Stages of the project

1- Classification of the problem

- What kind of machine learning problem is your project like? (classification, regression, clustering, etc)
- What task does your project relate to? (fraud detection, facial recognition, sentiment analysis, etc)?
- What is the main performance metric used to compare your models? Why this one?
- Did you use other qualitative or quantitative performance metrics? If yes, detail it.

2- Model choice and optimization

- What algorithms have you tried?
- Describe which one(s) you selected and why?
- Did you use parameter optimization techniques such as Grid Search and Cross Validation?
- Have you tested advanced models? Bagging, Boosting, Deep Learning… Why?

3- Interpretation of results

- Have you analyzed the errors in your model?
- Did this contribute to his improvement? If yes, describe.
- Have you used interpretability techniques such as SHAP, LIME, Skater… (Grad-CAM for Deep Learning…)
- What has (or not) generated a significant improvement in your performance?

Assessment methods:
Professional scenario: based on a proposed solution, the candidate will have to produce a summary report including: the explanation of the choices of AI solutions implemented, the interpretation of the results, the evaluation of the reliability of the algorithms and an optimization proposal.

1- Classification of the problem

Our project is a **regression problem**. This is because the target variable, **Ewltp (g/km)**, is a continuous variable representing CO_2 emissions. In a regression problem, the goal is to predict continuous values rather than discrete classes (as in classification) or unlabeled groups (as in clustering). Since we are predicting specific numeric outcomes (emission values), regression is the appropriate approach.

Furthermore, the task of our project is environmental modeling and forecasting CO_2 emissions (as measured by the Worldwide Harmonised Light-Duty Vehicles Test Procedure). This involves understanding how various vehicle technical characteristics and external factors (like the country of sale or registration) contribute to emissions levels. This task is often critical for regulatory compliance and environmental impact assessments, as understanding these emissions is essential for aligning with emission standards and achieving sustainability goals.

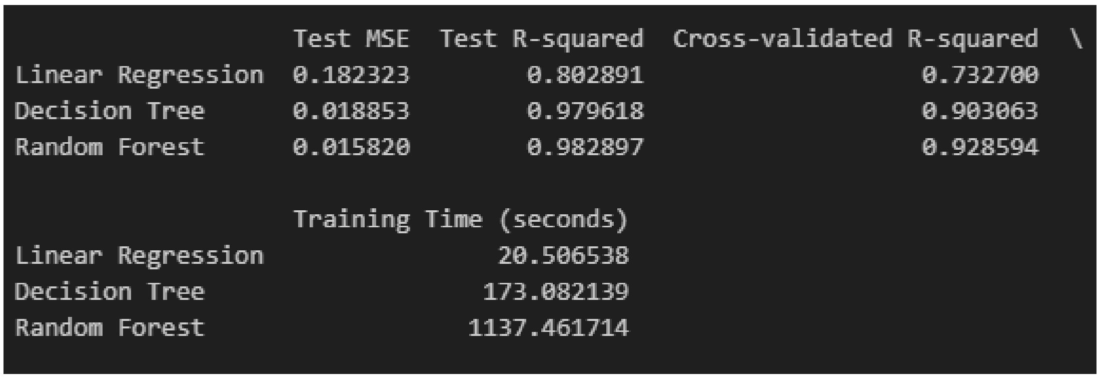
Additionally, our project could support tasks related to the engineering and design of new vehicles, enabling stakeholders to simulate and project the impact of their designs with respect to emission goals across Europe.

The primary performance metrics, which have been using for our regression models have been:

- **R-squared (R^2)**: This metric explains the proportion of variance in the target variable (Ewltp) that the model captures. An R^2 value close to 1 indicates that the model is capturing most of the variability in emissions, which is a key goal when trying to explain emissions as a function of vehicle features.
- **Mean Squared Error (MSE)**: These metrics quantify the average squared difference between predicted and actual emission values. Lower values indicate a model that more closely predicts the target variable.

The use of these metrics, especially, **R^2** provides a very interpretable metric for how well the model captures variance in emissions, making it easier to communicate model effectiveness to non-technical stakeholders.

An additional quantitative metric for evaluating the model was used, namely the training time (in seconds). Although computation time can be considered more of a practical factor rather than a core performance metric, it can still pose a real hindrance depending on the work environment the data scientist is operating in.



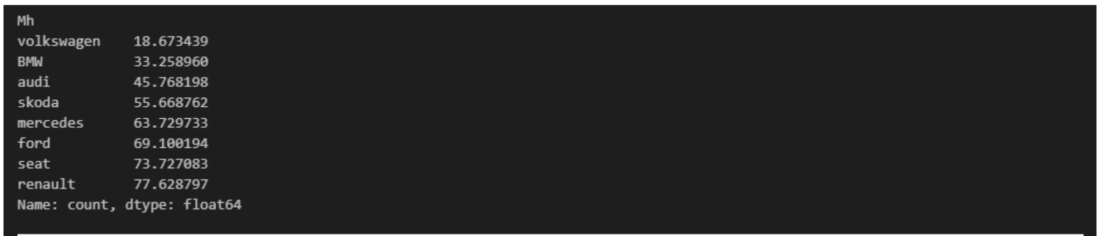
```
                    Test MSE  Test R-squared  Cross-validated R-squared  \
Linear Regression   0.182323        0.802891                   0.732700
Decision Tree       0.018853        0.979618                   0.903063
Random Forest       0.015820        0.982897                   0.928594

                    Training Time (seconds)
Linear Regression                 20.506538
Decision Tree                    173.082139
Random Forest                   1137.461714
```

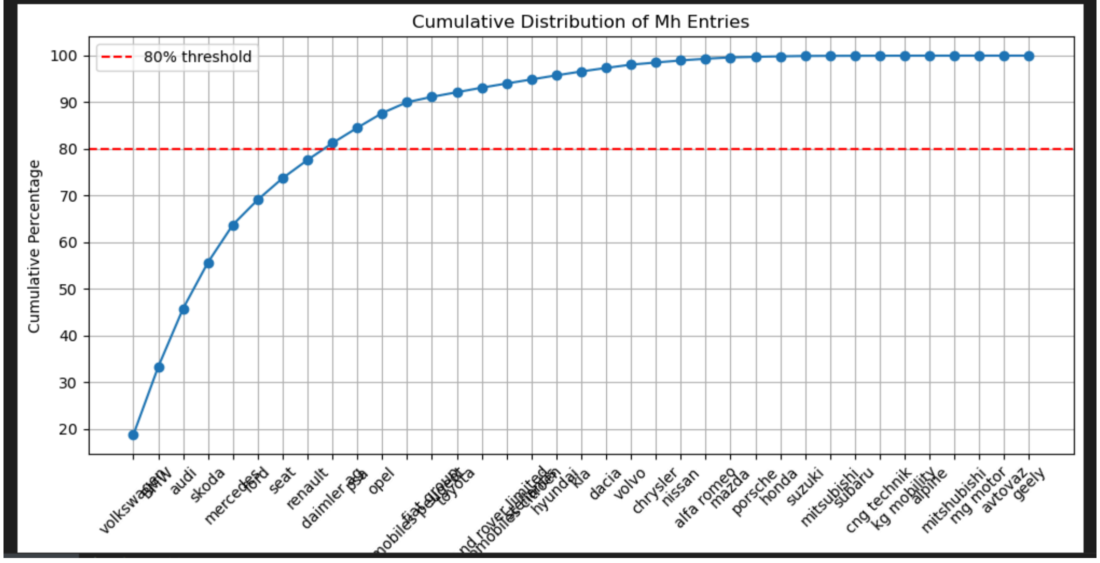
Here the most time-effective model is linear regression (20 sec) and random forest is the most time-consuming (around 19 mins).

Other quantitative metrics have been also used, albeit not for comparing the performance between models, but rather to assess and troubleshoot specific challenges in the modeling process. For instance, by setting a cumulative distribution threshold (e.g., 80%) for entries, we aimed to minimize the number of extra rows generated when applying the HotEncoder to categorical variables."

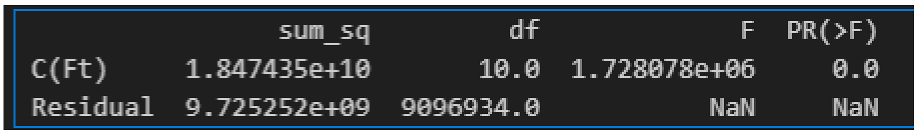
One example is the number of car manufacturers (Mh):



```
Mh
volkswagen    18.673439
BMW           33.258960
audi          45.768198
skoda         55.668762
mercedes      63.729733
ford          69.100194
seat          73.727083
renault       77.628797
Name: count, dtype: float64
```



Another example of metrics used is the F-statistic produced by an Analysis of Variance (ANOVA), which tests the statistical significance of the influence of the categorical variable **Ft** (fuel type) on the target variable (**Ewltp (g/km)**).



```
                sum_sq         df             F  PR(>F)
C(Ft)     1.847435e+10       10.0  1.728078e+06     0.0
Residual  9.725252e+09  9096934.0           NaN     NaN
```

2- Model choice and optimization

Models implemented so far have been: Linear Regression, Decision Tree Regressor, Random Forest Regressor, XGBoost Regressor (Extreme Gradient Boosting) and Deep Learning (DNN, Regressor and Classifier).

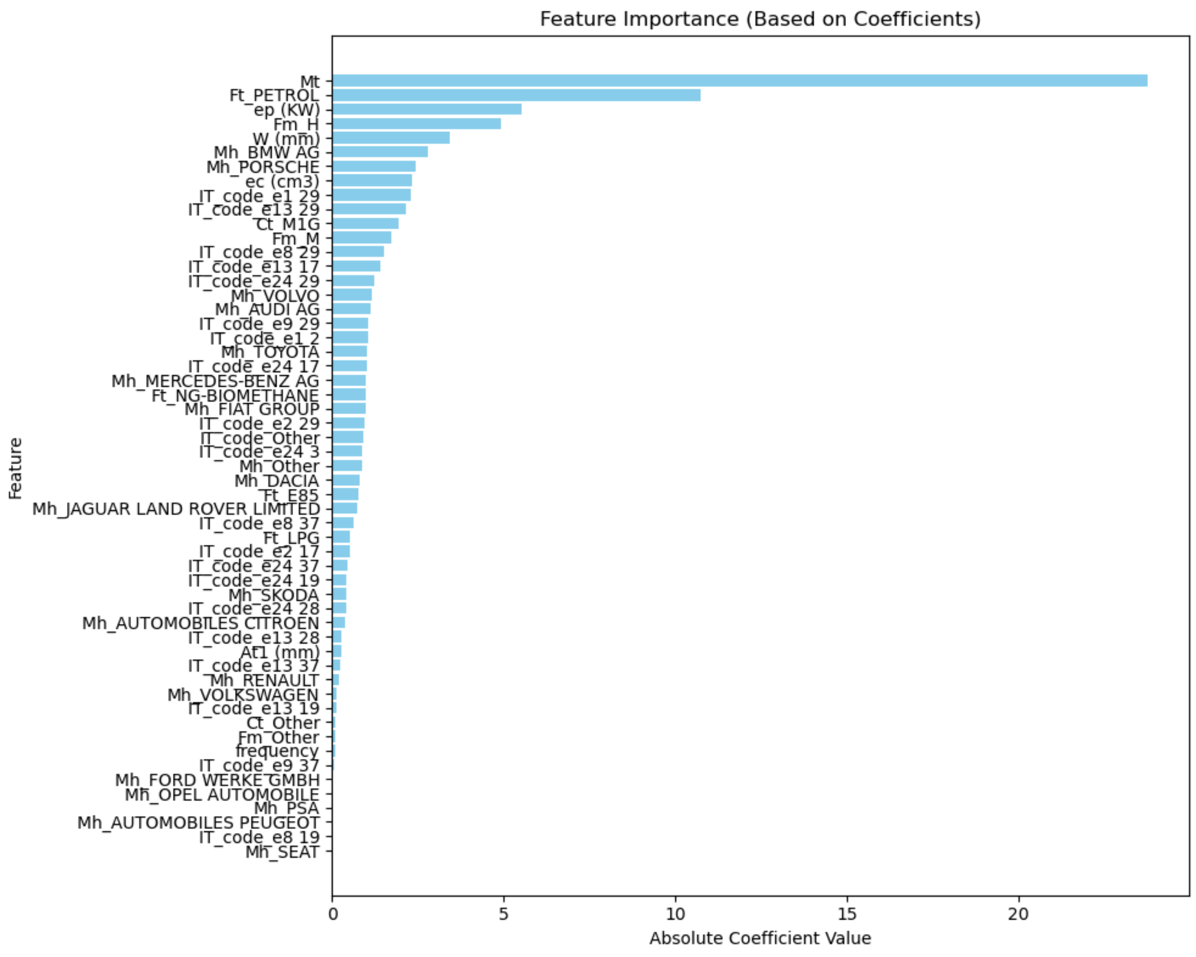
This selection is based on the nature of the data and the types of insights we aim to gain. Here's a rationale of why each of these models is suitable and beneficial:

Linear Regression (LR):

- Interpretability: LR offers high interpretability, making it easy to see the impact of each feature on the target variable (Ewltp). In a regulatory setting, where understanding the effect of vehicle characteristics on emissions is crucial, this can be highly useful.
- Baseline Model: LR serves as a good baseline model for regression problems, helping to assess whether more complex models provide substantial improvement.
- Low Complexity: LR has low computational complexity, so it's quick to fit and evaluate, even with large datasets. This allows for rapid initial analysis.

Limitations: Linear regression assumes a linear relationship between features and the target variable. This might not capture complex patterns in CO_2 emissions, especially if interactions and non-linear effects exist in the data.

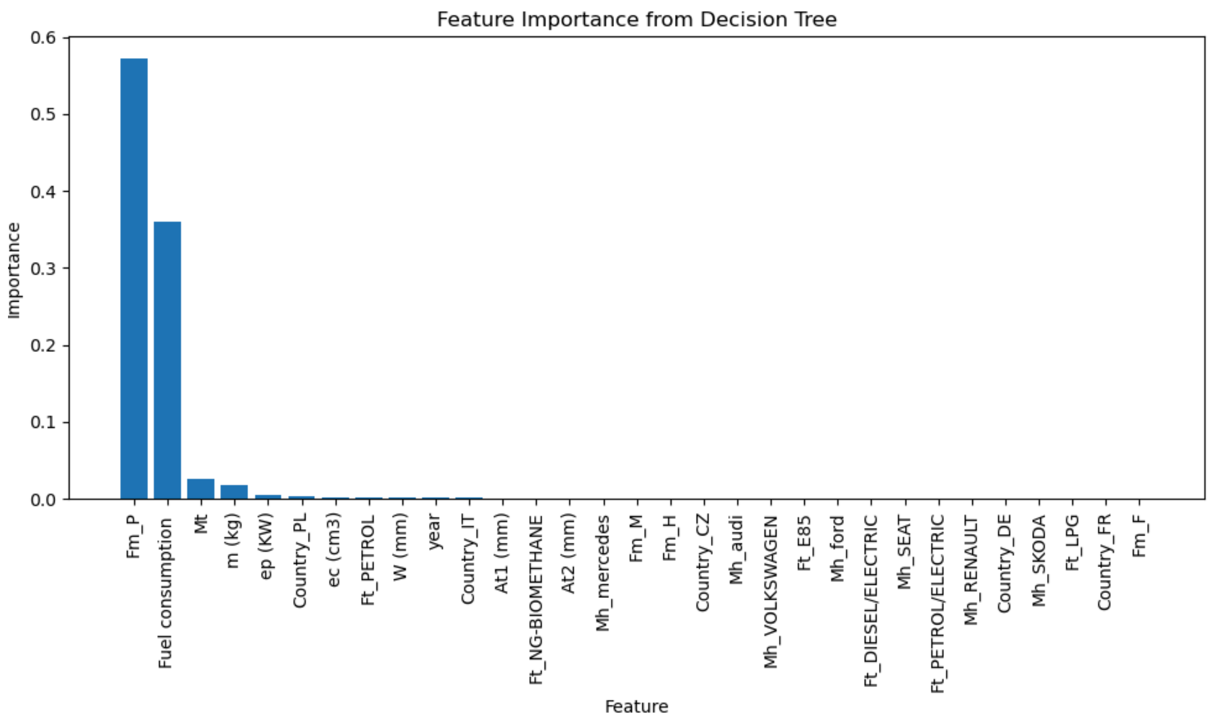
Here is an example of the good interpretability of LR. The absolute values of the linear coefficients of the attributes can be directly interpreted as feature importance:



Decision Tree Regressor (DT):

- Interpretability: DTs are known for being highly interpretable, providing a visual representation of the decisions that lead to different CO_2 emissions levels. This transparency is valuable in explaining why certain vehicle configurations might lead to higher or lower emissions.

Here is an example of Feature Importance for a DT:



- Non-linear Relationships: DTs can capture non-linear relationships between features and the target variable. This can be valuable if emissions are influenced by complex, non-linear interactions among vehicle characteristics, like fuel type, and country.
- Handling of Mixed Data Types: DTs naturally handle both numerical and categorical variables without requiring extensive preprocessing, making them versatile for datasets like ours that contain mixed data types.

Limitations: DTs are prone to overfitting, especially with deep trees, which can result in high variance. This can be partially addressed by tuning hyperparameters or using ensemble methods, like Random Forests.

3. **Random Forests (RF):**

- Interpretability: RFs are less interpretable than the simple models listed above. However, they provide feature importance scores, which can help to identify which vehicle characteristics most strongly influence emissions. This can be useful for feature selection or justifying decisions based on key factors.
- Reduced Overfitting: RFs - as an ensemble of multiple decision trees - reduce the risk of overfitting by averaging (or voting) the predictions from numerous trees. This generally leads to more robust predictions and higher accuracy on unseen data.
- High Performance for Tabular Data: RFs perform well on tabular data, especially in problems with non-linear relationships, making them suitable for datasets like ours with structured vehicle data.

Limitations: As mentioned before, RFs can be computationally intensive, especially with a large number of trees, and may not perform well with high-dimensional data where many features are irrelevant.

4. **XGBoost Regressor (Extreme Gradient Boosting) - (XG):**

- Interpretability: XGs like RFs are less interpretable than the simple models LR and DT. However, certain interpretability techniques can be used, e.g.. SHAP values.
- Feature Interactions: Gradient boosting methods like XG can capture complex feature interactions and non-linear relationships, making them ideal for datasets with intricate patterns, such as emissions influenced by multiple interacting vehicle attributes.
- High Predictive Power: According to the literature consulted, XG is known for its strong performance in predictive accuracy[1].
- Handling of Missing Values and Robustness: XGBoost has built-in mechanisms to handle missing values and is robust to outliers, making it well-suited for real-world data where some features might have missing entries.

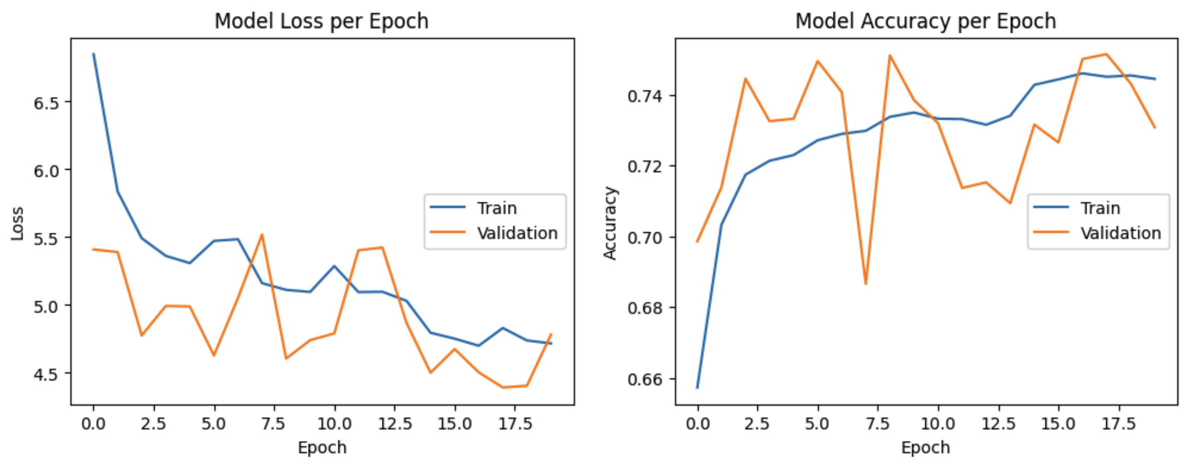
[1] See for example: Kharka, D (2023). Boosting Performance with Extreme Gradient Boosting. URL: https://medium.com/@dishantkharkar9/unravelling-the-power-of-xgboost-boosting-performance-with-extreme-gradient-boosting-302e1c00e555. Se also Tarwidi, D, et al, (2023). An optimized XGBoost-based machine 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.

Limitations: As previously mentioned, XGBoost is more challenging to interpret than simpler models. Additionally, it can be computationally expensive.

5. Deep learning (DNN).

To benchmark model performance against state-of-the-art machine learning models, we explored deep learning methods, particularly a deep neural network (DNN) classifier. DNNs have demonstrated remarkable power and versatility in recent years, making them appealing in both technical and commercial contexts. Thus, showcasing these models can also serve as a valuable selling point for clients. We developed two types of models: a DNN classifier and a DNN regression model.

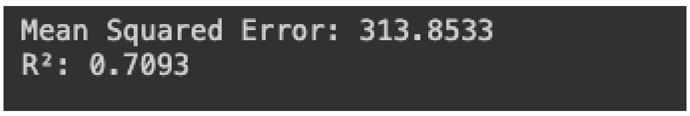
Classifier Model: To reduce computation time, the target variable was discretized into 20 bins, though this level of granularity is insufficient for an accurate representation of the problem. Achieving higher precision would require either a significantly larger number of bins or the direct use of the regression DNN model. The classifier model architecture consists of five layers: an input layer with 10 units, three hidden layers with 64, 256, and 128 units, and an output layer with 20 units (representing the number of classes). It was initialized with a Batch Normalisation layer, batch size of 30, 1 dropout layer (20% dropout), and 20 training epochs. Training this model on the default dataset of 3.5 million rows in Google Colab with a T4 GPU kernel took 48 minutes, making it considerably slower and less economical than other machine learning models tested.



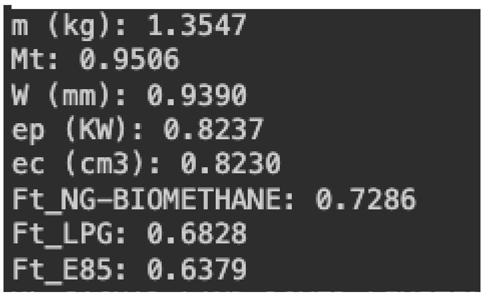
Additionally, the model’s performance lagged behind other methods, with accuracy reaching only about 75% after 20 epochs and improving more gradually afterward. To bring the DNN classifier to a competitive performance level, several enhancements could be considered: increasing the number and specificity of target variable bins, training for a larger number of epochs, reducing dropout layers (while managing overfitting), and applying cross-validation. However, these adjustments would be highly resource-intensive, especially for DNN models with complex architectures. The DNN regression model, with its finer granularity in the target variable, might be a more suitable candidate for such optimization. Further details on this model are discussed in the following section.

Regression model:

The Deep Neural Network regression model implemented consists of an input layer with 16 units, a hidden layer with 16 units, and an output layer with 1 unit. Additionally, a batch normalization layer and a dropout layer with a 20% rate were included to avoid overfitting. The model ran for 20 epochs with a computation time of 23 minutes on Google Colab. The results are weak compared to the other models presented here. However, it should be noted that the model had limited outlier handling (particularly not for the target variable), leading to noticeably poorer results than the other models listed. Nevertheless, it was decided to include it in the next optimization step, as random trials with DNNs showed that they deliver significantly better results with an optimized dataset. Feature importance here indicates the greatest influence from mass (m (kg)) and Mt; these two are collinear, so one of the two variables should be omitted or treated separately for the final model.



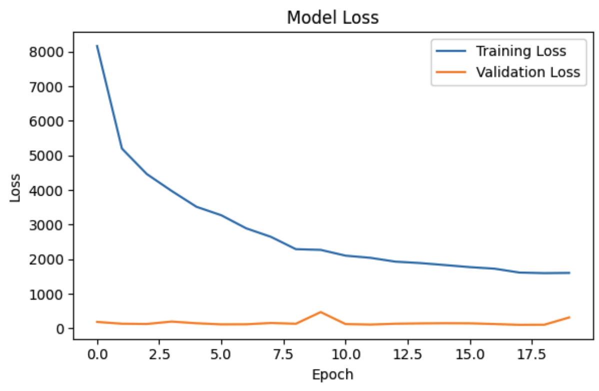
```
Mean Squared Error: 313.8533
R²: 0.7093
```

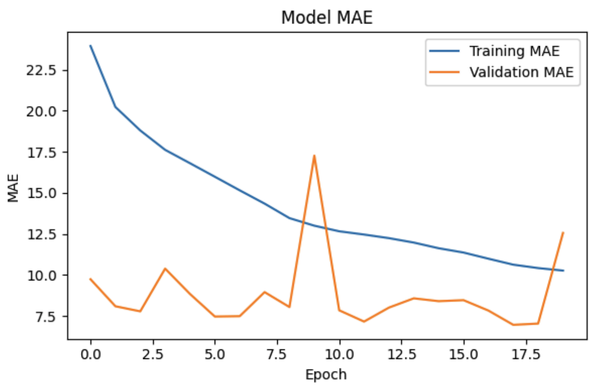


```
m (kg): 1.3547
Mt: 0.9506
W (mm): 0.9390
ep (KW): 0.8237
ec (cm3): 0.8230
Ft_NG-BIOMETHANE: 0.7286
Ft_LPG: 0.6828
Ft_E85: 0.6379
```

Here above we see the results as for the MSE and the R-square value for the test set. These are also listed in the final comparison table further below.
On the right side we see the feature importance listed. As described below, one of the variables m (kg) or Mt needs to be dropped due to crosscorrelation.

Below we see the training data history. The validation MAE shows significant fluctuations, suggesting potential overfitting. Additionally, the validation loss and validation MAE have stagnated and, at times, even deteriorated, reinforcing the concern of overfitting. It is essential to implement further parameter optimization in the final model to address this issue also because the overall model performance is comparatively bad.





Regarding parameter optimization (Grid Search and Cross Validation) and addressing regularization the following has been done:

Models/Techniques [2]	Grid Search	Elastic Net	Cross Validation	Elastic Net with CV
LR	No	No	No	Yes
DT	Yes[3]	–	No	No
RF	No	–	No	Yes, –
XG	No	– [4]	No	Yes, –[5]
Deep learning (DNN)	No	No	No	No

As seen above, Elastic Net with CV was the preferred technique addressing concerns of overfitting and collinearity. Elastic Net with CV is an automatic way of tuning the regularization parameters (alpha and l1_ratio).

However, this preferred technique was wrongly used for RF and XG. Here is the corrected table as plan:

Models/Techniques	Grid Search	Elastic Net	Cross Validation	Elastic Net with CV
LR	Optional	Yes (given multicollinearity if not treated during preprocessing)	Yes (evaluate performance and avoid overfitting)	Yes
XG	Yes (opt. parameters)	–	Yes (evaluate performance and avoid overfitting)	–
Deep learning (DNN)	No	Yes (L2)	No	No

This new table is obviating DTs and RFs, since only LR, XG and DNN will remain as the main three models to work further on.

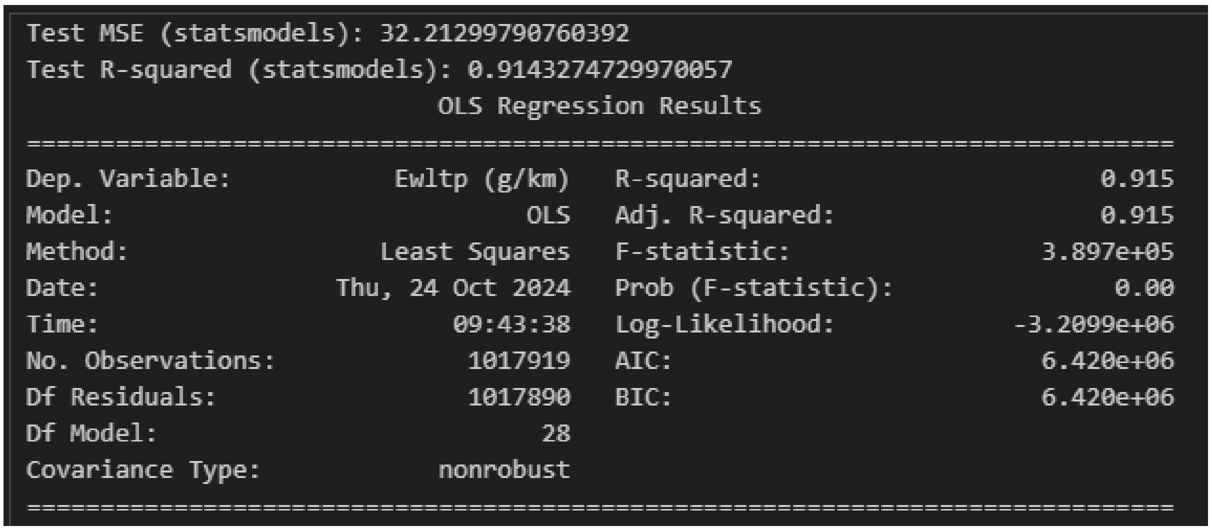
[2] Here is briefly the purpose of each technique:

- Grid Search for Hyperparameter tuning
- Elastic Net for Regularization
- Cross Validation for model validation and generalisation
- Elastic net with CV for Regularization and Hyperparamter tuning.

[3] “Pruning” was also performed.
[4] Not applicable for tree-based models.
[5] It was performed, but this is also not applicable for decision-trees based models.

3- Interpretation of results

As mentioned before, Mean Squared Error (MSE) along with R² have been consistently used. See for instance the output of this LR model:

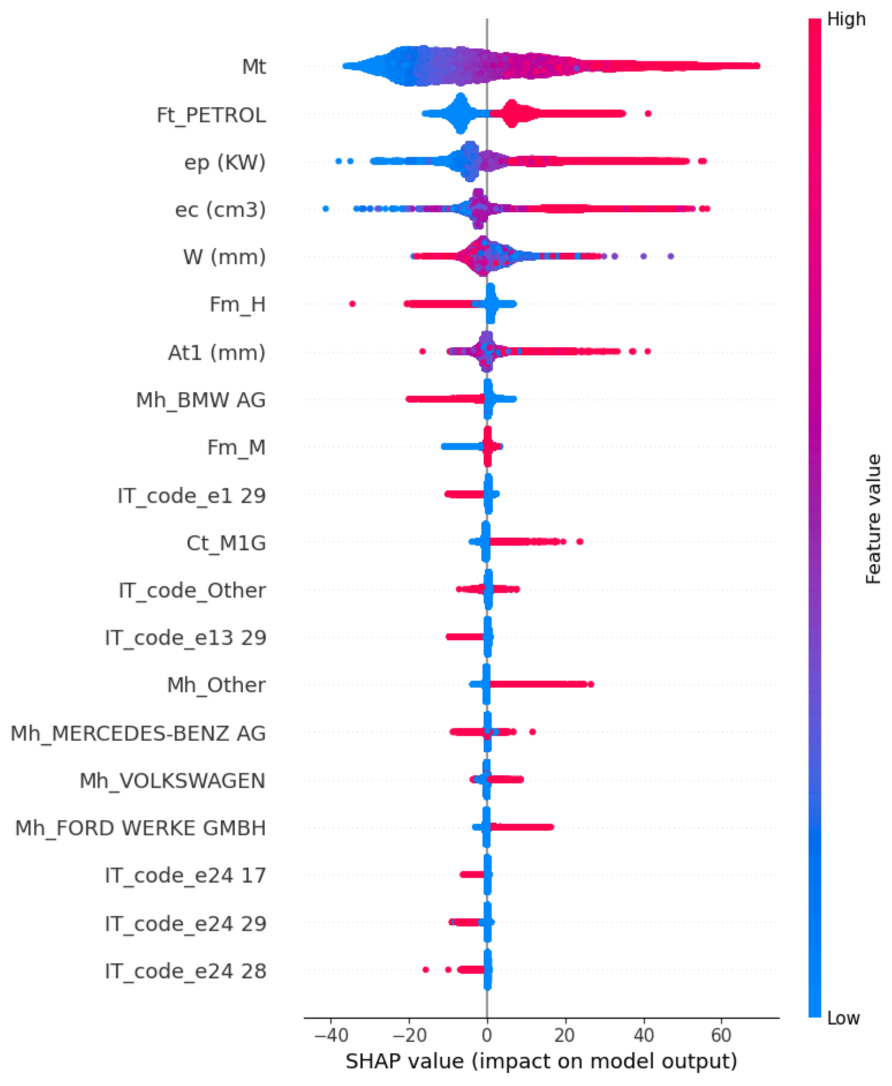


```
Test MSE (statsmodels): 32.21299790760392
Test R-squared (statsmodels): 0.9143274729970057
                            OLS Regression Results
==============================================================================
Dep. Variable:            Ewltp (g/km)   R-squared:                       0.915
Model:                             OLS   Adj. R-squared:                  0.915
Method:                  Least Squares   F-statistic:                 3.897e+05
Date:                 Thu, 24 Oct 2024   Prob (F-statistic):               0.00
Time:                         09:43:38   Log-Likelihood:            -3.2099e+06
No. Observations:              1017919   AIC:                         6.420e+06
Df Residuals:                  1017890   BIC:                         6.420e+06
Df Model:                           28
Covariance Type:             nonrobust
==============================================================================
```

Since MSE represents the average squared difference between predicted and actual values, in this case, an MSE of 32.21 indicates that, on average, the squared deviation between the model's predictions and the actual values of Ewltp (g/km) is 32.21.

However, more attention has been paid to the metric R². The reason is that R² helps to understand the models' overall effectiveness and interpretability in capturing the variance in CO_2 emissions. While MSE is informative for understanding the magnitude of error, R² provides a clearer picture of model quality in a **scale-independent** manner.

Regarding interpretability, apart from the one mentioned before (feature importance), SHAP values have been attempted on the XGBoost model. See the following visualization:



This can be interpreted as follows: The features are ranked vertically based on their importance. Features at the top have a larger impact on the model's predictions.

The color of the dots represents the direction of the feature's impact:

- Red dots indicate features that increase the prediction (positive impact).
- Blue dots indicate features that decrease the prediction (negative impact).

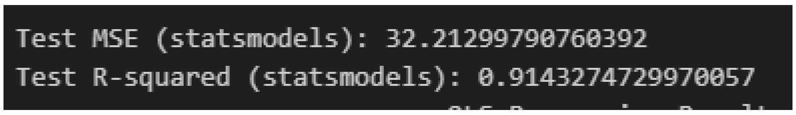
The horizontal spread of the dots for a feature shows the range of values that the feature takes in the dataset. A wider spread indicates a larger variation in the feature's values.

The position of the dots on the horizontal axis shows how much a particular feature value contributes to the model's prediction. Dots farther to the right indicate a larger positive contribution, while dots farther to the left indicate a larger negative contribution.

Finally, and regarding the question about what elements have improved (or not) the models tested so far:

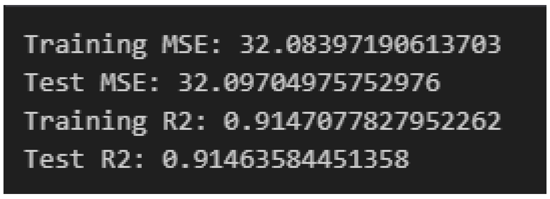
LR: Multicollinearity and overfitting has been a concern. Both have been addressed via Elastic Net with CV. However, no major improvement has been observed:

Here the results without Elastic Net CV:



```
Test MSE (statsmodels): 32.21299790760392
Test R-squared (statsmodels): 0.9143274729970057
```

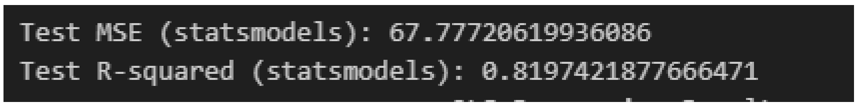
And here after Elastic Net CV was used:



```
Training MSE: 32.08397190613703
Test MSE: 32.09704975752976
Training R2: 0.9147077827952262
Test R2: 0.91463584451358
```

Training MSE and Test MSE are very close, and the Training R² and Test R² values are also very similar (both around 0.92). This indicates that the model is generalizing well, which means overfitting does not seem to be a significant issue here.

Another concern expressed by the team has been the large impact (explainability) of the fuel consumption variable. A model without the variable was ran, and as expected, the performance dropped by around 10 points:



```
Test MSE (statsmodels): 67.77720619936086
Test R-squared (statsmodels): 0.8197421877666471
```

As already mentioned, Elastic Net with CV was applied to DTs and RFs models to address overfitting. However, as ascertained before, this method is not applicable for these models. In any case, after application not significant improvement could be attested.

Comparison of semi-final models: LR, XG Boost and DNN

As mentioned before, we have tested several algorithms, yet given the results and nature of the problem, we started to focus only on Linear Regression, XG Boost and DNN. Here are the preliminary results (R2) with a note about optimization.

Model	Score	Optimization	Note:
Linear Regression	Training R2: 0.84 Test R2: 0.84 Mean R-squared across folds: 0.82	Regularization with elastic net (Lasso and Ridge)	Also CV = 5 was performed to double check generalizability
XG Boost	Best Model Test R-squared: 0.97 Mean R-squared across folds: 0.95	Grid search (Hyperparameter tuning)	Also CV = 5 was performed to double check generalizability
DNN	Test set: R-squared: 0.71	No - see optimization proposal	No - see optimization proposal

Optimization Proposal:

For model comparison, we utilized a default preprocessed dataset across all models. Early findings indicate that further optimization of the dataset is necessary along the pipeline. This especially pertains to handling outliers and determining the optimal stage in the preprocessing pipeline for their treatment. Several models were compared, but given our objective of predicting a continuous variable, classifier machine learning models proved insufficient, as binning the target variable led to a loss of granularity and precision. Nonetheless, these classifier models were trialed on the default dataset to explore their potential.

A potential avenue for optimization includes incorporating a weight parameter for recurring entries, i.e. the frequency of specific attribute combinations (which have been dropped as duplicates), allowing these weights to directly influence model training. The statistical accuracy of predictions of real world-data would be enhanced as the weighted algorithms punish errors on more frequent observations more. Also, having tested the various models on a standardized minimal dataset, some data previously dropped could be reintroduced. This regards alternative NaN handling (so far all NANs have been dropped) as well as potentially reintroducing whole columns. For example “year” and “country” in the minimal-series datasets could be reincorporated. While this might improve results for the XGBoost models, it could be less valuable in a real-world business context where attributes like registration year and country may be less relevant. For a systematic approach to optimizing the final data preprocessing pipeline and feature selection, we propose systematically comparing results from a quick linear regression model to gauge how different datasets impact prediction precision.

The second step would then involve applying XGBoost and deep neural networks (DNN) to the refined dataset(s), as these have shown considerable potential in our preliminary trials and in the literature. These models will be tested with an optimized dataset, cross-validation, and GridSearchCV for hyperparameter tuning, as well as keeping an eye on overfitting. However, it's worth noting that these approaches, particularly XGBoost with hyperparameter optimization, demand extensive computational resources, with some runs taking over two hours. As such, pipeline optimizations will initially be assessed using a fast linear regression model.

Although the regression DNN currently performs worse than comparison models, this result was obtained without some crucial preprocessing steps, specifically outlier removal for the target variable. Due to the high computational requirements, hyperparameter optimization has not yet been approached. Nonetheless, we will pursue DNN improvements due to its potential. We plan to explore the following strategies to reduce overfitting:

- **Early Stopping**: Halt training when validation loss begins to rise.
- **Batch size adjustment:** The batch size needs to be set to a power of 2 (e.g., 16, 32, 64) due to binary computer hardware. This choice optimizes computations on GPUs or TPUs and optimizes calculation time.
- **Regularization**: Introduce techniques like L1 or L2 regularization to penalize model complexity.
- **Data Augmentation**: Expand training data size and diversity to minimize overfitting to specific examples.
- **Model Architecture**: Experiment with simpler architectures or add dropout layers to reduce overfitting.

It's also worth noting that XGBoost with hyperparameter tuning, like DNN, requires significant computational resources, with some runs taking over two hours. Thus, dataset optimization will initially be tested using the faster linear regression model. Additionally, it is recommended to save a compiled, computation-intensive model (such as XGBoost or DNN) using *joblib* or *pickle*, and apply it to various optimized datasets to test their performance. This approach allows for further optimization of data preprocessing and selection, enhancing overall model effectiveness.

Appendix

We began testing Model variations systematically on our minimal_dataset and will expand this approach for the final optimization.

Dataset:
minimal_without_fuel_consumption_tn20_mcp00.10.parquet

							(*) Test_size = 20%, random_seed = 42					
#	Dataset	Model Type (*)	With duplicates	With weight (freq)	Outliers treatment	Features	R² train	R² (c) valid	R² test	RMSE train	RMSE valid	RMSE test
7	minimal_without_fuel_consumption_tn20_	XGBoost (Reg.)	yes	no	no	all	0.98	0.98	0.98	4.47	4.52	4.5
9	minimal_without_fuel_consumption_tn20_	XGBoost (Reg.)	no	no	no	all	0.98	0.98	0.98	4.69	4.87	4.85
8	minimal_without_fuel_consumption_tn20_	XGBoost (Reg.)	yes	no	no	wo: m (kg), At1, At2, W, instead WA (m2)	0.97	0.97	0.97	5.69	5.69	5.72
11	minimal_without_fuel_consumption_tn20_	XGBoost (Reg.)	no	no	no	wo: m (kg), At1, At2, W, instead WA (m2)	0.97	0.97	0.97	5.9	6.04	6.01
4	minimal_without_fuel_consumption_tn20_	Lin. Reg.	no	no	no	all	0.85	0.85	0.85	12.57	-	12.55
5	minimal_without_fuel_consumption_tn20_	Lin. Reg.	no	no	no	wo: m (kg)	0.85	0.85	0.85	12.88	-	12.87
6	minimal_without_fuel_consumption_tn20_	Lin. Reg.	no	no	no	wo: m (kg), At1, At2, W, instead WA (m2)	0.84	0.84	0.84	12.98		12.96
1	minimal_without_fuel_consumption_tn20_	Lin. Reg.	yes	no	no	all	0.81	0.81	0.81	15.14	-	15.1
2	minimal_without_fuel_consumption_tn20_	Lin. Reg.	yes	no	no	wo: m (kg)	0.79	0.79	0.79	16.01	-	15.97
3	minimal_without_fuel_consumption_tn20_	Lin. Reg.	yes	no	no	wo: m (kg), At1, At2, W, instead WA (m2)	0.78	0.78	0.78	16.2	-	16.16