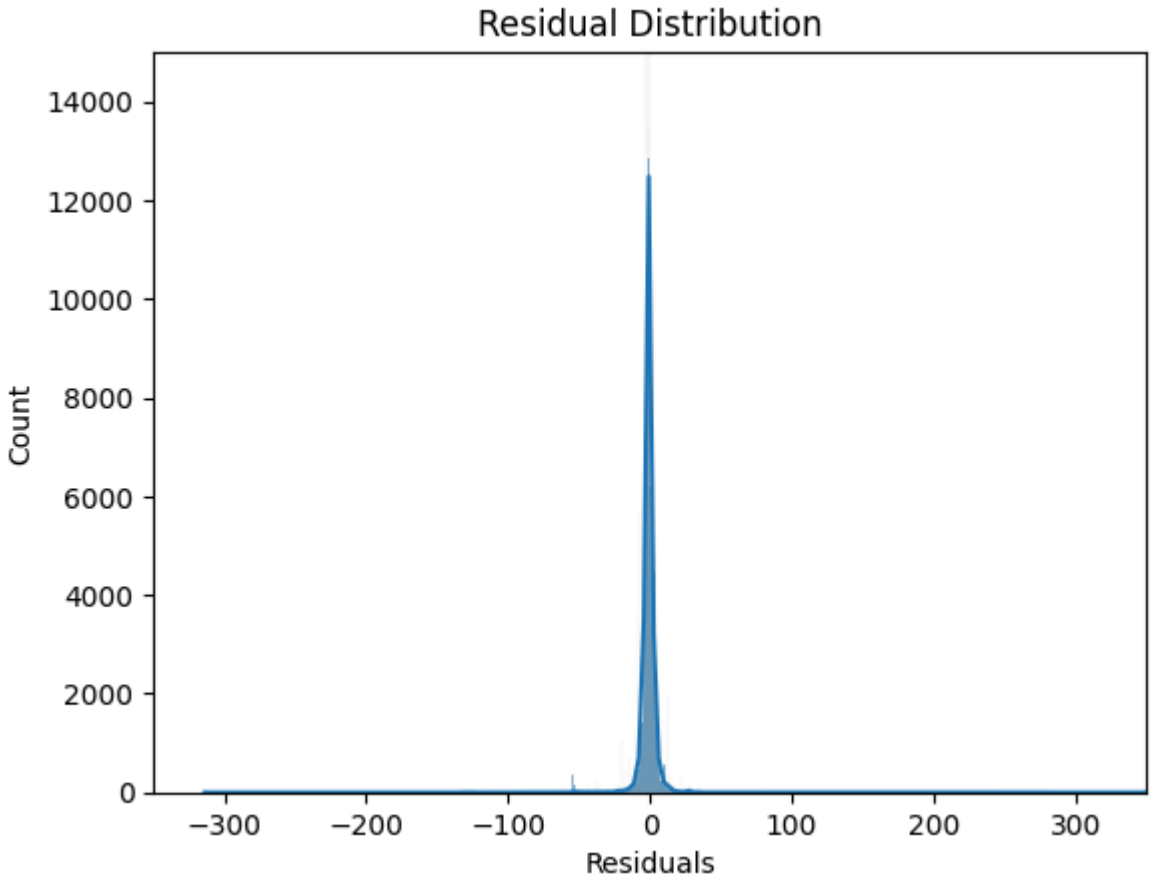
- loss converges to a stable value without diverging after first epoch.
- in epochs 1-4 and 6-7 smallest gap between loss and val_loss compared to other epochs.
- at later stage overfitting:
    - loss remains around same value, but val_loss starts increasing after some epochs.
    - the model performs well on training data but poorly on validation data.
- Conclusion: could be related to outliers which negatively impact nn model training or model is too complex for this usecase.



Solid performance within electric energy consumption range 0 - 300

Above that range the model sometimes predicts values significantly lower than actual value.

Below that range the model predicts values significantly higher than actual values.



Residual plot shows small residuals for majority of predictions

Statistical Summary

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

Case 2

- constructed simple dense neural network on small sample of dataset
- X_train shape: 10.000 rows
- X_test shape: 1.140.481 rows

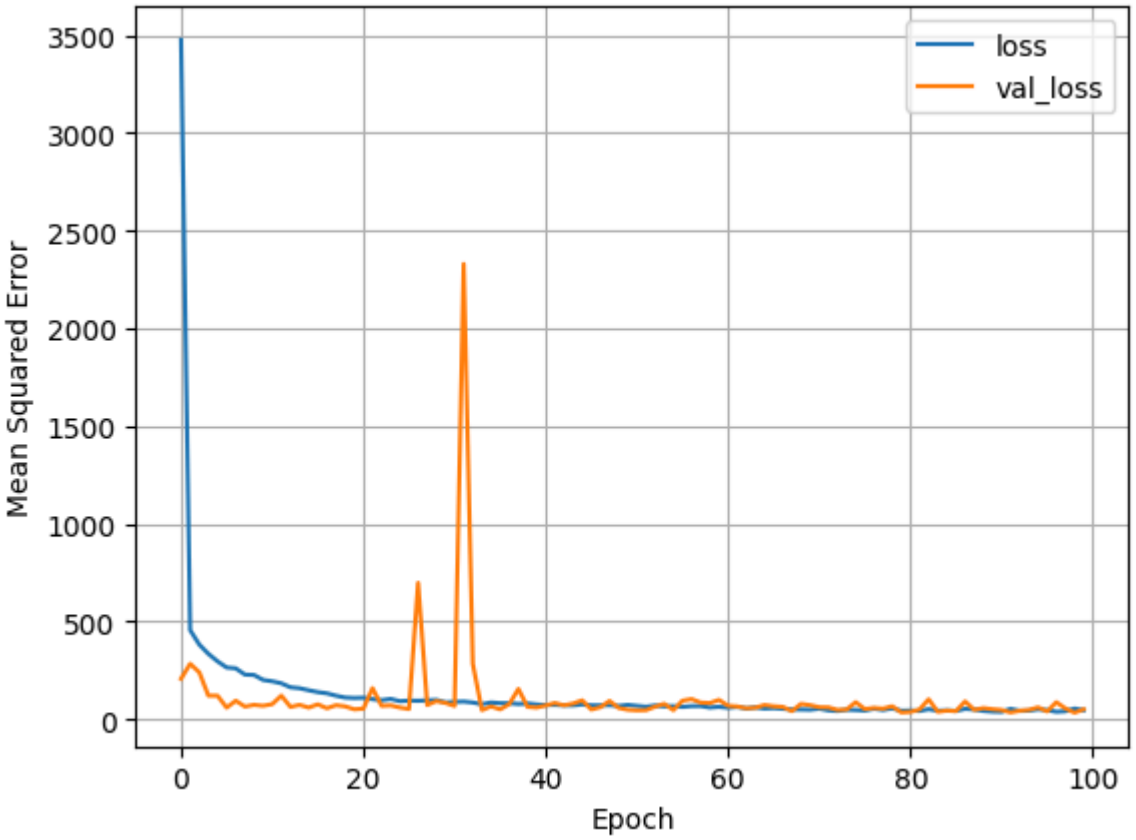
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 298)	0
dense (Dense)	(None, 128)	38,272
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)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

Total params: 49,409 (193.00 KB)

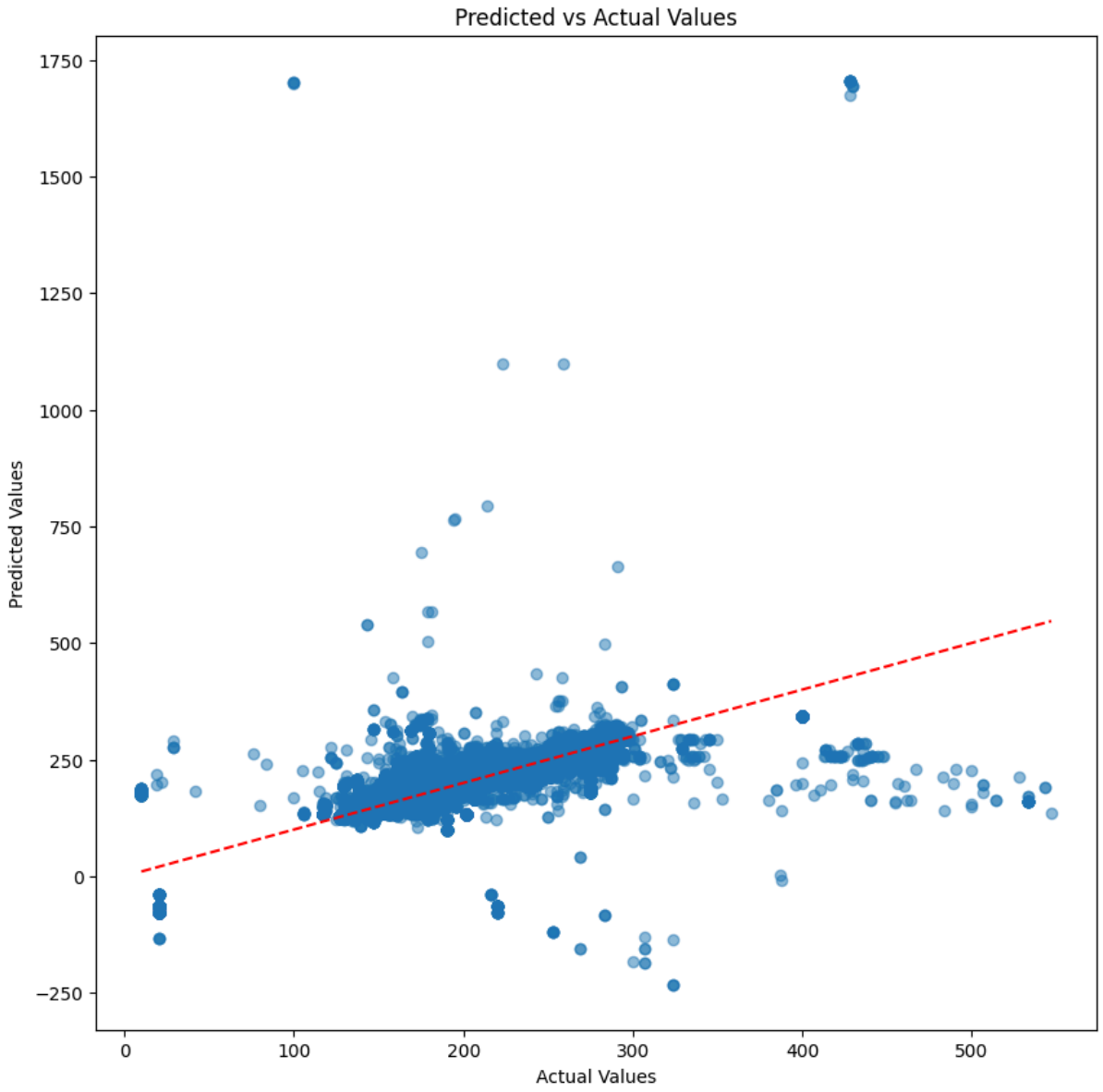
Trainable params: 49,025 (191.50 KB)

Non-trainable params: 384 (1.50 KB)

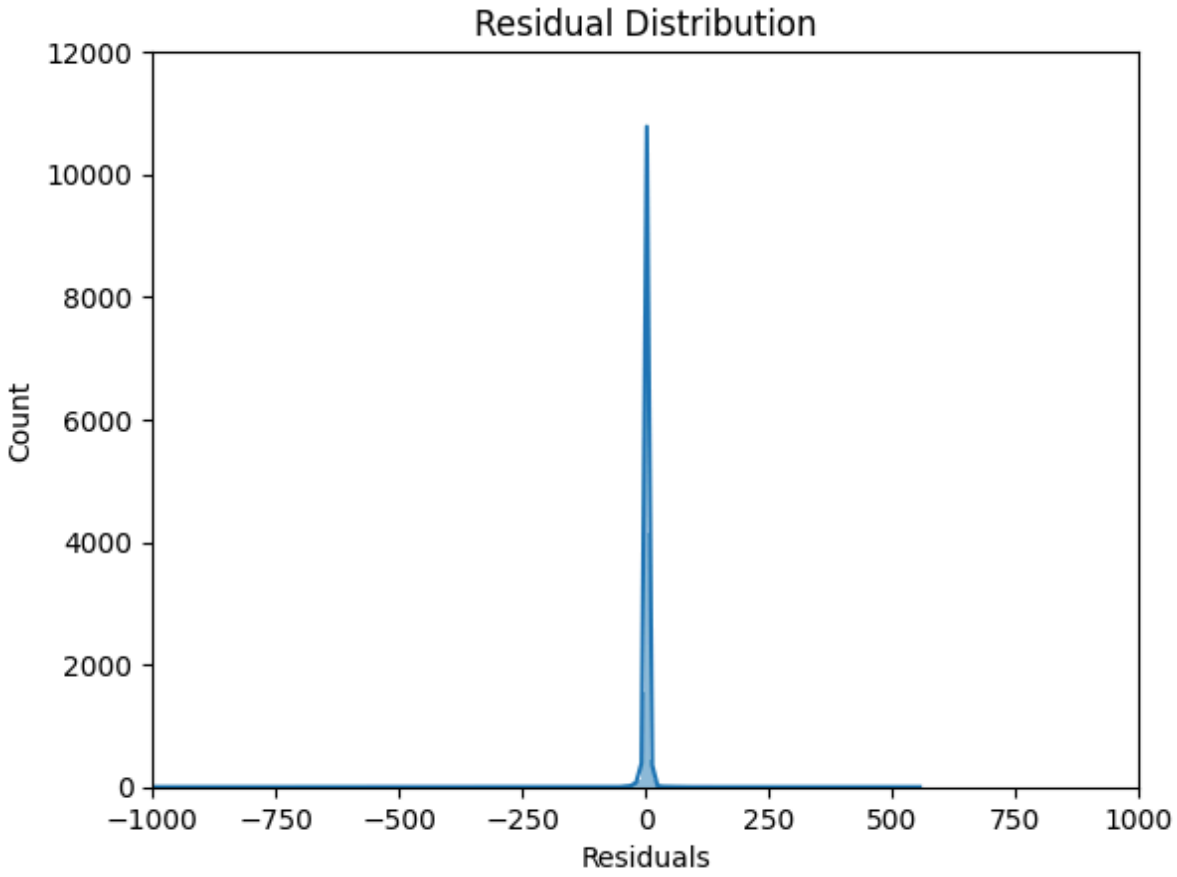
- Dense with relu as activation function for regression problem -> Multiple layers but sequentially reducing number of units
- BatchNormalization: Normalizes activations to stabilize training.
- Dropout: Prevents overfitting by randomly disabling neurons.
- loss: MSE (Mean Squared Error)
- evaluation metric: MAE (Mean Absolute Error)



- loss and val_loss converge to a stable value without diverging.
- val_loss only between 20-40 epochs increasing but neural networks recovers from that and val_los converges around loss at later stage
- Conclusion: Looks like solid nn model



Solid performance with some wrong predictions especially in electric energy consumption outside of range 100-300. After 300 the model predicts too small electric energy consumption compared to actual.



Statistical Summary

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

—-

9. Conclusions

(attempts and first quick brainstorming):

- step by step explanation + reasons of:
    - 1_5-electric-preprocessing_train_test_split
    - 2_0-electric-lazy-model_selection
    - 2_1-electric-model-tuning-trees
    - 2_2-electric-model-final
    - 2_3-electric-model-final-evaluation

Pick ExtraTreeRegressor as “final model” in the report

- good combination of R squared, RMSE
- compared to other model very stable and low compute time
- works well with a lot of features
- good interpretability

—--DRAFT—

Guideline: Classification of the problem

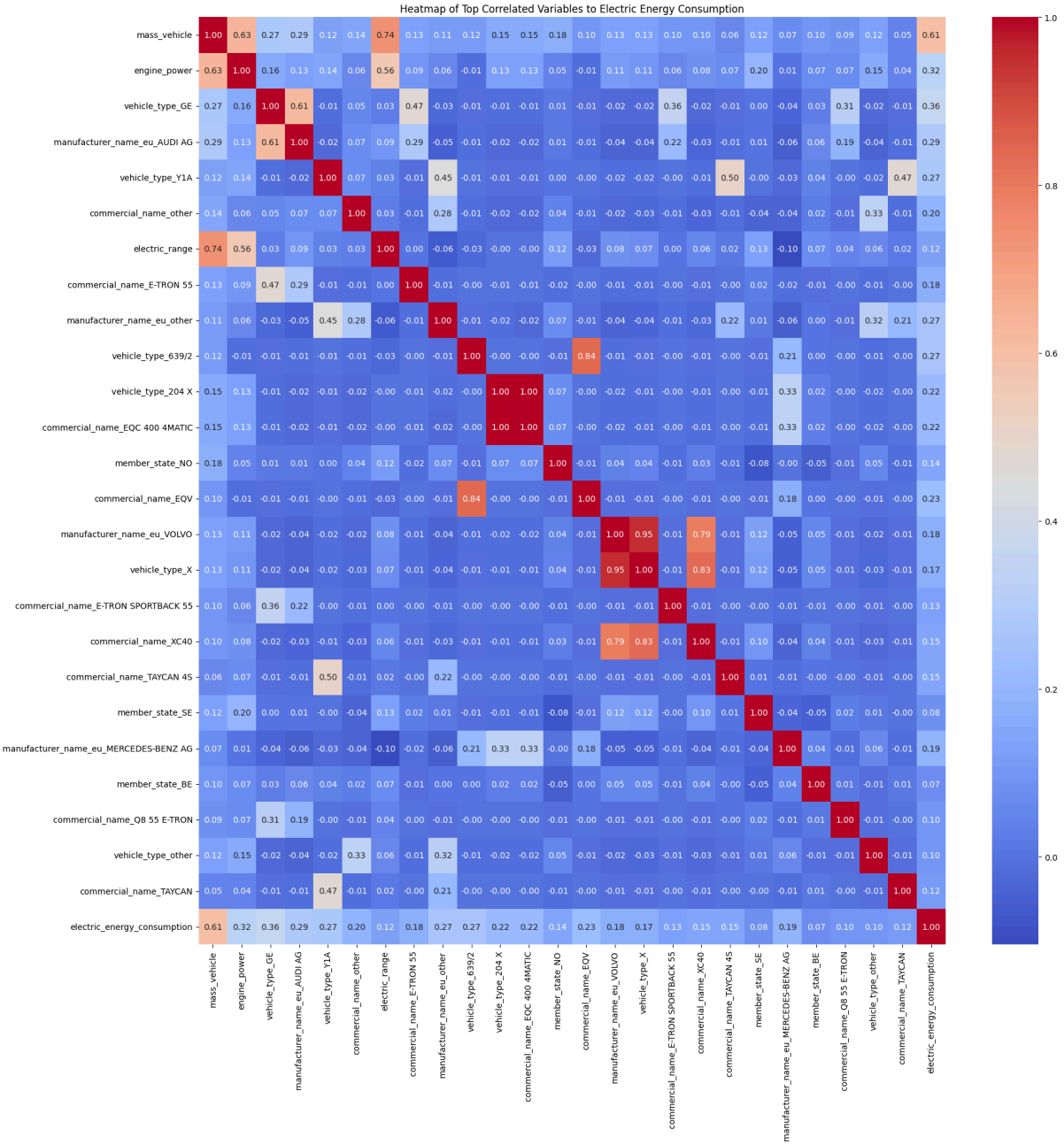
- bla bli blub

Guideline: Model choice and optimization

Developed notebooks which will be used for future app emulation.

- 1_5-electric-preprocessing_train_test_split
- 2_0-electric-lazy-model_selection
- 2_1-electric-model-tuning-trees
- 2_2-electric-model-final

1. Preprocessing for electric to allow us using more features (optimize dataset for electric use case)
    a. further preprocessing on manufacturer name , vehicle type and commercial name
        i. reducing of unique values through grouping
        ii. one hot encoding



    b. Split up dataset into train (70%) and test sample (30%).
    c. Feature scaling of numerical columns.
    d. Random downsampling for lazy predictions down to 10% of dataset size to stay within time constraints and keep compute of model trainings stable for our hardware ressources.

2. LazyRegressor to test dataset and regression-usecase on all standard models

a. Execute LazyRegressor on downsampled electric dataset.

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	4.10	1019.17
LGBMRegressor	0.96	0.96	4.95	11.30
MLPRegressor	0.96	0.96	4.95	1350.02
HistGradientBoostingRegressor	0.95	0.95	5.50	23.23
GradientBoostingRegressor	0.89	0.89	8.19	149.53
BayesianRidge	0.86	0.86	9.27	17.80
RidgeCV	0.86	0.86	9.28	19.11
Ridge	0.86	0.86	9.28	8.82
TransformedTargetRegressor	0.86	0.86	9.28	13.14
LinearRegression	0.86	0.86	9.28	14.63
PoissonRegressor	0.86	0.86	9.29	13.26
LassoCV	0.86	0.86	9.29	112.62
LassoLarsIC	0.85	0.85	9.41	16.77
LassoLarsCV	0.85	0.85	9.41	19.41
ElasticNetCV	0.85	0.85	9.42	44.39
HuberRegressor	0.84	0.84	9.86	102.53
LinearSVR	0.83	0.83	10.04	301.83
OrthogonalMatchingPursuitCV	0.81	0.81	10.66	19.98
OrthogonalMatchingPursuit	0.81	0.81	10.66	8.97
PassiveAggressiveRegressor	0.77	0.77	11.66	12.41
LassoLars	0.76	0.76	11.95	9.03
Lasso	0.76	0.76	11.99	14.59
TweedieRegressor	0.74	0.74	12.41	8.72
RandomForestRegressor	0.74	0.74	12.53	292.67
BaggingRegressor	0.74	0.74	12.54	47.58
GammaRegressor	0.74	0.74	12.55	8.75
DecisionTreeRegressor	0.74	0.74	12.60	12.39
ElasticNet	0.73	0.73	12.69	10.23
RANSACRegressor	0.72	0.72	12.94	32.86
LarsCV	0.71	0.71	13.13	30.83
XGBRegressor	0.67	0.67	14.13	13.01
AdaBoostRegressor	0.40	0.40	19.06	98.53
DummyRegressor	-0.00	-0.00	24.54	7.48

b. Results of LazyRegressor:
    i. Picked the following regressors based on promising values for R-Squared and compute time:
        1. Trees and Forest regressors
        2. Gradient Boosting regressors
        3. Ridge regressors

3. Hyperparameter tuning on promising models
    a. XGBoost (leonelleitebarros@gmail.com results of the models you experimented with)
    b. Trees and Forest regressors
        i. RandomizedSearchCV on ExtraTreeRegressor, ExtraTreesRegressor, RandomForestRegressor, DecisionTreeRegressor

Model	Best Params	Best CV Score	Test R^2 Score
ExtraTreeRegressor	min_samples_split: 2, min_samples_leaf: 2, max_features: None, max_depth: 20	0.97	0.96
ExtraTreesRegressor	n_estimators: 50, min_samples_split: 10, min_samples_leaf: 1, max_features: None,	0.89	0.89

	max_depth: 10		
RandomForestRegressor	n_estimators: 50, min_samples_split: 10, min_samples_leaf: 1, max_features: None, max_depth: 10	0.95	0.70
DecisionTreeRegressor	min_samples_split: 5, min_samples_leaf: 2, max_features: None, max_depth: 20	0.98	0.73

4. KFold Cross-validation on models with parameters based on RandomizedSearchCV hyperparamter tuning.

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

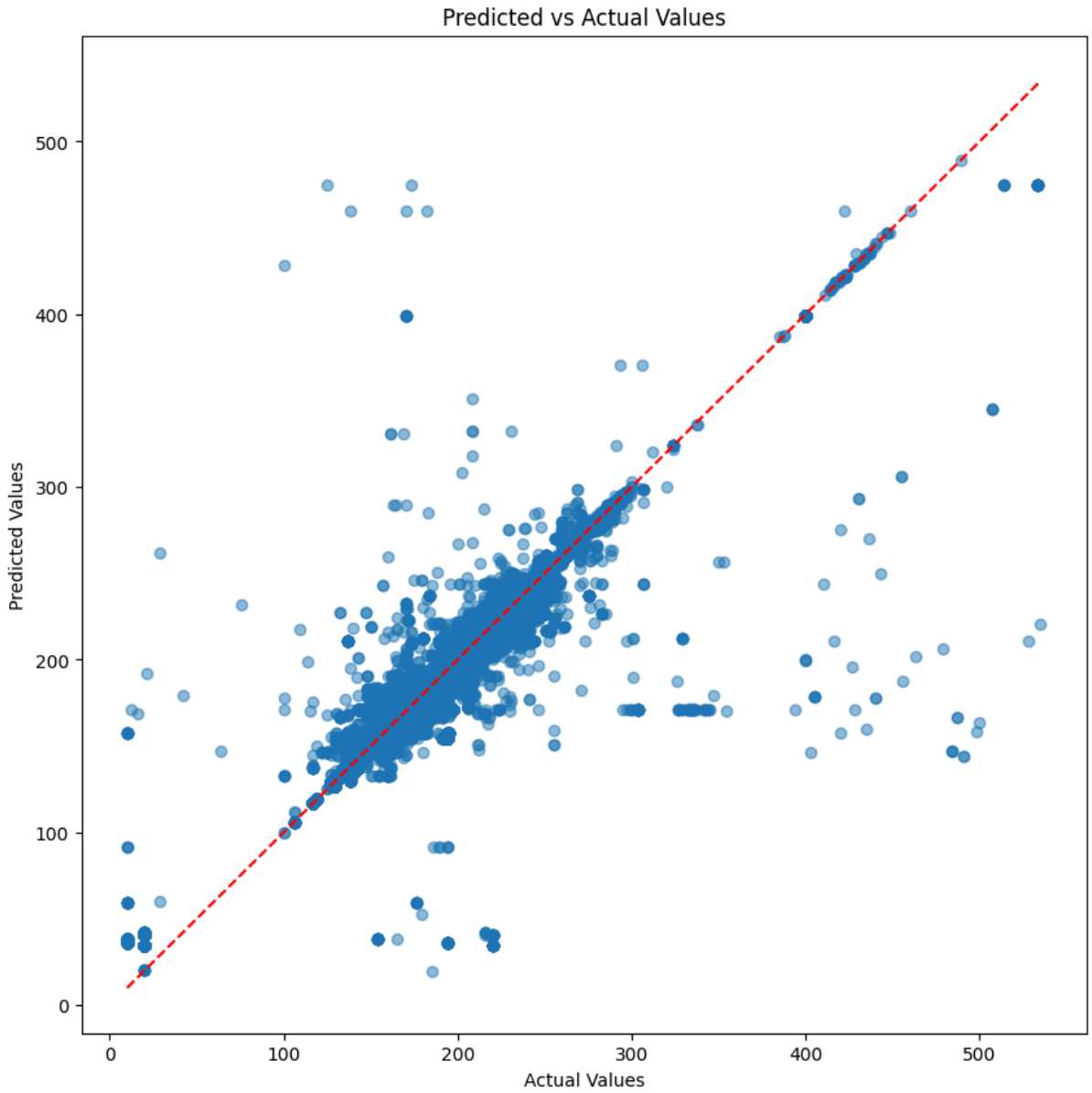
   a. picked the model with best RandomizedSearchCV result + best generalization + score within kfold cross-validation

ExtraTreeRegressor showed best performance in RandomizedSearchCV and KFold Cross-validation with CV Score of 0.97, R^2 Score of 0.96, R^2 Score Average of 0.95 and R^2 Variance of 0.00.

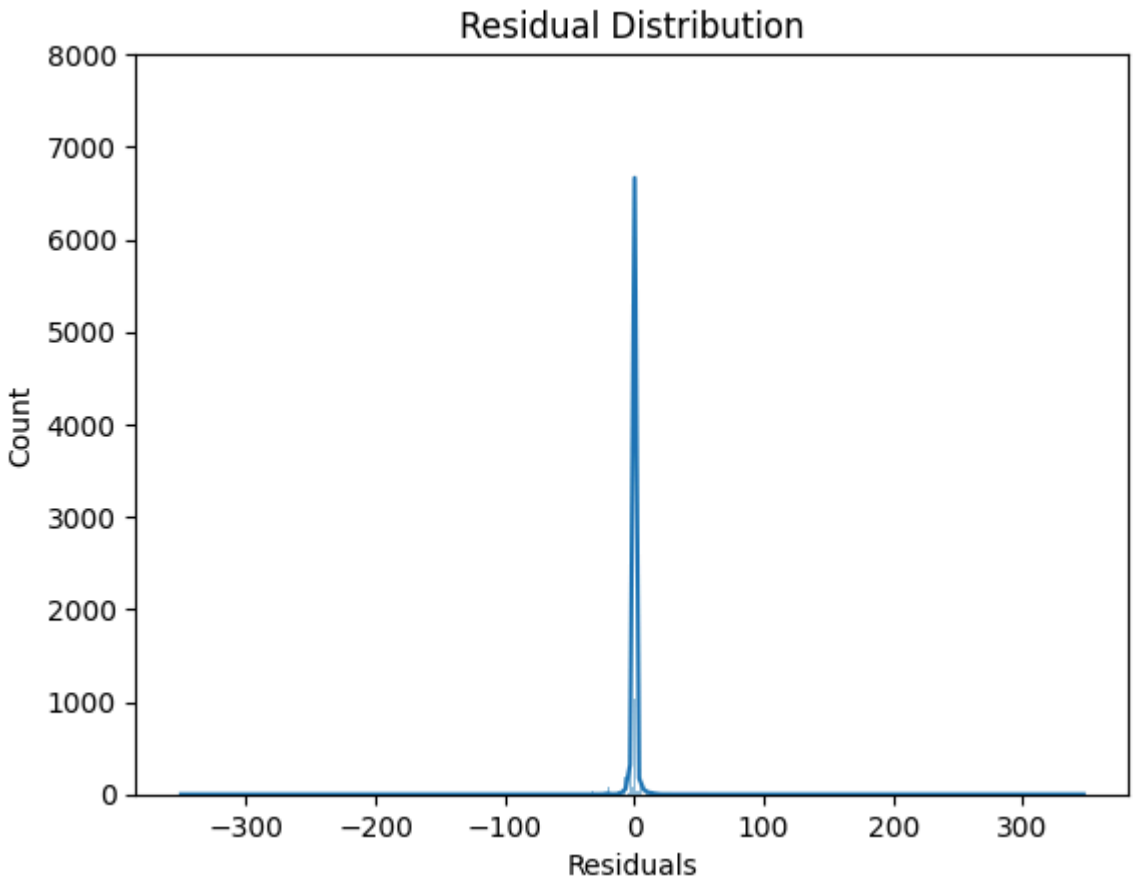
5. Trained ExtraTreeRegressor model with tuned hyperparameters on entire dataset +
6. train model with deep learning and analyse (todo)

Guideline: Interpretation of results

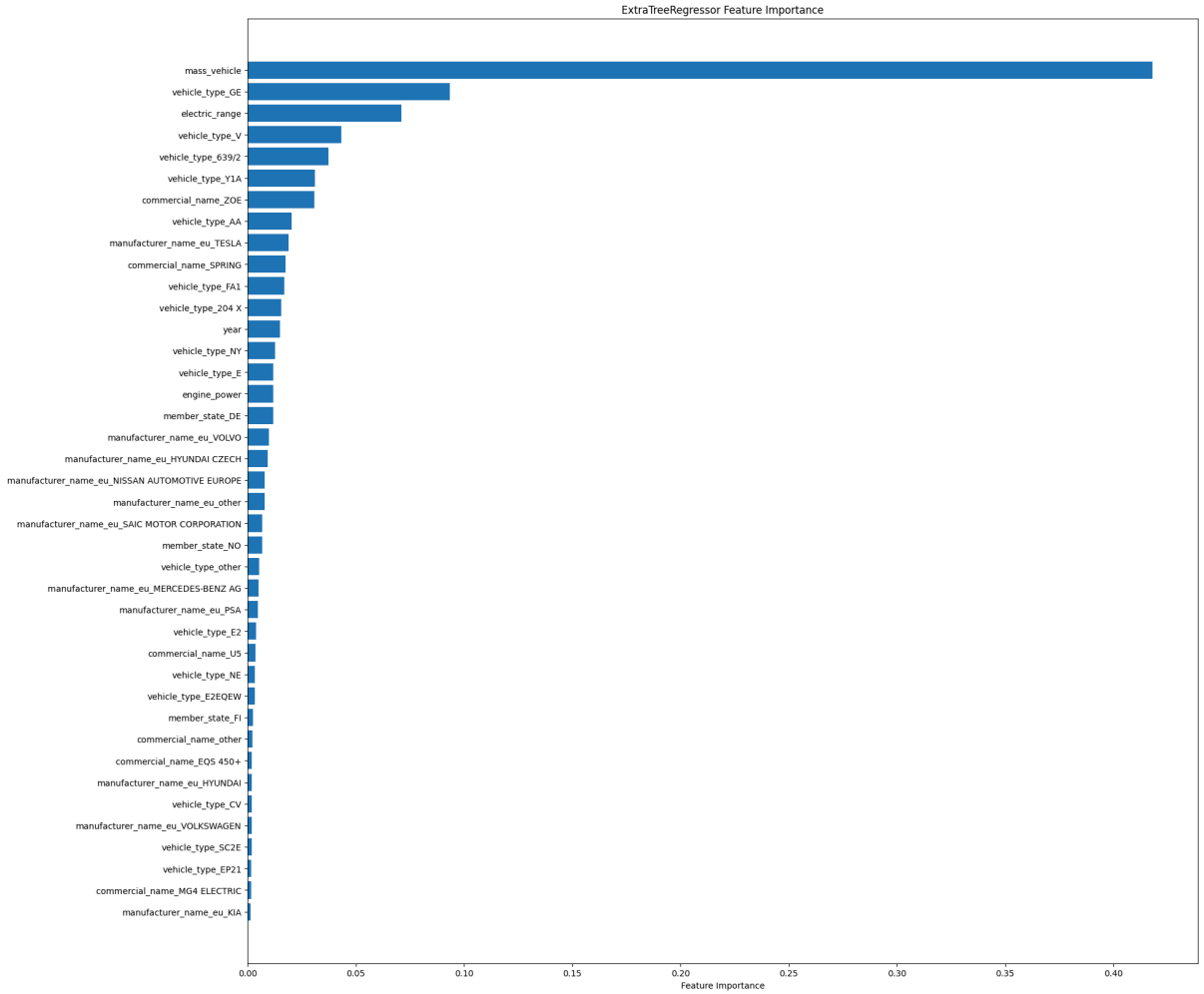
1. evaluation of model (todo finalize)
   a. R-Squared: 0.9706514944335698
   b. Adjusted R-Squared: 0.9706438238558385
   c. MSE: 17.6157250532277
   d. RMSE: 4.197109130488234
   e. RMSE error percentage compared to mean: 2.50% within threshold of 5%



For some predictions the predicted values differ from actual values. The residual plot shows that such false predictions are small in quantity. Majority of predictions show small residuals and are close to actual values.



2. interpreatability of model (todo)



3. describe what improvements we see / want to make (e.g. some of the preprocessing steps you did last week leonelleitebarros@gmail.com)

Part 2 - Predicting Combustion Fuel Consumption

1. Basic description of the task

As asset we got the the data of new cars registered in the EU-Market and their features containing the weltp_test_mass of the cars, the vehicle_type, fuel_type, engine_power, engine_capacity, specific_co2_emissions for wltp-testcycle, fuel_consumption (and more, not enumerated completely). After preprocessing the data as described in report_1, our task was to examine different models to predict either fuel_consumption, specific_CO2_emissions or the electric_energy_consumption.

Since the datasets differed for combustion engines and electric engines in the features fuel_consumption or specific-CO2-Emissions for combustion engined cars and electric_energy_consumption for electric cars, we decided to split the dataset up into one dataset for combustion engines and one dataset for electric engines.
We distributed the modelling task for the combustion engines to Thomas and me. The modelling for the electric engined cars was done by Philipp and Leonel.

2. Type of machine learning Problem

The target is a quantity, so we concentrated on regression models.

3. Main performance metrics used

To evaluate the performance of the models I choose the metrics score and rmse (root-mean-squared-error), which can be easily retrieved via python functions and which are easy to understand.

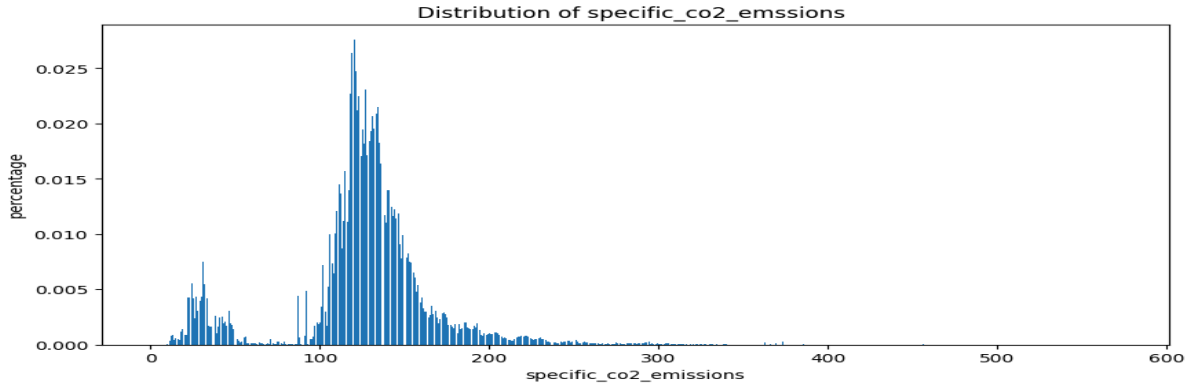
4. Algorithms used

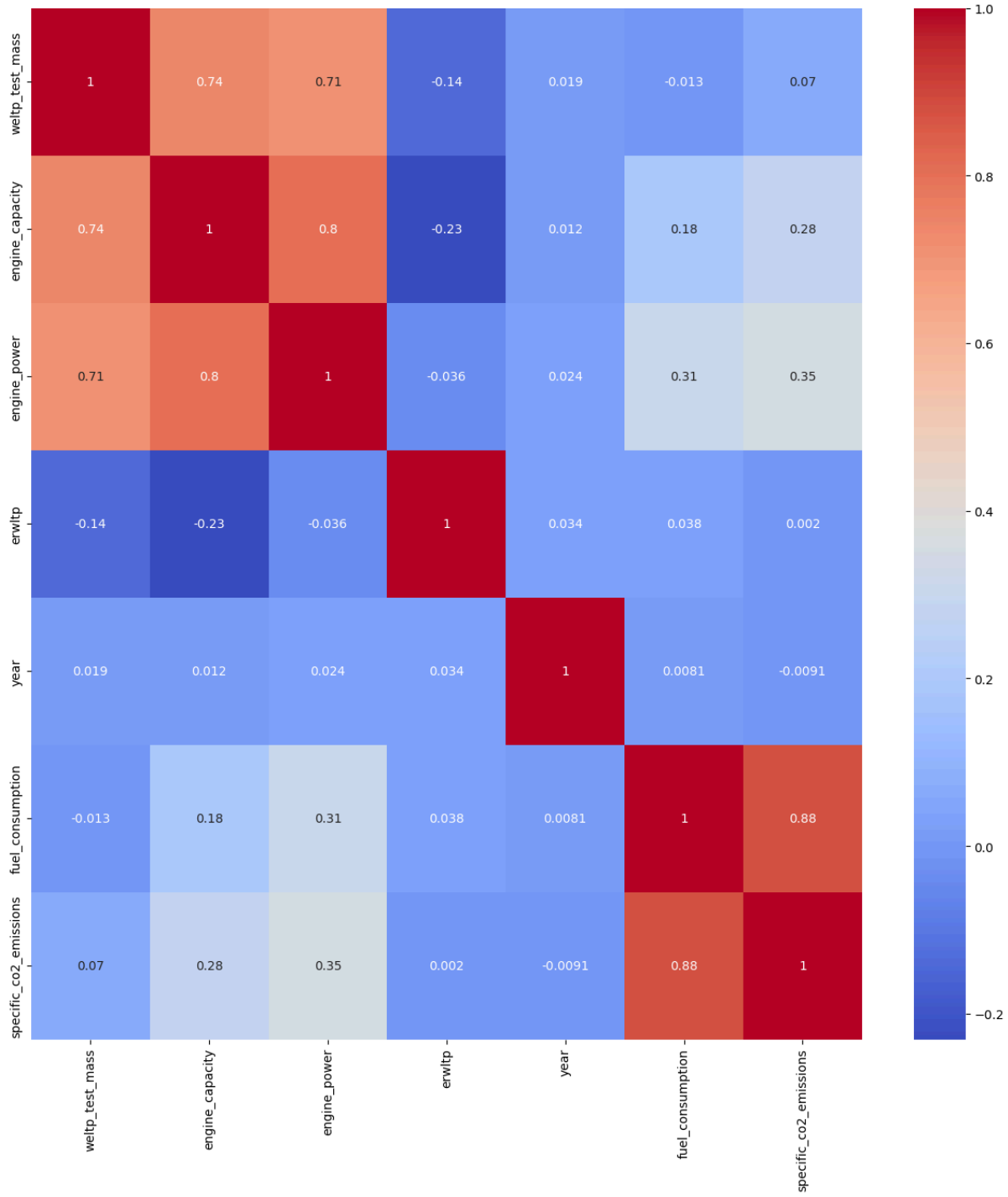
After cleaning the dataset of non existent values, I kept following features for the modelling process:

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

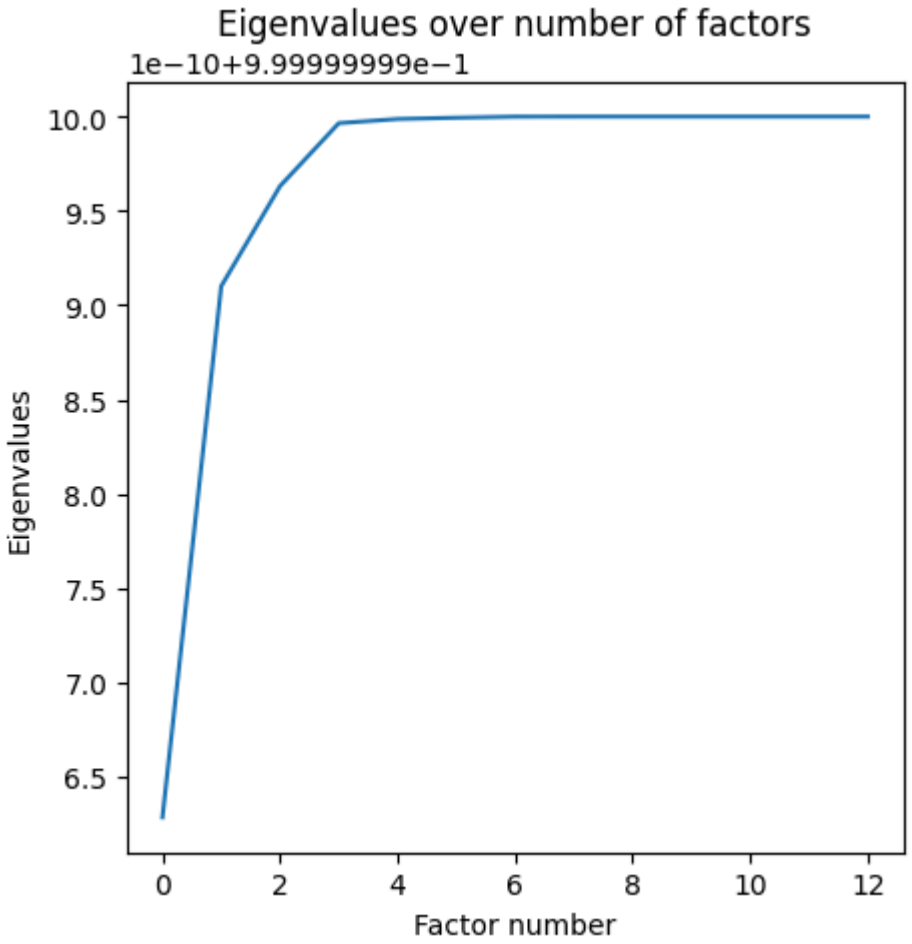
I divided the features after variable type into numerical columns (type int64 or float64) and categorical features (type object)
Numerical columns were standardized with StandardScaler. This was done to get the values to a unified scale between 0 and 1.
Categorical columns were labelencoded by LabelEncoder.
I also displayed a correlation heatmap.

The correlation heatmap led to the elimination of one of the two possible targets out of the dataset. Specific-co2-emmissions and fuel-consumption can be calculated out of each other. So I decided to take specific-co2-emissions as target and eliminate fuel-consumption out of the dataset.





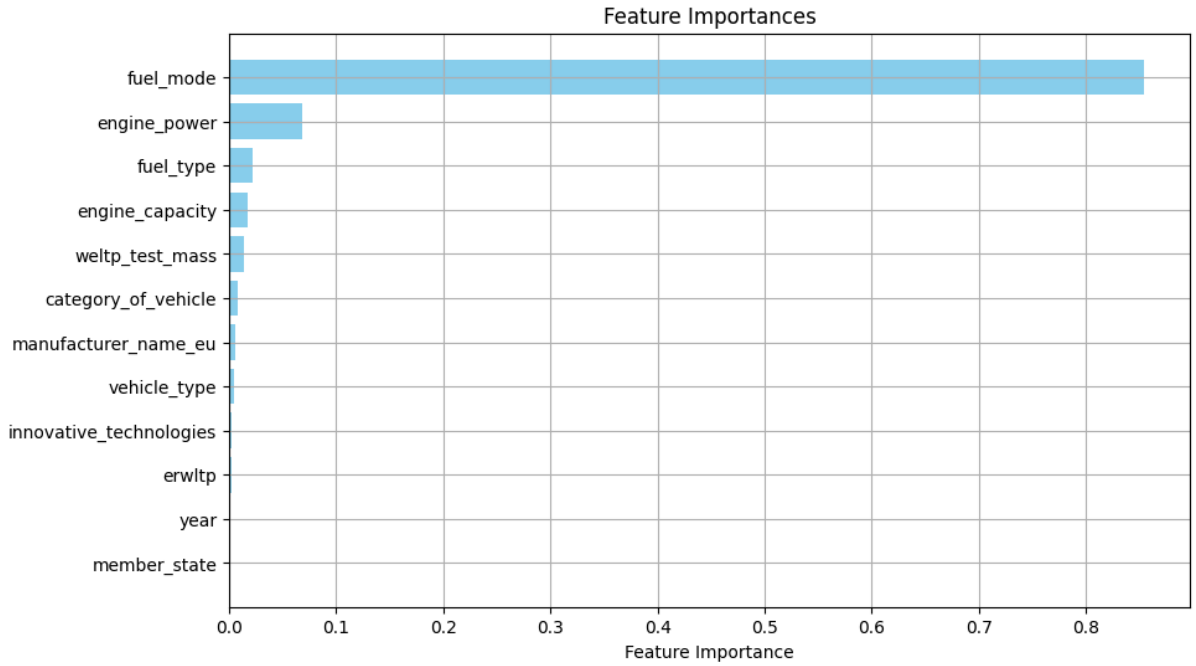
I further used train_test_split to divide the samples into a training and a test set. I kept 20% of the data for testing. First I started with a simple LinearRegressionModel. This resulted in a training and testing score of 0,4 and a rmse of 30g/km. (mean of target: specific_CO2-emission: 126g/km)
A LassoCV led to similar unconvincing results.
The XGBRegressor delivered the best results so far with a score of 0,997 and an rsme of 4g/km.
Further I tried out a GridSearchCV. This required further data preparation due to runtime problems. With the full size of my training dataset, even after more than 10h runtime no result was in sight. So I drastically reduced the training data to 1 percent of the original set. I additionally used a PCA.



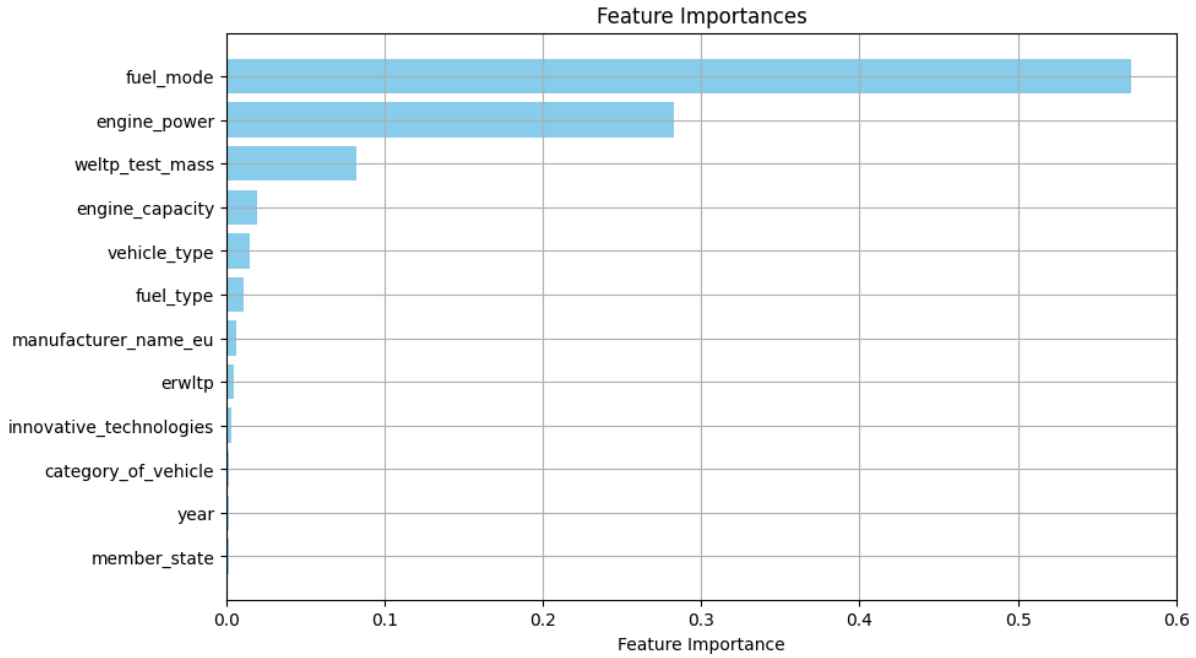
With the reduced dataset, I ran a RandomizedGridSearchCV which didn't deliver a final result due to too long runtime. But the so far delivered intermediate result led me to sort out LinearRegression.
I then supported my decision by running another RandomizedSearchCV with the models XGBRegressor, DecisionTreeRegressor and RandomForestRegressor. I ended up throwing out the LinearRegressor with the worst results of the three models. A LinearRegressor without any parameters reached an rsme of 31,8 g/100km for the specific CO2-Emission.
XGBRegressor with n_estimators =1000 and learning_rate = 0,3 reached an rsme of 3,5716 g/100km. (2% of the mean CO2-emission)
DecisionTreeRegressor with max_depth = None reached 4,9961 g/100km (4% of the mean CO2-emission)

5. Interpretation of results

In order to analyze the best models I displayed the sorted feature importances for the XGBRegressor. This led to the result that "fuel_mode" is the most relevant feature for this model followed with large distance by "engine_power", "engine_capacity" and "wltp_testmass".



I also displayed the sorted feature importances for the DecisionTreeRegressor.
Here also “fuel_mode” appeared as most relevant feature followed with not as large distance as above by “engine_power” and “wltp_testmass”
Then I trained the XGBRegressor with the best parameters (n_estimators = 1000, learning_rate = 0,1) and with 80% of the dataset in order to calculate as further metrics mean-absolute-error, mean-squared_error of the test_data.
This resulted in 1,4015 g/100km as absolute error, 6,1988g/km as mean-squared-error and 0,9963 as R squared.



I followed the same process with the DecisionTreeRegressor leading to a mean-absolute error of 0,6781g/100km, a mean-squared-error of 3,6499g/100km and R squared of 0,9977.
For the DecisionTreeRegressor I also tried to edit the decision-tree. (again a runtime problem occurred)

As a last part of the project, I built up a simple sequential neuronal Network with keras. with one input layer and three dense layers. The first dense layer contained 64 neurons and reLU as activation function. In between I added a dropout layer with dropout 0,5. The output layer has, 32 neurons with a softmax activation function. The last layer contained only one neuron.
Result is still open.

Part 3 - Predicting Combustion Fuel Types (Classification)

1. Basic description of the task

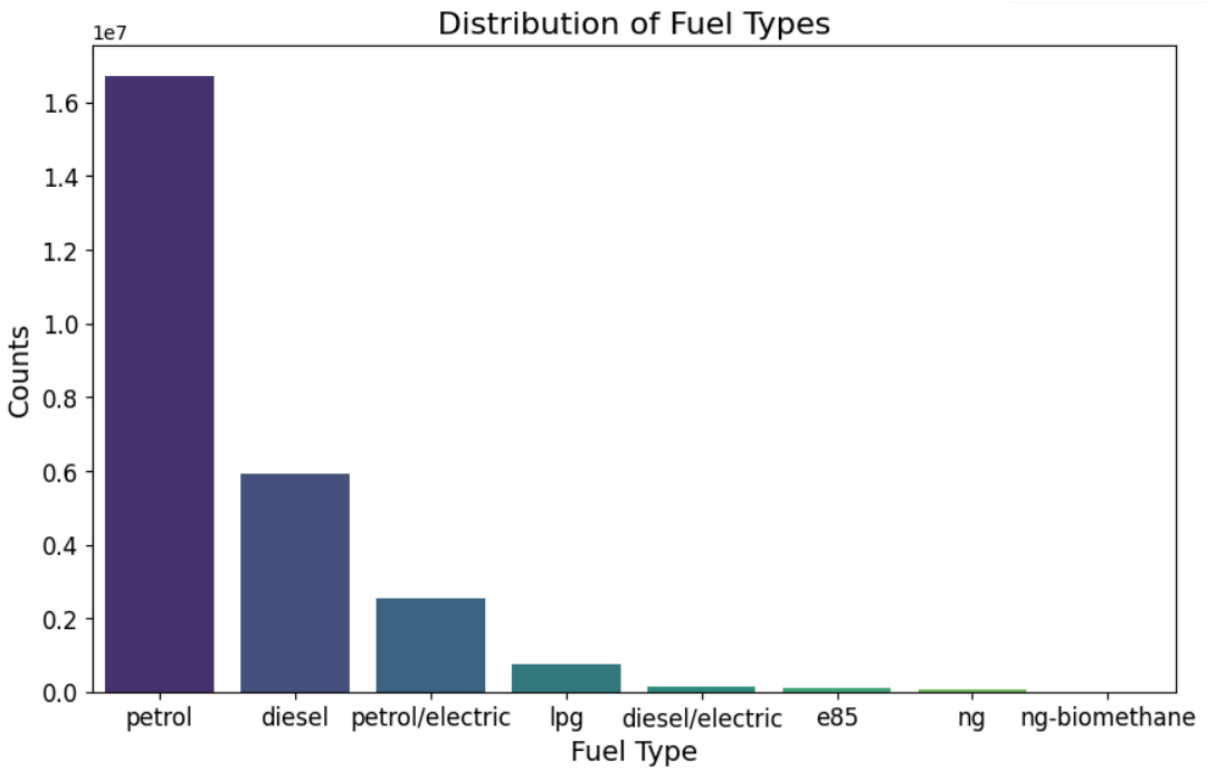
In this analysis, I wanted to focus on **predicting the type of fuel** used by vehicles based on various performance metrics. The interest was more scientific and explanatory in nature, and less practical oriented, though maybe I could find something that would aid in the main Regression task of this project.
The data is derived from the preprocessing files in */app_emulation* by Philipp. He segmented the dataset to emphasize combustion-related features for Richard and me.

I further worked on the dataset and reduced it over the course of this project stage.
As a start, I prioritized numerical variables that directly influence vehicle capabilities, such as mass, engine power, and fuel consumption.
We removed categorical variables that could introduce bias, such as vehicle name, manufacturer, and the country of the manufacturer, to maintain an unbiased approach in our predictions.
I also excluded variables from earlier preprocessing steps that still pertained to electric vehicle features, streamlining the dataset to focus strictly on combustion engines.
Following a coaching session with Romain, I was also advised to exclude variables directly associated with environmental impact, such as CO2 emissions, and focus solely on the vehicle characteristics themselves.
We also made a decision as a team between keeping either *mass_vehicle* and *weltp_test_mass* out of redundancy, and dropped the latter one.

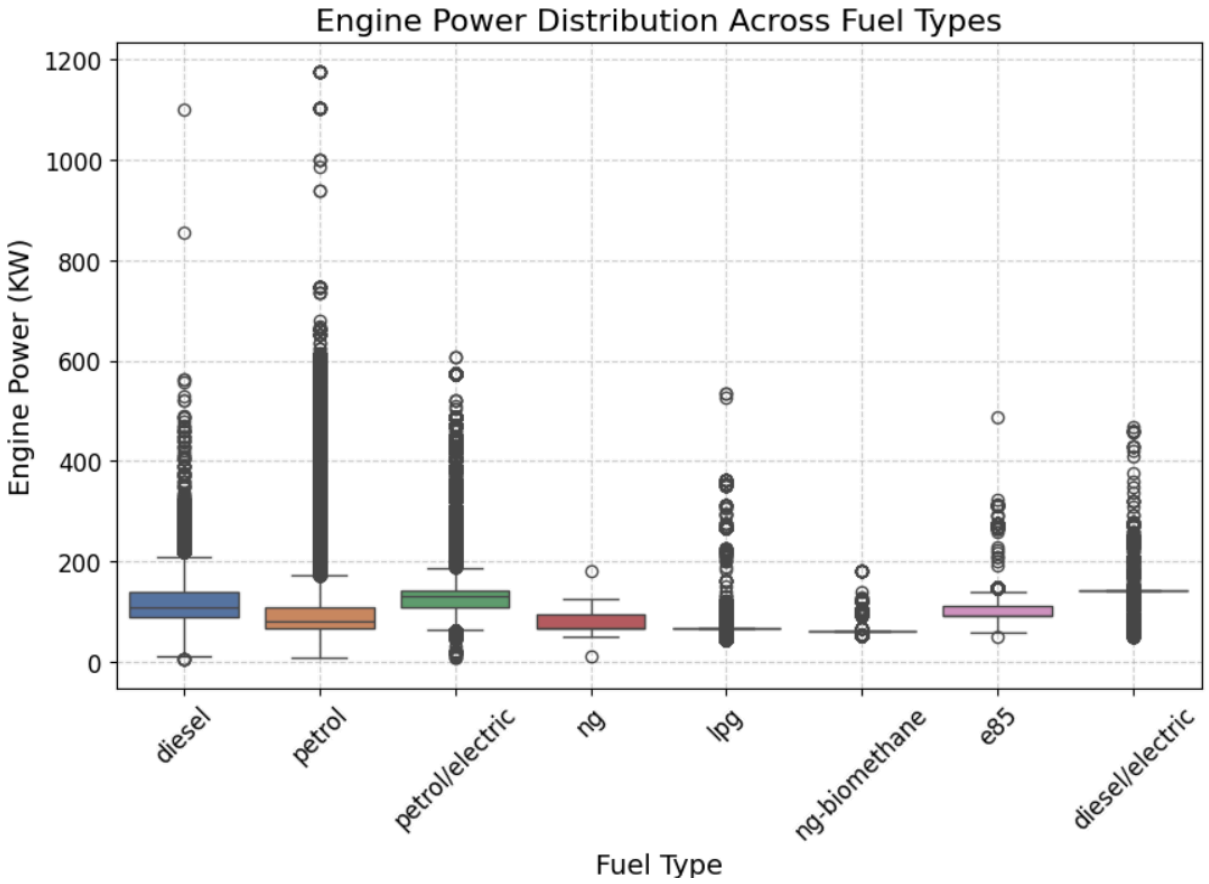
The primary goal of this analysis, as already mentioned, is to classify vehicles based on their fuel type. This will be achieved by applying **Classification machine learning models** to predict fuel types like petrol, diesel, and various hybrids.

2. First insight through plots

This report part begins by examining the distribution of fuel types within the dataset, setting the stage for further detailed analysis on how different variables influence fuel type classification.



We can see that the **classic fossil fuels are very prevalent in the data**, and more modern or gas-based fuels are in the vast minority. This could influence some decisions later down the line.
Now to find out whether my approach was relevant and the data is actually distinguished enough for a prediction model to do meaningful work, I wanted to check the different vehicular variables. One of them was the engine power, which I decided to plot in a graph:



We see a medium variance in the data and very distinguishable outliers between the fuel types. This difference suggests that engine power could potentially serve as a robust feature for distinguishing between fuel types, and the general approach was promising.

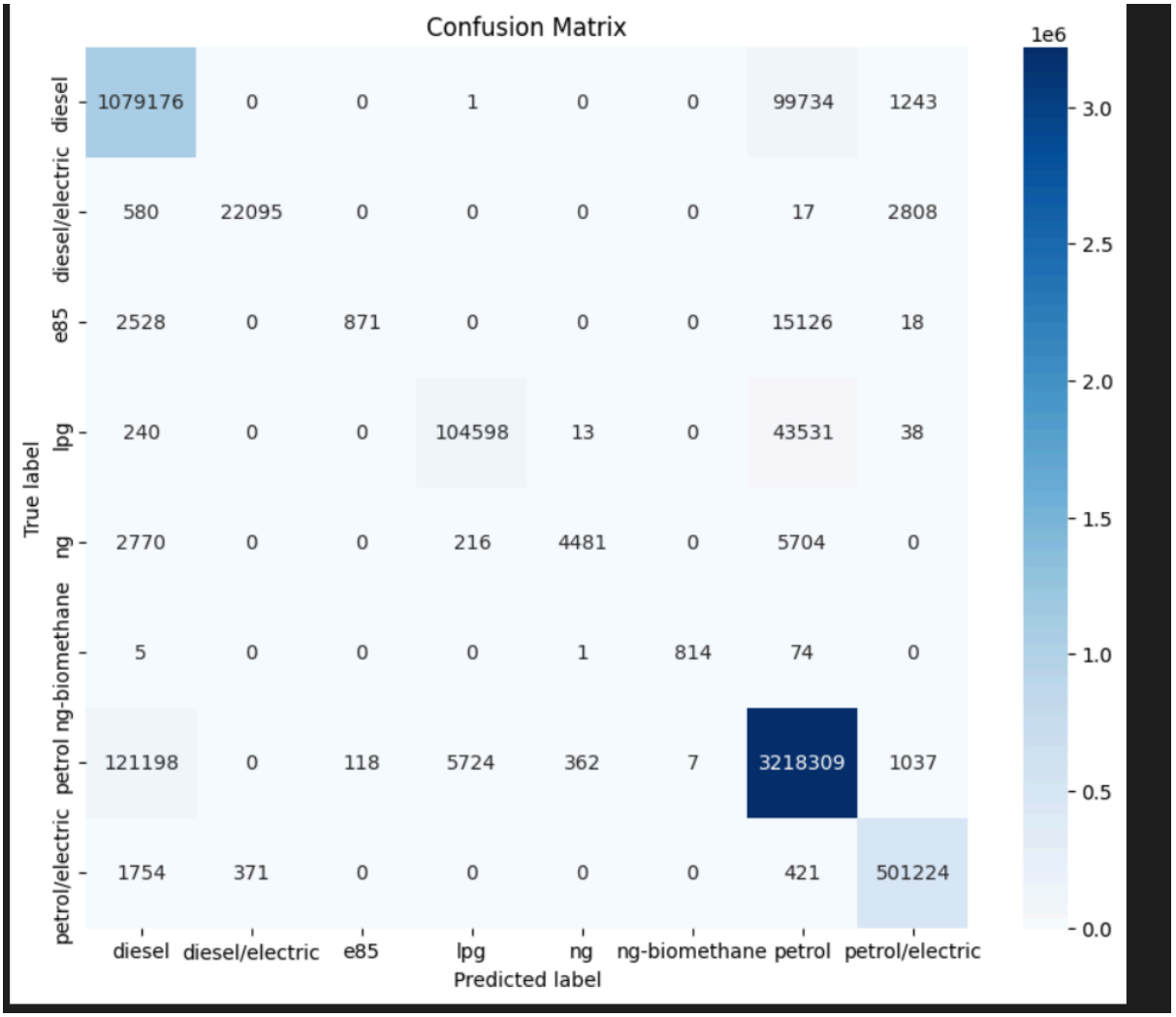
3. Tree Models and Issues with Performance and Distribution

For the modeling, I decided to use **Accuracy as the main performance metric**, to gauge basic effectiveness of the model in categorizing fuel types correctly. If there was time, I also wanted to include other metrics.

I wanted to **focus on tree models, due to their high interpretability**, which was my goal with this sub-project. **Random Forest** was the first I tried, however, due to the extensive computation time and the model seemingly being overwhelmed in managing a larger dataset, the approach needed reevaluation. Later as a team we noticed we all faced these challenges and collectively decided on a reduced dataset to combat this problem. Having included 3 years worth of data from the EU endpoint proved to be more of a curse than a boon.

But at this project stage, I didn't know my colleagues were facing the same issues, and I shifted subsequently to **XGBoost**. I selected it as the preferred model, as XGBoost offers good handling of larger datasets and maintains reasonable interpretability, due to also being tree-based, which is essential for my more scientific inquiry.
After also applying **Dimension Reduction in the form of PCA** (retaining 0.85 components) to reduce the runtime further, XGBoost proved efficient and effective, without sacrificing accuracy.

This was the first confusion matrix I generated from a model runthrough:



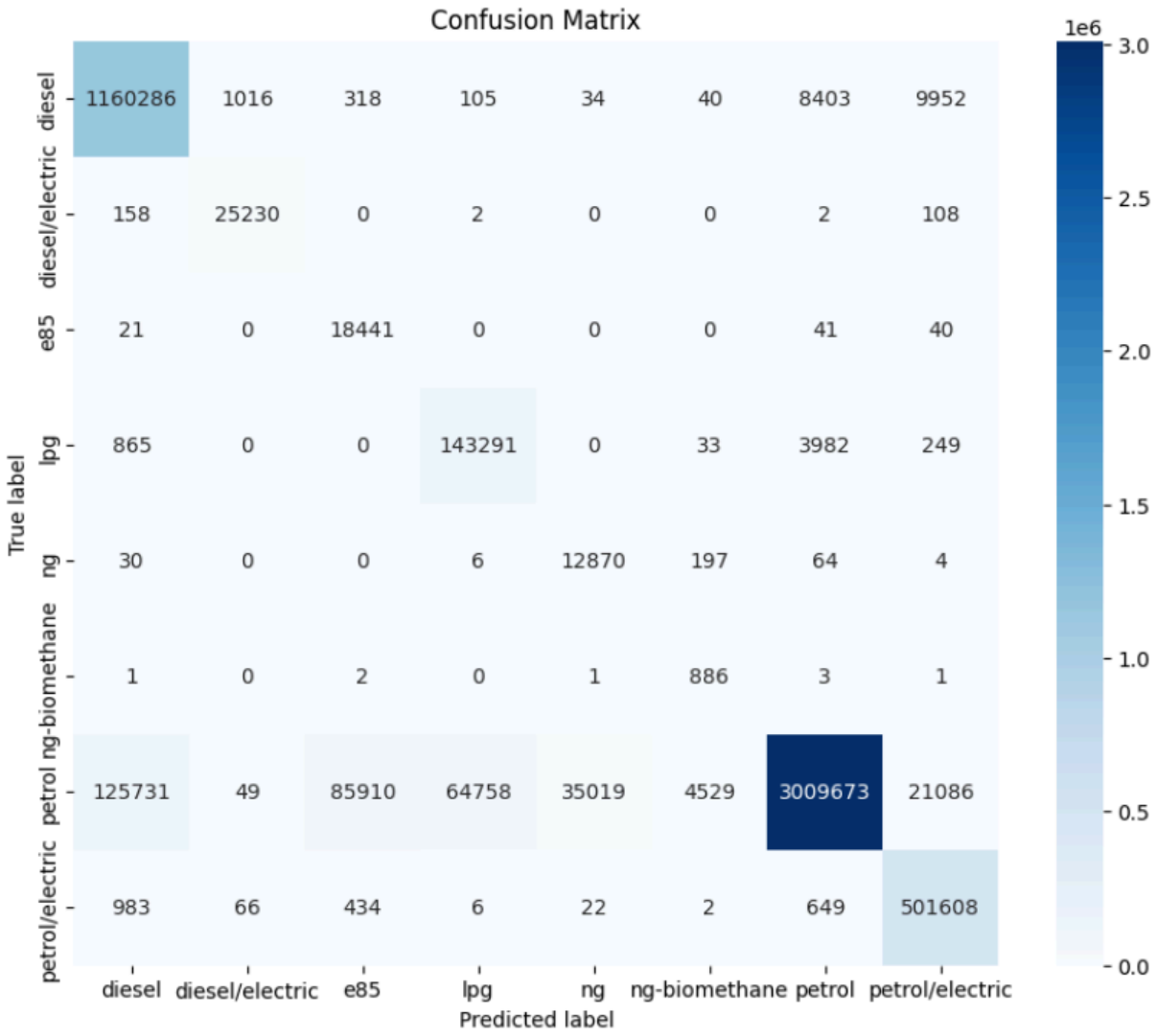
Single Fuel Accuracies:
Diesel: 91.45%
Diesel/Electric: 86.65%
E85: 4.70%
LPG: 70.01%
NG-Biomethane: 34.59%
Petrol: 94.15%
Petrol/Electric: 99.33%

The substantial variance in accuracy across the classes, particularly the **lower accuracy for E85 and NG-Biomethane**, with LPG also having only moderate accuracy, correctly (as we already know) suggests an imbalance in the dataset - LPG and also E85 and NG-Biomethane had a lot fewer samples compared to other classes such as Diesel and Petrol.

Harkening back to the first graph of fuel distribution, I decided to employ **SMOTE** as a technique to combat these challenges.

SMOTE could help create new examples in the minority classes (like E85 and NG-Biomethane) to balance the distribution, so the model could perform well for all fuel types and not overfit to the majority classes.

This was the confusion matrix generated after a XGBoost runthrough with SMOTEd data.



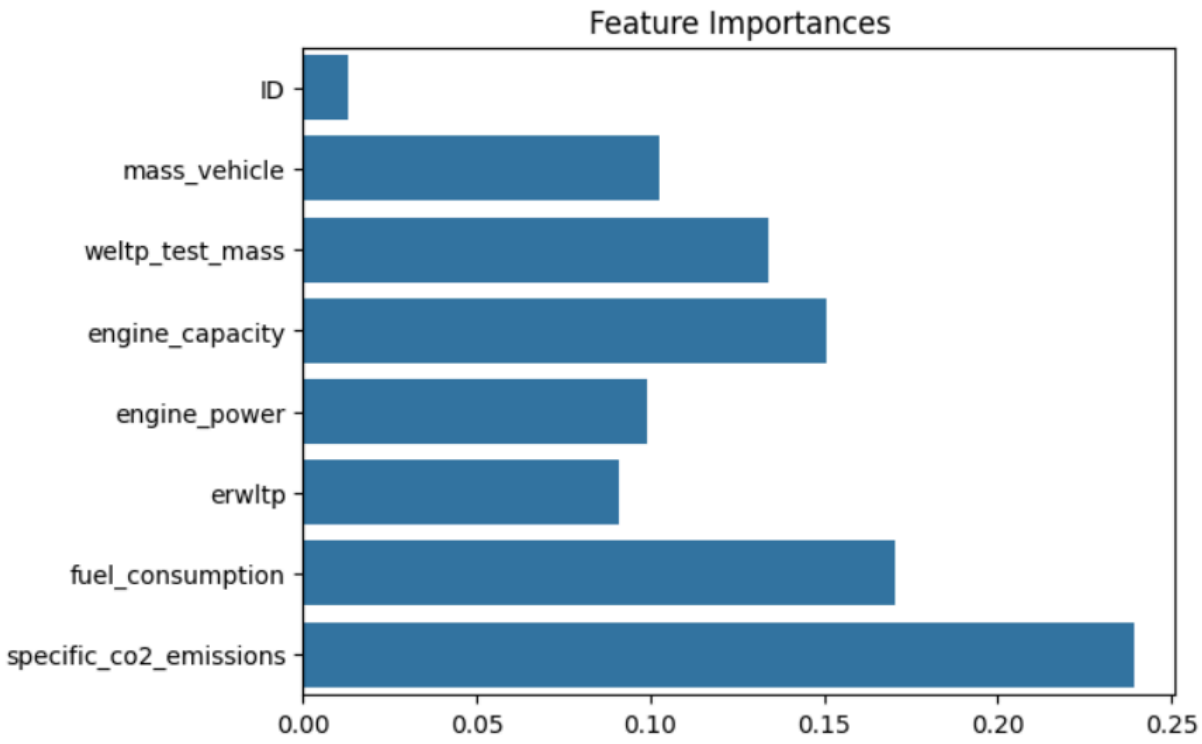
Single Fuel Accuracies:
Diesel: 98.57%
Diesel/Electric: 98.94%
E85: 99.45%
LPG: 96.54%
NG-Biomethane: 97.76%
Petrol: 88.44%
Petrol/Electric: 99.33%

As we can see, **accuracy improved a lot, especially for the beforehand underrepresented E85, LPG and NG-Biomethane**, which now were better represented in the dataset. Definitely a successful employment of SMOTE. Petrol, the most prevalent datatype, took a small hit in accuracy. But for a

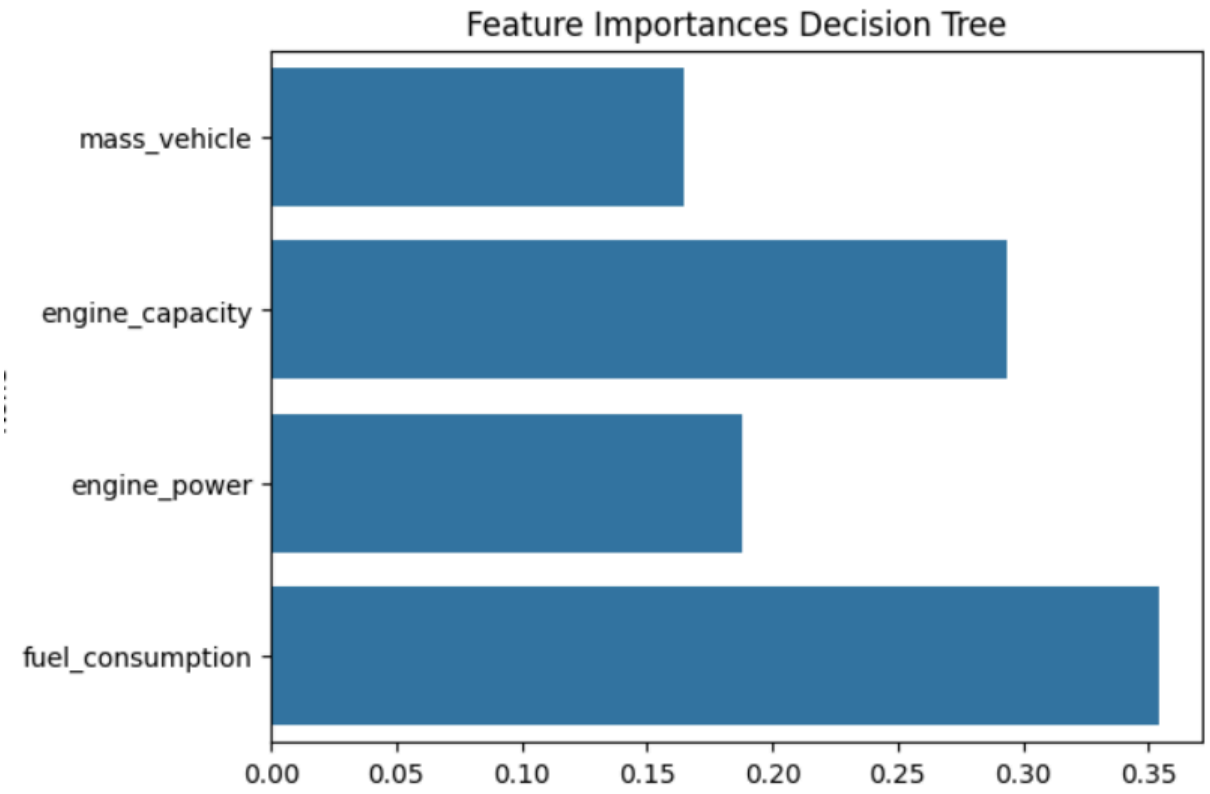
universally useful classification model that doesn't overfit to Petrol (and classifies that almost always correctly, but the minority classes wrongly), this was definitely a good result.

4. Feature Importance Analysis

For a meaningful Feature Importance analysis, we needed XGBoost to work without PCA in order to retain the features properly. Even then though, I had to backward-encode the target variable, as XGBoost requires a classification with a categorical target to be encoded.



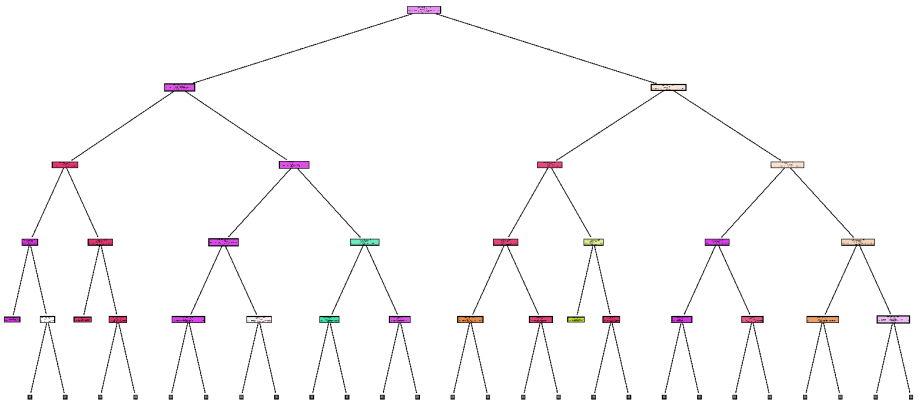
As you can see, this Graph was done in an earlier stage of modeling, still retaining the two mass variables, emission variables, and even ID.
What is interesting is that in comparison with the following later graph here, where of the remaining variables, mass is the lowest in importance…



…while in the graph beforehand, the *mass_vehicle* variable was more important than engine power (and *weltp_test_mass* even more important).

5. Tree Graphs with Decision Tree

(The graph is truncated due to computability reasons. Not because of the runtime of a Python script, but because a full resolution view of the Tree Graph with hundreds of nodes is simply not displayable on my machine and even as an SVG file is over 100 megabytes in size.)



A tree graph for interpretability was actually one of the first ideas I had in mind, but only later realisable through the reduced data and lower runtime, even for other Tree-based models than XGBoost. This graph was done via a **Decision Tree** model.
In all honesty, I did not get much information out of the graph, other than getting technical insight on how such Tree Graphs work. And that, as would be expected, **variables like engine_capacity and fuel_consumption, which are variables with higher feature importance, show up more early and more often in the tree nodes to make distinctions**.

6. Limited Deep Learning Classification: Keras

As a final step, and with severe time constraints, I did a very rudimentary implementation of a deep learning model, with a very limited epoch amount. But I wanted to make at least a simple comparison in accuracy with the machine learning models.
Categorical Crossentropy was used because we have a deep learning problem with Classification as the base, using categorical target variables that can not be reduced to a binary Classification problem.

The end result after a simple runthrough with **Deep Learning through a Keras model** were:
Test Loss: 0.034
Test Accuracy: 0.989

This result, judging at least by the accuracy, is keeping up with the Classification models.
With more time, it would be interesting to explore more about using Optimization and also Interpretability Techniques on this Deep Learning approach to dive in even further into the topic.

7. Possible Next Steps

The Classification for fuel types worked very well using the given vehicular characteristics in the dataset.
As next steps to improve model performance even more, we could go into Grid Search/Random Search to fine-tune model parameters for the prominent contestant XGBoost.
The insight on the feature importance graph, with the relevance of the different explanatory variables shifting **might put into question that we removed wltp_testmass in favor of test_mass**, so we could do further study to see if that assumption actually holds weight, doing more statistical analysis of model runs by adding and removing different variables, and re-adding weltp_test_mass and trying some model runs without test_mass.