

Final report

CO2 Emissions by vehicles

Group Members: Alexander Perez , Andreas Lathan, Victor Weissenbach, Tillmann Stralka

1. Introduction

a) Project Purpose and Goals: This project focuses on using machine learning to help the automotive industry meet the EU's 2035 target of 0 g CO₂/km for passenger cars and vans. By analyzing extensive vehicle data, machine learning models can identify factors influencing CO₂ emissions, aiding manufacturers in designing low-emission vehicles. This ensures compliance with regulations, reduces penalties, and enhances brand reputation. The project also aims to optimize production strategies by enabling early design adjustments, especially as the industry shifts towards zero-emission vehicles and considers alternative energy sources like hydrogen or electricity for appliances in motorhomes (as a collateral effect).

b) Technical Goals include preprocessing data, selecting features, training models, and evaluating results. Real-time tools will be developed to predict CO₂ emissions in new designs. The project will also explore how innovative technologies for CO₂ reduction might be applied across different industries.

c) Economic Benefits involve avoiding penalties, cutting redesign costs, enabling competitive pricing, accessing government incentives, and boosting marketing through eco-friendly recognition.

d) Scientific Contribution includes identifying emission-related data patterns and supporting sustainable vehicle tech development, while acknowledging the project's limited scope: focusing only on CO₂ emissions during vehicle operation, not the entire product lifecycle. The main objectives are to find correlations between vehicle features and CO₂ emissions and to accurately predict emissions based on technical characteristics.

The primary dataset for this project is the **EU (EEA) dataset**, named "*CO2 emissions from new passenger cars*", provided by the European Environment Agency. It contains CO₂ emission data for newly registered cars across 30 countries from 2010 to 2023. This dataset is substantial, initially over 16 GB, but was reduced through data transformation and consolidation.

Link: [EEA Europe](https://data.eea.europa.eu/dataset/co2-emissions-from-new-passenger-cars)

2. Preprocessing

The following section describes the final data processing performed with the original dataset from the EEA_Europe source. Several side quests and supplementary investigations were necessary to develop these process steps, though they are not included in the final notebook. The following section explains what was done and why.

2.1 Initial loading and preprocessing of raw data:

First the data was downloaded in a per year form in csv format from the European Environment Agency (EEA):

<https://www.eea.europa.eu/en/datahub/datahubitem-view/fa8b1229-3db6-495d-b18e-9c9b3267c02b>

As the data was too large to merge into a single dataframe right away it was first handled on a per year basis (years 2010 to 2015 were handled as one file as the datasets were much smaller in the earlier years).

For each year the csv files were read into a dataframe with datatypes specified as categorical for string type columns and appropriate initial numerical datatypes for numerical columns, so that no data loss occurred. The datatypes were then further optimized for memory usage by downcasting numerical columns.

To further optimize memory usage the datasets were compressed by removing “duplicates” and instead storing the information about the frequency of “duplicate” rows (i.e. cars) in a newly calculated column named “#identical occurrences”, later called “frequency”. “Duplicate” is set in quotation marks here, because the dataset does not contain any true duplicates. Every row at least has a unique ID and in the later years every individual car has a specific “date of registration”. These two columns were disregarded for identifying our “duplicates” as neither date of registration nor ID were of specific interest for our purpose. In order to handle NaNs correctly in the identification of duplicates (NaN != NaN) they had to (temporarily) be transformed to numerical representations (-99).

After these transformations the dataframes were saved to disk individually in the space-saving parquet format and after that merged into a single dataframe, which was again memory-optimized by downcasting numerical columns. Also, all kinds of missing values were standardized to np.nan for numerical and pd.NA for categorical columns.

All numerical columns were transformed to float32 (can handle np.nan) except “ID” and “year” (Int64).

The initial size of the downloaded files for all years was over 16 GB. Through the above transformations it was reduced to ~ 300 MB on disk and ~1.6GB in memory. The shape of the resulting data set is 14.442.792 rows, 41 columns. This refers to the whole Dataset of 30 countries and the timespan of 2010 to 2023.

2.2 Renaming/Mapping of Categories:

Many of the categorical variables contained high numbers of unique categories, often due to misspellings or name variations . For example, in the "Country" variable, where the vehicle was registered, different terms for countries appear: GB, UK, United Kingdom. Additionally, some strings may have trailing spaces, while others do not. All of this was handled using a callable function that takes a column and a dictionary for mapping as input. In order to clean up the data and standardize the categories, this approach was applied to the following columns: Ct, Cr, Fm, Ft, Country, Mp, Mh, and IT.

2.2.1 Specific treatment of the IT variable

The "IT" variable contains codes for "Innovative Technologies" recognized and approved as decreasing CO2 emissions by the EU (like specific LED lights and/or 12 volt efficient alternators). The challenge here was that if a car had several of those technologies the codes were lumped together in the IT column in various different ways. Significant effort had to be put into parsing and extracting individual IT codes and identifying them correctly. In the end they were spread over five columns "IT_1" to "IT_5" to accommodate cars having up to 5 of these technologies built in.

After these steps we had a properly cleaned and formatted large dataset which contained a maximum of valid data to further choose from and feature engineer for our modeling process.

2.3. Preprocessing pipeline part two.

The second part of the preprocessing pipeline consists of choosing specific columns, rows and categories plus NAN handling, outlier handling, encoding of categorical variables and some feature engineering. We streamlined the process, tried a lot of variations of the dataset and settled on a rather minimal dataset, mainly due to computational considerations for the modeling process.

- **We chose "Ewltpl (g/km)" as the target variable and removed Enedc (obsolete testing standard).**
- **We dropped all years prior 2019**
- **We chose only the biggest three countries (DE, FR, IT)**
- **We dropped all electric and hybrid cars (Electric cars have 0 emissions. Hybrids would require separate treatment).**

2.3.1 Choosing attributes

2.3.1.1 Categorical Attributes:

As our goal was to predict CO₂ emissions from technical characteristics **we dropped all non-technical categorical attributes like Individual car identifiers, Brand and Model names etc..** The only not really technical characteristic we kept is “Mh”, which is one of the representations of the manufacturer. The reasoning behind that was that manufacturers probably differ in their respective engineering and production technologies so that “Mh” might serve as a proxy for many manufacturer-specific unobserved technical characteristics.

Categorical attributes kept:

“Mh”: Manufacturer name EU standard denomination

“Ct”: Category of Vehicle (Passenger cars vs. light goods and offroad)

“Ft”: Fuel type (Petrol, Diesel, LPG etc..)

“Fm”: Fuel mode (Mono-Fuel, Bi-Fuel etc..)

“IT” : Innovative Technologies

2.3.1.2 Numerical attributes:

We **excluded fuel consumption** from our analysis because it is primarily a surrogate for the target variable, CO₂ emissions and shows high collinearity. CO₂ emissions in the Wltp test procedure are even calculated by multiplying the measured fuel consumption with a fuel-specific factor. Including fuel consumption would therefore make it nearly impossible to accurately isolate the effects of other technical characteristics on emissions.

Additionally, we **removed “Erwltp (g/km)”** and **“Ernedc (g/km)”** from our feature set. These variables represent the total emissions reductions achieved through the application of “innovative technologies.” As such, they are outcome metrics rather than intrinsic technical characteristics of the vehicles. Including them would introduce redundancy, as they are measured in a manner similar to the target variable itself and do not provide independent technical insights.

Instead, the underlying technical factors contributing to emissions reductions are captured by the various “Innovative Technologies” represented in the “IT” attribute. This attribute directly reflects the engineering and technological innovations employed, offering a more precise and meaningful predictor for our CO₂ emissions model.

We also **removed “m (kg)”** as it didn't make sense to keep two measures of mass (highly collinear).

Measures of size:

Regarding measures of size we introduced **one new feature: “Area (m2)”**

The dataset includes three measures of length: the wheelbase (“W”) and axle widths (“At1” and “At2” in millimeters). We combined these measurements to create a composite feature representing the car's size:

$$\text{Area (m}^2\text{)} = W * (\text{At1} + \text{At2}) / 2$$

Introducing this area feature offers several advantages:

- Physical relevance: The area serves as an indicator of the car's overall footprint, which can influence aerodynamic properties and, consequently, CO₂ emissions.
- Mathematical relevance: By incorporating the area, we introduce a quadratic term into the model. This allows the model to capture non-linear relationships between the car's dimensions and its CO₂ emissions, potentially leading to more accurate predictions.
- Statistical interaction: The area is effectively an interaction term between the wheelbase and axle widths. This interaction can capture the combined effect of these dimensions on emissions, which might not be apparent when considering each measurement individually.

The inclusion of the area feature can potentially enhance the model's accuracy by providing a more comprehensive representation of the vehicle's size-related characteristics. We initially considered eliminating all length measurements in favor of using only the Area feature. However, results showed that models retaining one length measure alongside Area produced slightly more accurate predictions. Additionally, from a mathematical perspective, keeping at least one linear size-measure makes sense as well.

Numerical attributes kept:

“Mt”: Test mass in kg as measured for the Wltp test.

“ep (KW)”: Engine power in kW

“ec (cm3)”: Engine capacity in cm³

“At1 (mm)”: Axle width (steering axle) in mm

“Area (m2)”: Area in m²

2.3.2 Choosing categories (attribute values of the categorical columns)

Categories were selected automatically by specifying two parameters: `top_n` and `min_cat_percent`. The `top_n` parameter retained the 20 most frequent categories up to the specified number, while `min_cat_percent` ensured that each retained category represented at least 0.1% of the total dataset (mainly to prevent problems in modeling when categories could fall empty when dividing the dataset). Categories that did not meet these criteria were labeled as "Other".

2.3.3 Handling outliers and skewed attributes

Attributes with an approximate gaussian distribution had outliers replaced with their median. Those with non-gaussian distributions had the outliers capped and set to the boundaries of the outlier range. Outlier handling was performed twice: initially at the beginning of batch processing and subsequently at the end. This two-step approach was adopted to accommodate any distribution changes that occurred during the preprocessing steps.

Two highly skewed attributes, "ep (KW)" and "ec (cm3)", were transformed using the Box-Cox method, with the optimal lambda calculated via the maximum likelihood estimation. This made outlier removal on these attributes less aggressive.

We initially applied the standard IQR multiplier of 1.5 to define the outlier range. However, we ultimately selected a multiplier of 3.0 after determining that a less aggressive approach to outlier removal mitigated collinearity issues and enhanced prediction performance.

2.3.4 NaN Handling

Since we had enough data points for our models and were okay with having a smaller dataset, we chose not to try to salvage rows with imputing values for NaNs. Therefore, we dropped all rows with missing values except for the IT attribute, where missing values are expected because not every car includes innovative technologies.

2.3.5 Encoding

All categorical variables were one-hot encoded, with one category omitted for each to prevent the dummy variable trap. This approach avoids multicollinearity by ensuring that the encoded variables are independent of each other. The IT codes were one-hot encoded to indicate whether a car possesses each specific technology, (regardless of which of the five IT columns the code appeared in).

2.3.6 Drop duplicates

As a final step we removed all duplicate rows and recorded the number of occurrences in a “frequency” column to preserve the representation of similar vehicles (also taking to account the already aggregated values of “#identical occurrences”) Since many non-technical attributes that could uniquely identify individual cars were removed, the duplicates again do not represent true duplicates but rather multiple instances of similar cars. This approach maintains the true distribution of the dataset for use in weighted models.

The final dataset has 2,000,450 rows, 56 columns and is 282.4 MB in memory.

3. Modeling

Since our project involves predicting a continuous target variable, Ew/tp (g/km), this is inherently a regression problem. Our primary approach was to establish a robust baseline using **multilinear regression**, a widely accepted model for regression tasks¹. This choice allowed us to evaluate the model's performance under straightforward, interpretable assumptions about linear relationships between features and the target variable.

3.1 Choice of Algorithms

Following the initial baseline model, we tested several advanced algorithms with the aim of improving predictive performance and capturing complex, non-linear relationships. The algorithms we selected were:

1. **Decision Trees**
2. **Random Forests**
3. **XGBoost**
4. **Dense Neural Networks (DNN)**

The rationale behind the choice of these algorithms is based on their complementary strengths and suitability for regression tasks involving structured, tabular data:

Decision Trees were chosen for their interpretability and ease of handling non-linear relationships. Decision Trees can model intricate feature interactions without requiring scaling or extensive preprocessing, making them a natural next step from linear models when seeking to capture non-linearity in the data. Here techniques such as *pruning* were tested in order to deal with overfitting, which were suspected given the extremely good results such as:

Training MSE: 0.0061

Training R-squared: 0.99

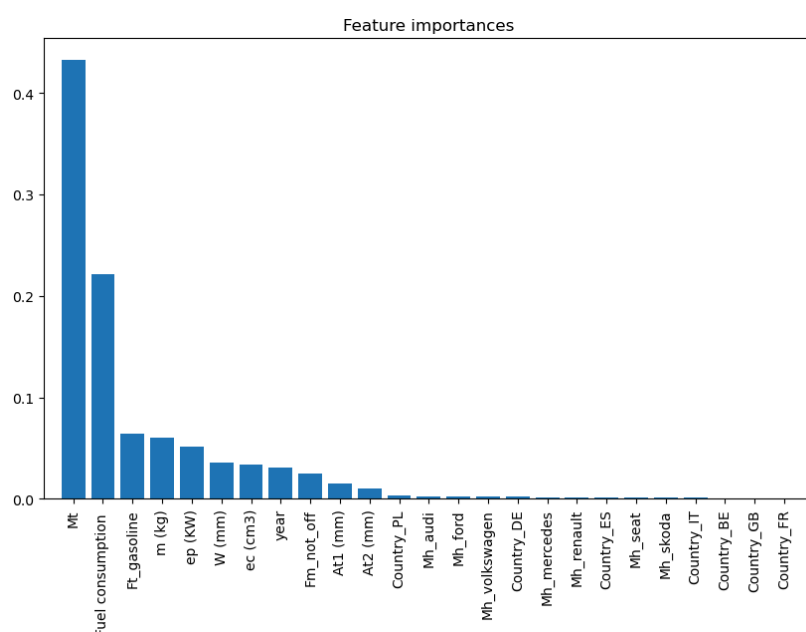
With the R-squared of 0.99 indicating that the Decision Tree model is explaining 99% of the variance in the training set and an extremely low MSE, which suggests that the model is performing almost perfectly on the training data, I assumed that the model was overfitting. Although at this moment, I did not make a crucial comparison

¹ The following paragraphs reference the notebooks saved at the Github repository: ..
\\notebooks\\Alex\\2- First modeling_EU_2019_2023

with the R-squared of the test set, this suspicion was enforced given the low R-squared obtained via cross validation: 0.64.

Random Forests, an ensemble method that aggregates multiple Decision Trees, were selected due to their robustness and reduced tendency to overfit compared to a single Decision Tree. The use of multiple trees enhances predictive accuracy and stability, as Random Forests average the results of various weak learners to form a stronger model. This model is also relatively well interpretable, allowing for feature importance analysis, which is valuable for understanding which predictors have the most significant impact on *Ewltp (g/km)* emissions.

See for example:



Here a Test R-squared: 0.98 was achieved², and via cross-validation a R-squared of 0.92 was obtained. In experimenting with this model, a major setback was the attempting of hyperparameter tuning via Grid Search. Only a limited range of hyperparameters could be tested given the extensive use of computational resources.

See for example:

² Note, however, at this moment the group was still discussing the best preprocessed dataset to use. For instance, these first models included the variable Fuel consumption, which was later removed for being basically the same as the target one - in physical terms.

```
# n_estimators: 5,10,20. max depth: none,
param_grid = {
    'n_estimators': [3, 5, 10],
    'max_depth': [None, 2, 4, 6],
    'min_samples_split': [2, 3, 5],
    'min_samples_leaf': [1, 2, 3]}
```

XGBoost³ was later included due to its advanced boosting mechanism (as found in the literature⁴), which improves performance by sequentially building trees and learning from residual errors. XGBoost has proven effective in a wide range of structured data applications due to its optimization techniques, handling of missing values, and ability to combat overfitting through parameters like regularization. Given its scalability and high performance on regression tasks, XGBoost represented a robust choice for achieving accuracy on complex datasets.

The model was run with and without hyperparameter tuning , however, the results did not differ, and both were satisfactory.

Metrics XG Boost	Metrics XG Boost with Grid Search
Training R-squared: 0.97	Best Model Test R-squared: 0.97
Test R-squared XG: 0.97	Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 150, 'subsample': 1.0}

XG Boost was also subjected to cross validation (CV=5), using the best parameters found in the step before, and the results were also satisfactory.

Cross-validated R-squared scores: [0.95309782 0.95203137 0.97476494..]

Mean R-squared across folds: 0.96

³ The following paragraphs reference the notebooks saved at the Github repository: .. \notebooks\Alex\DataPipeline3_only models\newset

⁴ See for example: Kharka, D (2023). Boosting Performance with Extreme Gradient Boosting. URL:

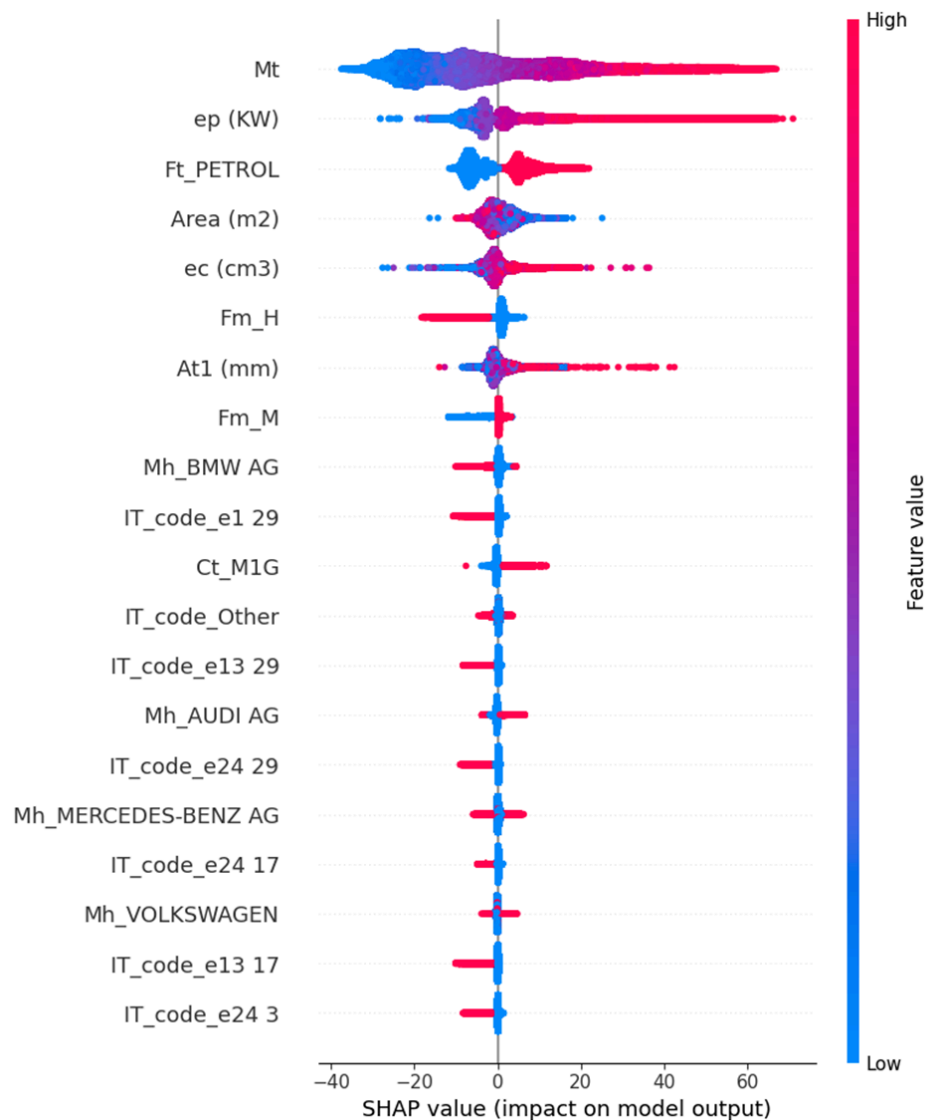
<https://medium.com/@dishantkharkar9/unravelling-the-power-of-xgboost-boosting-performance-with-extreme-gradient-boosting-302e1c00e555>. Se also Tarwidi, D, et al, (2023). An

optimized XGBoost-based machine learning method for predicting wave run-up on a sloping beach. URL:

<https://www.sciencedirect.com/science/article/pii/S2215016123001206#:~:text=XGBoost%20is%20now%20the%20most.and%20large%20and%20complex%20datasets>.

Standard deviation of R-squared across folds: 0.0092

Now, the models discussed so far are easy to interpret. See for instance SHAP values for XG Boost, which provides insight into the importance and impact of different features on the model's predictions:

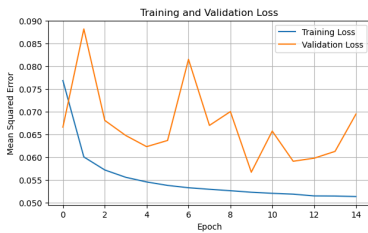


The plot highlights that vehicle specifications such as Ewltip mass ("Mt"), engine power ("ep (kW)"), engine capacity ("ec (cm³)"), and manufacturer can significantly affect the predictions. This level of interpretability helps to understand the driving factors in the model's decision-making process and emphasizes the importance of

both numerical and categorical variables in the context of this dataset. In contrast, the following model generally lacks interpretability⁵.

Dense Neural Networks (DNN) were lastly introduced as a deep learning approach to explore the possibility of capturing highly complex interactions among features that may not be adequately handled by tree-based algorithms. DNNs can theoretically approximate almost any function, making them highly flexible for modeling non-linear and intricate relationships. Moreover, with advancements in regularization techniques such as dropout and weight regularization, DNNs have become more feasible for regression tasks, where avoiding overfitting is crucial.

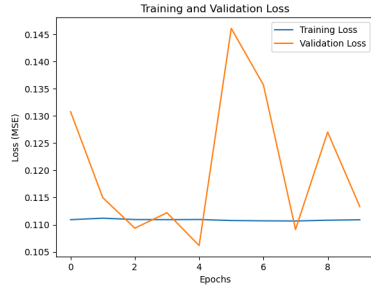
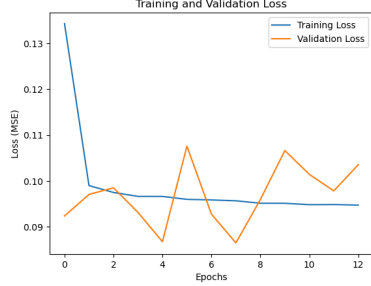
Various regularization techniques, architectural adjustments, and dropout rates were tested to improve model performance and mitigate overfitting. Despite these efforts, the simpler model ultimately demonstrated better generalizability, with reduced overfitting compared to more complex architectures. Across four trials, each model explained over 90% of the variability in the target variable, but the validation loss curve was more stable and consistent in the simpler unregularized initial model, indicating superior reliability and robustness⁶. See the following table for an overview⁷.

Trials	Key Metrics	Plot Training and Validation Loss
<p>Attempt 1:</p> <p>Simple architecture and small batch size. It includes an early stopping</p>	<p>Training MSE: 53.3 Test MSE DNN1: 53.7 Training R-squared: 0.9 Test R-squared DNN1: 0.9</p>	

⁵ I am aware that there are recent advancements in interpretability techniques, such as Layer-wise Relevance Propagation (LRP). However and besides the fact that there was not time left to deep dive into it, it seems that Despite these tools, DNNs remain more challenging to interpret in terms of understanding individual feature contributions compared to simpler models, which can be a limitation in applications requiring high transparency

⁶ At this stage, I am employing the final preprocessed dataset:
minimal_withoutfc_dupesdropped_frequencies_area_removedskew_outliers3_0_NoW_tn20_mcp00.10.parquet.
It can be found at: [Link DataSet](#)

⁷ The code for these trials can be found at the Github repository: .. \notebooks\Alex\10 - Modelpipeline _ tests DNN

<p>Attempt 2:</p> <p>A <code>kernel_regularizer=l2(0.001)</code> was added and the dropout was increased from 0.2 to 0.3</p>	<p>Training MSE: 81.6 Test MSE DNN2: 81.8 Training R-squared: 0.9 Test R-squared DNN2: 0.9</p>	
<p>Attempt 3:</p> <p>The changes before were kept, but the batch size was increased to 32</p>	<p>Training MSE: 80.1 Test MSE: 80.4 Training R-squared: 0.91 Test R-squared: 0.91</p>	
<p>Attempt 4:</p> <p>The changes before were kept, but the batch size was increased to 32</p>	<p>Training MSE: 67.0 Test MSE: 67.2 Training R-squared: 0.9 Test R-squared: 0.9</p>	

3.2 Exclusion of Decision Trees and Random Forests

While Decision Trees and Random Forests initially showed promising results in terms of performance metrics like R2, these models were ultimately excluded from the final analysis. The primary reasons for their exclusion are as follows:

1. **Redundancy with XGBoost:** Both Decision Trees and Random Forests are tree-based models, with Random Forests representing an ensemble of Decision Trees and XGBoost implementing a more sophisticated version of boosted trees. XGBoost's advanced algorithm surpasses the performance of Decision Trees and Random Forests by optimizing each iteration to correct for previous errors, resulting in a more precise and flexible model. Given that XGBoost incorporates tree-based strengths with enhanced regularization and

boosting mechanisms, the standalone Decision Tree and Random Forest models were deemed redundant.

2. **Performance and Computational Efficiency:** XGBoost generally performs better on complex, high-dimensional data due to its advanced handling of residual errors and overfitting. In our tests, XGBoost not only yielded higher accuracy but also demonstrated more stable performance across various data splits. Although Random Forests can also mitigate overfitting to an extent, their performance did not significantly surpass that of XGBoost, which rendered them less efficient given the computational resources required.
3. **Interpretability Concerns:** While Decision Trees are relatively interpretable, Random Forests are less so due to the averaging of multiple trees, which complicates direct interpretation. XGBoost, while complex, still allows for feature importance analysis and SHAP value interpretation, making it more versatile for interpretability in high-stakes scenarios. Consequently, the exclusion of Decision Trees and Random Forests helped to streamline the analysis without compromising interpretability or model performance.
4. **Scalability and Flexibility:** XGBoost is optimized for scalability with large datasets and offers greater control over hyperparameters, making it better suited for fine-tuning. Given our dataset's complexity, XGBoost's flexibility with hyperparameter tuning via grid search further improved our ability to optimize performance. DNNs, though not inherently interpretable, offer another pathway for exploring non-linear relationships, making Decision Trees and Random Forests less necessary in comparison.

3.3. Model comparison: Linear Regression, XG Boost and Dense Neural Network

As previously noted, model testing was conducted with progressively refined data preprocessing. For the final evaluation, each selected model was applied to the latest version of the preprocessed dataset, ensuring consistency in comparison. The table below provides an overview of the optimization and evaluation techniques used for each model, along with the interpretability methods applied to gain insights into model behavior.

Models/Techniques	Grid Search	Elastic Net	Cross Validation	Interpretability
Linear Regression	No	Yes (given persistent multicollinearity)	Yes (to evaluate generalizability)	Feature Importance
XG Boost	Yes (opt. parameters)	Not applicable	Yes (evaluate generalizability)	Shap values
Dense Neural Network)	No	Not applicable, but Ridge regularization was applied	No, but a validation set was used.	Not applied

Finally, all the optimized models were re-run, and their results were compared. The evaluation metrics used for comparison included Mean Squared Error (MSE), Test R-squared, Cross-validated R-squared, and Training Time. Since the DNN-model was trained with the target variable also scaled this was adopted for the two other models to make the scores better comparable. The following table summarizes the results of this comparison:

Model	Test MSE	Test R-squared	Cross-validated R-squared	Training Time (minutes)
Linear Regression	0.126959	0.873043	0.873155	7.07
XG Boost	0.026375 (*)	0.973626	0.973652	1.41
Dense Neural Network	0.061685	0.938316	N/A	0.26

(*) would be equivalent to an RMSE back-transformed-value of ~ 5 g/km CO2 Emissions in the units of the target variable: Ew/tp.

Based on this table we can conclude the following:

1. Performance on Test Set (Test MSE and Test R-squared):
 - XGBoost demonstrates the best performance among the models with the lowest Test MSE (0.026375) and the highest Test R-squared (0.973626). This indicates that XGBoost has the best predictive accuracy on unseen data, capturing around 97.4% of the variance in the target variable, which suggests a highly effective fit.
 - The Dense Neural Network (DNN) follows closely, with a Test MSE of 0.061685 and a Test R-squared of 0.938316. While DNN performs well, its error is somewhat higher than XGBoost.
 - Linear Regression has the highest Test MSE (0.126959) and the lowest Test R-squared (0.873043) of the three models. While it serves as a good baseline model, it lacks the flexibility and predictive power of the more complex models like XGBoost and DNN, particularly for potentially non-linear relationships in the data.
2. Cross-validated R-squared:
 - Both XGBoost and Linear Regression have similar cross-validated R-squared values, with XGBoost outperforming at 0.973652 compared to Linear Regression's 0.873155. This consistency across folds highlights the robustness of XGBoost, confirming that it maintains high predictive accuracy even across different subsets of the data.
 - Dense Neural Network (DNN) does not have a cross-validated R-squared value available, possibly due to constraints with cross-validation on neural networks or time limitations.
3. Training Time:
 - Dense Neural Network (DNN) shows a significantly shorter training time (0.26 minutes), which is advantageous if rapid training and iterative tuning are needed. Despite this speed, it achieves relatively high accuracy, making it a good option when computational efficiency is a priority.

- XGBoost takes 1.41 minutes, balancing training time and high accuracy. Although it is slower than DNN, its more accurate and stable predictions justify the slight increase in computation time⁸.
- Linear Regression takes the longest time (7.07 minutes). This is likely due to the extensive cross-validation required in the ElasticNetCV implementation. While it is the simplest model and interpretable, the high training time coupled with lower predictive performance may make it less ideal for complex data.

Summary

XGBoost emerges as the strongest model in terms of predictive accuracy and robustness, with the lowest MSE, highest R-squared, and a reasonable training time. Its decision tree-based structure allows it to effectively capture non-linear relationships, which linear regression cannot. XGBoost's balance between performance and efficiency makes it a highly suitable choice.

Dense Neural Network is a close runner-up, with a fast training time and high R-squared. However, as mentioned before, it may require more tuning to prevent overfitting or instability across different data subsets.

Although Linear Regression provides interpretability and simplicity, its lower performance metrics suggest that it may not be sufficient for this dataset. It serves well as a benchmark but falls short compared to the flexibility and accuracy of the more advanced models.

In conclusion, XGBoost is recommended as the primary model due to its optimal balance of accuracy, stability, and efficiency.

⁸ This metric should be interpreted with caution for XGBoost, as the model has already undergone optimization steps. However, if we were to consider the complete grid search process for XGBoost, which took between 120 and 190 minutes in various runs, XGBoost would emerge as the most computationally intensive algorithm tested.

Conclusion

a) Achievement of the project goals

In general terms, shifting to electric powered vehicles emitting by definition 0 gr/km CO₂ is the way to go and which has been adopted by almost all car manufacturers. Since the required infrastructure for charging these electric powered vehicles differs from country to country the transition will take some time. It is here where the general data-based recommendations for fossil fuel powered vehicles could be found. The developed ML-models make a good prediction of the CO₂ emissions based on almost only technical features. Feeding the model with new data will allow CO₂ emission predictions.

For the case of CO₂ emissions minimization the following can be said:

Reducing the following parameters will also reduce the CO₂ emissions: Weight, size, engine capacity (ccm), engine power. Adopting single or better combined innovative technologies like exterior LED lights and/or 12 volt efficient alternators will also reduce the CO₂ emissions. As for a fossil fuel type diesel engines have a higher efficiency compared to petrol engines but at the cost of other emissions besides CO₂ like micro particles. As for what the user can do is reduce the amount of travels and reduce the speed, as the speed correlates with the CO₂ emission (ADEME dataset, where emissions at different speeds are also recorded).

Concerning the exploration of how innovative technologies for CO₂ reduction might be applied across different industries, no newer technologies were found that could be transferred, but it can be assumed that further innovations will arrive and are now in the approval process.

As for the scientific contributions concerning finding correlations between vehicle features and CO₂ emissions and so accurately predicting emissions based on technical characteristics it can be said that the major impacting or correlating features (mostly technical) were found. Further details can be found under the following chapter: Contribution to scientific knowledge (detailed).

b) Continuation of the project

Avenues for improvement

Feature Engineering: Further refining the feature set might enhance model performance. For instance, creating more interaction terms between features (e.g., weight and engine capacity for CO₂ emissions) could capture more complex relationships. Additionally, analyzing non-linear relationships could reveal transformations that make the features more relevant for predicting emissions.

Ensemble Methods: While XGBoost was used as a high-performing ensemble model, further ensemble approaches could be tested. For instance, combining predictions from Linear Regression, XGBoost, and Dense Neural Networks (DNN) in a stacked ensemble could leverage the strengths of each model and potentially improve overall accuracy.

Hyperparameter Tuning: Although grid search was used to optimize some models, more exhaustive tuning or Bayesian Optimization⁹ could find optimal parameters in a more computationally efficient manner, particularly for complex models like XGBoost and DNN.

Deep Learning Architectures: In the case of the Dense Neural Network, trying architectures such as Convolutional Neural Networks (CNNs) for structured data, or employing Residual Connections or Layer Normalization might reduce overfitting while capturing more nuanced relationships in the data.

Cross-validation with Time Series Splits: Since the dataset spans multiple years, testing a time-based cross-validation approach could help ensure that the model generalizes well across different time periods, making predictions more robust.

Data Augmentation: If some categories, such as certain car types or fuel types, are underrepresented, data augmentation techniques or synthetic data generation could balance the dataset. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) might create more balanced training samples.

Utilizing model weights: The “frequency” column can serve as a valuable weight parameter in various models, including the ones we explored in this project. While recalculating all models with weights exceeded the scope of this project, we recommend leveraging weights in future work to more accurately reflect the original

⁹ We did not have this approach in our courses, however I understand that it builds a probabilistic model of the objective function and uses this model to select the most promising hyperparameters to evaluate next.

dataset distribution. This approach can be beneficial because each duplicate entry represents a unique occurrence of a similar vehicle on the road, contributing to the overall prevalence of specific car types. Without weights, models may simplify the data distribution, which could lead to an under- or overemphasis on certain vehicle types in the reduced dataset.

Testing model accuracy metrics on a reduced, unweighted dataset provides useful insights; however, weights can enhance the alignment of model predictions with real-world scenarios by ensuring that more common vehicles exert a proportional influence on the model's learning process. Applying weights in future analyses could provide additional accuracy by preserving the natural frequency of different car types, which may improve the model's ability to generalize and make predictions that are closely aligned with real-world distributions.

Contribution to scientific knowledge (detailed)

This project has contributed to the scientific understanding of CO₂ emissions in the following ways:

- **Data-Driven Insights on Emission Factors:** By analyzing a rich dataset that includes technical and categorical features, this project provides empirical insights into how various factors—such as vehicle weight, fuel type—affect CO₂ emissions. The findings could be valuable for policymakers and car manufacturers aiming to reduce vehicle emissions across Europe.
- **Model Comparisons for CO₂ Prediction:** The project contributes to the discussion on which machine learning models are most effective for predicting emissions. By rigorously comparing Linear Regression, XGBoost, and DNN, this study provides a reference for the performance and computational trade-offs of each approach, guiding future research on emissions modeling.
- **Identification of Key Predictors:** The project identified and quantified the importance of different features in predicting emissions through feature importance analysis. These insights help highlight which factors are the most influential in vehicle emissions, potentially guiding future regulations or engineering decisions.
- **Contributions to Model Interpretability in Complex Systems:** The use of SHAP (SHapley Additive exPlanations) values and other interpretability techniques provides a framework for explaining complex models like XGBoost, in a way that is accessible to non-technical stakeholders. This work demonstrates how black-box models can be made interpretable in emissions research, bridging the gap between accuracy and interpretability.
- **Methodological Framework for Future Studies:** The project establishes a methodological framework for analyzing emissions data using machine learning, which can be replicated or expanded upon in future studies. By demonstrating a complete workflow—data preprocessing, model training, evaluation, and interpretation—this work can serve as a blueprint for other researchers studying similar environmental data.

These insights and methodologies not only enhance the understanding of CO₂ emissions from vehicles but also provide a foundation for future research aiming to use machine learning to address pressing environmental challenges.

Appendix

Difficulties encountered during the project

Decision for a Data Source

One of the challenges in our project was choosing which data to work with. We were fortunate to have a wide range of data points available, which required us to carefully determine which of these would be most valuable for our goal. This required a deep dive into the technical details of the problem and significant learning along the way. It's worth noting that this is a fictional project; in a real scenario, a consultant from the hypothetical car company would ideally be available to provide technical expertise and advisory support.

Three different data sources were considered: "EU data", "French Gov (2001-2015)", "French_Gov_ADEME" (see Report 1). Each source required evaluation and time investment to make an informed decision. Key considerations included identifying the target variable and assessing the explanatory variables available within each dataset. A challenge we faced was that these variables differed substantially between sources. Ultimately, a single source was selected, based on the following considerations.

Decision on a Target Variable and Year Cutoff

The goal was to develop a model that could predict a car's CO2 emissions based on its technical characteristics. However, there are several different methods for measuring CO2 emissions from vehicles, meaning there are also various potential target variables. These measurement methods have evolved both technically and within EU regulations over the years. Given these changes, as well as the impact of the diesel emissions scandal, we decided to use the most recent and EU-wide standardized emissions measurement, based on the Worldwide Harmonized Light Vehicles Test Procedure (WLTP). Since this measurement only appears in the datasets from 2018 onward, all earlier data was excluded, allowing us to focus on a single dataset. This also excludes any other variables intended to represent or approximate CO2 emissions.

Additionally, it should be noted that the expertise required to understand the variables grows exponentially, and the possibilities for engaging with the subject matter in depth are vast. Besides the understanding of the variables the data-preprocessing (cleaning) also grows significantly since the EU dataset is a collection of the EU countries who source the data from year to year. A logical and economically sound decision had to be made regarding how extensively one can and should delve into these details.

Decision on a Default Data Pipeline and Dataset

The team is made up of individuals with diverse perspectives, and each member worked on different parts of the project, leading to varying requirements for the dataset. Model developers, for instance, need certain specific features in the dataset, while data scientists focused on data processing aim to retain as much information as possible throughout the pipeline. Deciding which columns to keep must follow a certain technical and statistical logic; however, the necessary expertise is lacking. In a real-life project, this knowledge would typically be provided by a project partner or client.

In our project, there was no managerial role to lead calls or make key decisions, which significantly slowed down or sometimes completely stalled decision-making. This cost the team valuable time that was urgently needed for skill development, as the overall workload is very high. *This serves as clear feedback to the organizers of the training program.*

Identical Cars

Aside from removing completely identical rows as duplicates from the dataset, we also considered how to handle cars that were listed multiple times. This particularly involved examining the columns VHF, TAN, and TVV with the goal of identifying distinct cars (for example, a 2014 Fiat Punto with the exact same specifications should be removed). Several problems arose in this process:

- There are different identifier numbers.
- Some identifier numbers are erroneous (e.g., those containing a "?" within the TANs).
- There are cars with identical identifiers that exhibit significant differences in other attributes.

This is partly due to erroneous entries in the data, where some values are measured, leading to further variations. We sought a logical approach to address these issues, but ultimately, these efforts were deemed unsuccessful because they would have excessively minimized the dataset, removing many erroneous rows entirely. Additionally, there was the question of how to handle cars that share the same identifiers but have considerable differences in their technical specifications. Ultimately, it was decided to retain these frequently occurring cars in the dataset, as it makes economic sense to incorporate a weighting factor for commonly sold cars into the prediction model.

Dataset Size

With the final data source being the EU dataset, the dataset is too large to be easily processed on personal computers. Uploading it into memory takes a long time, and for some team members, this was simply not possible and alternatives like working on google-colab were partly adopted. While this approach made it possible to load the datasets and train the models the integration with Git/GitHub within google-colab could not be reached due to merge-conflicts. To work around this, we applied clever techniques to reduce the effective data size and make it manageable. This raises questions about how such issues would be handled in a real corporate environment—whether a high-performance server would be provided, or if a Colab Pro account might be an option. Excluding team members from the process due to hardware limitations is far from ideal.

Model Calculation Time

Especially with the more complex models, such as XGBoost and DNN, compiling with the final dataset sometimes took over two hours. Even with purchased compute time on Google Colab, performance improvement was minimal. Generally, it should be noted that the team lacks sufficient expertise in this area, so much of the work is still done in a trial-and-error mode. With such long computation times, this approach becomes impractical.

Description of Code files & structure in the repository

The main files:

The final standalone code consists of three files. One notebook focuses exclusively on data processing, as described in detail in the section above. This notebook, located in the repository folder under [notebooks/Datapipeline.ipynb](#), loads the raw file from the source, processes it through a customizable batch processing function, and saves the default dataset for further processing in the modeling data file. This includes the fitting/creation of the three best/major models and describes how they are created and optimized, which can be found in [notebooks/Modelpipeline.ipynb](#). Lastly, a [utils_co2.py](#) file was created, containing all callable functions used in the two notebooks. The data processing functions, in particular, are noteworthy because they are powerful yet large. To keep the data pipeline notebook organized and not overloaded, these were saved separately in the project repository's folder under [src/utils_co2.py](#).

Models

The finalized, compiled, trained, and optimized three models were saved using `joblib` or `keras.model.save()`. They are located in the repository folder under: `models:` `regression_model.joblib`, `xgb_model.joblib` and `dnn_model.joblib`. The corresponding

j-notebook: Modelpipeline.ipynb is located in the repository folder under notebooks. This notebook contains: Training and scoring of: the linear-regression-model as Elastic-Net, a grid-search for the XGBoost-model with a SHAPE analysis of the best parameters found, the training and scoring of the DNN-model and a comparison of all three models having the features and target variable scaled.

Preliminary tasks & supplementary tasks:

In the preliminary work for the project, various tasks were also conducted using alternative data sources. An initial overview of the data structure was created, different treatments were tested, and multiple test models were developed until the three final models were ultimately selected. All of this work was saved in separate notebooks, located in the folders [notebooks/ALL](#) or [notebooks/NameOfTeammember](#). These notebooks are not optimally organized or fully commented, and they are not submitted as official notebook files here, but they are kept in the repository to preserve the prior work done.

Bibliography

EU (2007). Directive 2007/46/EC of the European Parliament and of the Council of 5 September 2007 establishing a framework for the approval of motor vehicles and their trailers, and of systems, components, and separate technical units intended for such vehicles. Retrieved from [EUR-Lex](#).

EU (2007). Regulation (EC) No 715/2007 of the European Parliament and of the Council of 20 June 2007 on type approval of motor vehicles with respect to emissions

from light passenger and commercial vehicles (Euro 5 and Euro 6) and on access to vehicle repair and maintenance information. Retrieved from [EUR-Lex](#).

EU (2017). Commission Implementing Regulation (EU) 2017/1153 of 2 June 2017 setting out a methodology for determining the correlation parameters necessary for reflecting the change in the regulatory test procedure and amending Regulation (EU) No 1014/2010. Retrieved from [EUR-Lex](#).

EU (2019). Regulation (EU) 2019/631 of the European Parliament and of the Council of 17 April 2019 setting CO₂ emission performance standards for new passenger cars and for new light commercial vehicles, and repealing Regulations (EC) No 443/2009 and (EU) No 510/2011. Retrieved from [EUR-Lex](#).

EU (2019). Guidelines on the monitoring and reporting of CO₂ emissions from light-duty vehicles. Version 11 – December 2019. Retrieved from [CIRCABC](#).

Kharka, D. (2023). Boosting Performance with Extreme Gradient Boosting. Retrieved from [Medium](#).

Tarwidi, D., et al. (2023). An optimized XGBoost-based machine learning method for predicting wave run-up on a sloping beach. Retrieved from [ScienceDirect](#).

Innovative Technologies:

EU (2022) List of eco-innovations approved under WLTP Retrieved from [CIRCABC](#)

EU (2021) List of eco-innovations approved under NEDC Retrieved from [CIRCABC](#)

ICCT (2018) Overview and evaluation of eco-innovations in European passenger car CO₂ standards Retrieved from [ICCT](#)

EU (2022) Eco-innovative technologies reducing CO₂ emissions of light-duty vehicles: evaluation of interactions Retrieved from [POffice-EU](#)