**CO2 emissions by vehicles**

**Project by**

**Roberta Hübner,**

**Mehdi Rahmati and**

**Jörg Grünklee**

[**1. Objective 2**](#_or6oz7nho3w8)

[**2. Scope 2**](#_mmnlwot0ohs7)

[**3. Visualisations 2**](#_ggkbzyfamey8)

[**4. Preprocessing 7**](#_3siqt4pb5wse)

[4.1. Removing redundant features 7](#_gwmsb9aqrxwc)

[4.2. Removing highly correlated features 7](#_s3zwncffr0bl)

[4.3. Selected features 8](#_8fvhqqvidwse)

[4.4. Handling missing data 8](#_ijqzclos746v)

[4.4.1. Emissions (Ewltp (g/km)) 8](#_fzu7b8tozm8e)

[4.4.2. Fuel Consumption 8](#_fquy9wrjg7ef)

[4.4.3. Engine Power (ep (KW)) 8](#_lckd2oth43i8)

[4.4.4. Emission Reduction from Innovative Technologies (Erwltp (g/km) ) and Innovative Technologies (IT) 9](#_kzst9qc9hngw)

[4.5. Other changes made to the data 9](#_2a7xgyy6jmev)

[4.5.1. Renaming 9](#_ghz31wgeacle)

[4.5.2. One hot-encoding 9](#_7vnq8pgjbzgz)

[**5. Modeling 9**](#_f111vsfonyb2)

[5.1. Standardization of the data 10](#_e9sng2h8ggzm)

[5.1.1. Target encoding 10](#_l10cx1e93b8a)

[5.1.2. Splitting the data into train, test and validation sets 11](#_muicgu8rmhfu)

[5.1.3. Data normalization 11](#_olpwlh3kyzfg)

[5.2. Regression 11](#_apfd2gyl3r8)

[5.2.1. Models 11](#_vfqy6t9rmdmk)

[5.2.2. Evaluation metrics 12](#_bhtg6j7oattq)

[5.2.3. Results 12](#_kfjnz945quq4)

[5.2.4. Optimization 15](#_o3rur881uegn)

[5.2.5. Interpretation 17](#_ldb9na98ygzc)

[5.3. Classification 18](#_ai1pexu4udsj)

[5.3.1. Models 18](#_5dyvimipkzww)

[5.3.2. Evaluation metrics 19](#_rpn10qudj4ry)

[5.3.3. Results 19](#_fe6vextijban)

[5.3.4. Optimization 22](#_h4hm2lchg8ic)

[5.3.5. Interpretation 24](#_ea441ls585y3)

# Objective

We aim to identify the vehicles with the most CO2 emissions and find the cars contributing the most to air pollution.

# Scope

We will use the dataset “Monitoring of CO2 emissions from passenger cars” of 2021, distributed by the European Environment Agency.

The data is discussed in the [Data Audit File](https://docs.google.com/spreadsheets/u/0/d/1ioeW2hdSXdYjk5JqGuD8VdjvgsGTSTa76-5IRSZh1XQ/edit).

# Visualisations

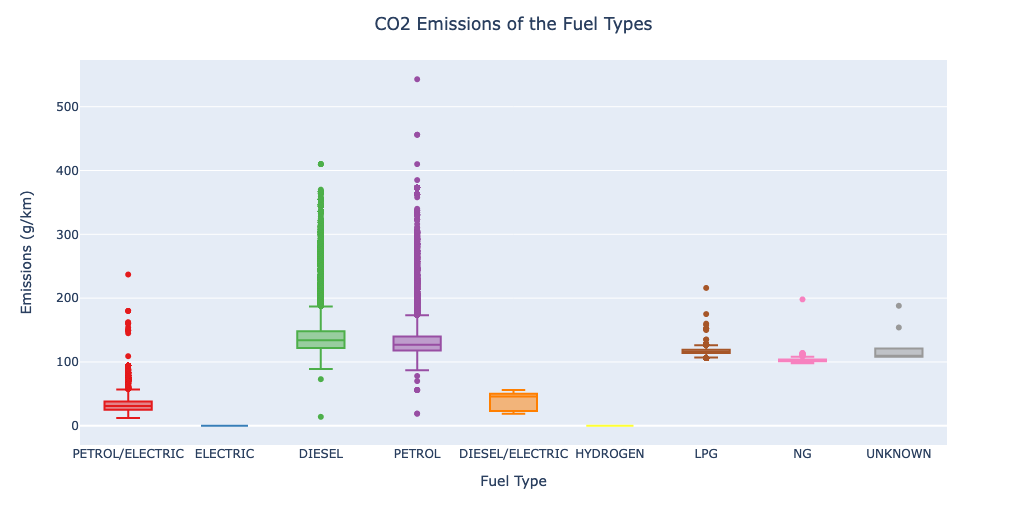


Figure 1: CO2 emissions of cars with different fuel types.

The graph for the emission per fuel type shows (Figure 1) that the emissions for cars using electric and hydrogen are set to zero, most likely due to the difficulty of calculating emission values for electricity usage and hydrogen manufacturing. The highest emission values are seen for vehicles using petrol, followed by diesel. The average emissions are around 100-150 grams of CO2 per kilometer, whereas hybrid cars have an average emission of around 40 grams of CO2 per kilometer. From the distribution of the data points, especially for petrol, we can see that there is a wide range of emission values with a substantial amount of cars with emissions up to 400 grams per kilometer with some outliers even going up to around 550 grams per kilometer. In summary, petrol and diesel, the most widely used fuels, are the fuels with the highest emissions overall, while there are only differences in average emissions between mono-fuel and hybrid vehicles.

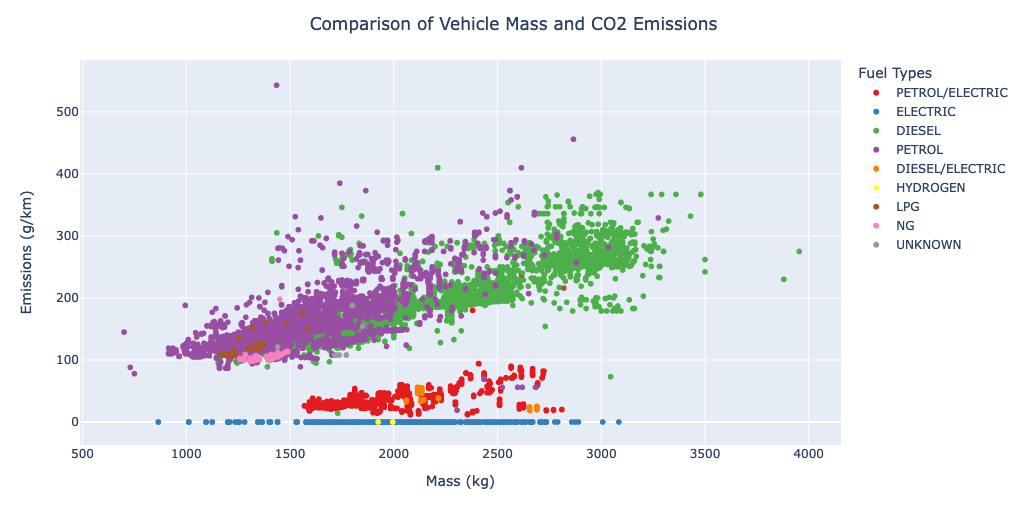
****

Figure 2: Relationship between vehicles mass and their CO2 emissions.

The graph (Figure 2) shows that the emissions increase with the mass of the car in a linear fashion. For diesel fuel, we can also see that there is not one mass favored, as the distribution is quite homogenous and does not aggregate or dissipate at around a specific mass. For petrol, the emissions seem to be more unstable for higher mass vehicles, with fewer vehicles with a mass over 2000 kg compared to diesel. It seems that the dependency of hybrid cars emissions on vehicle mass is happening with a lower slope while for other cars it happens with a higher slope. The hybrid types have much less emissions, most likely due to the calculations that use weighted values. Cars running on gas (LPG, NG) are only seen weighing between 1200 and 1600 kg, with no exceptions. This could be either due to technological limitations or the fact that construction is not cost-effective. Overall we can see that diesel is the fuel used for light to heavy vehicles, with heavier vehicles having higher emissions, closely followed by petrol.

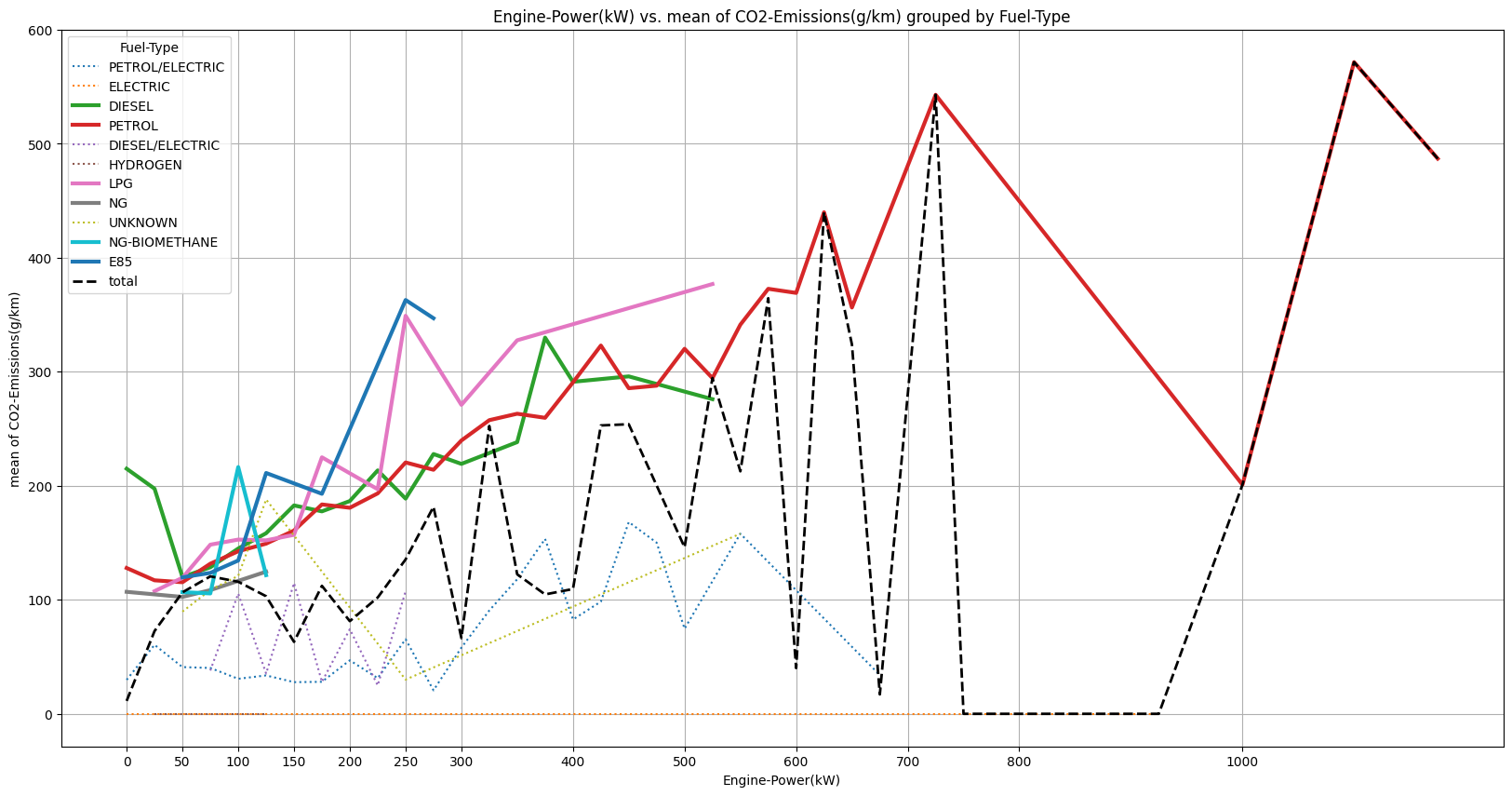


Figure 3: The relationship between engine power and mean CO2 emissions among different fuel types.

According to Figure 3, it seems that observing the relationship between engine power and CO2 emissions without taking into account the fuel types (see the black-dashed line) might be misleading. However, by grouping cars by the fuel types, one can see that is correlated with CO2 emissions, more specifically in the classic Fuel-Types. It seems that pure or hybrid fuel types can mislead the interpretation of the relationship between engine power and CO2 emissions since they have no CO2 emissions.

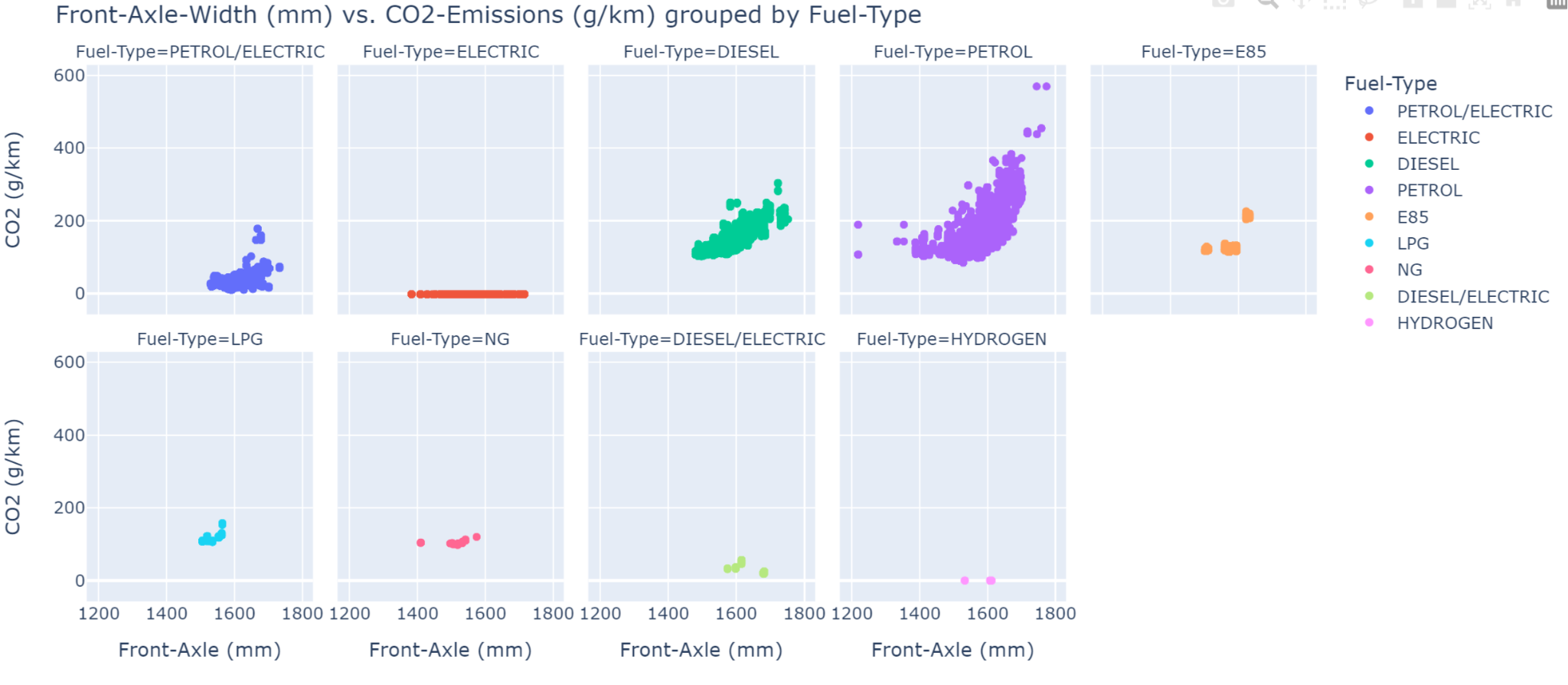


Figure 4: The relationship between front-axle width and CO2 emission among different fuel types.

According to Figure 4, the width of the front axle is not correlated with the CO2 emissions except for diesel and petrol types where a linear relationship between the width of the front axle and CO2 emissions is observable.

As Fuel consumption highly correlates with Ewltp (g/km) (correlation is 0.872), we look at the distribution of the fuel consumption (Figure 5), where we differentiate the fuel type. One can conclude that the Fuel consumption variable is not equally informative for all the fuel modes where electric cars have 0 fuel consumption, implying that no conversion into a unified fuel equivalent is performed. This implies that the relationship between fuel consumption and Ewltp (g/km) should be examined separately for the six-car types (grouped by fuel mode)

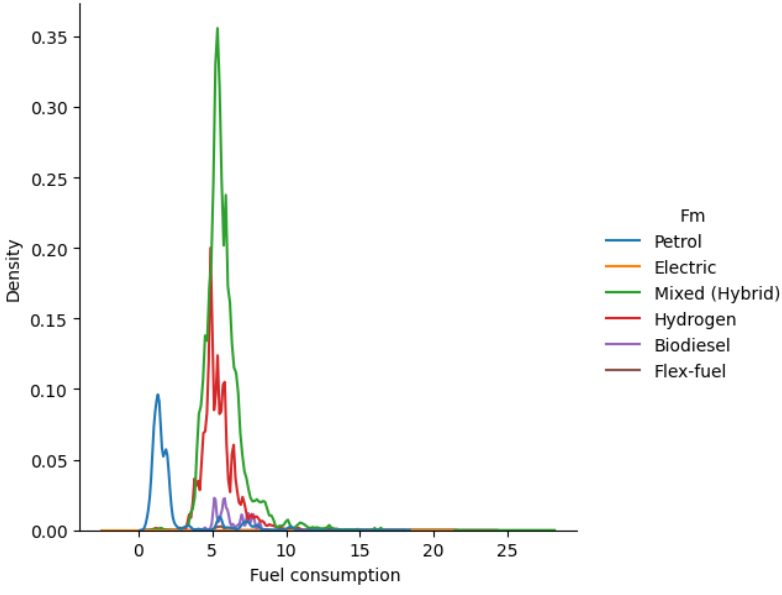


Figure 5: The distribution of the fuel consumption among different fuel types.

# Preprocessing

The dataset was filtered for cars in Germany and France, to reduce the size of the data and to concentrate on our mutually relevant countries.

## Removing redundant features

We dropped columns with no data (Ernedc (g/km), De, Vf, MMS), columns with one constant value (Status, r), and columns with unnecessary data such as dates (year, Date of registration, Country) or identification numbers (ID, VFN, Tan) and columns for electric cars and rows of electric and hydrogen cars, as their emissions are zero in this dataset. We also dropped the column Enedc (g/km), because the NEDC test is older and we will use the data of the newer WLTP test.

## Removing highly correlated features

The following feature pairs are highly correlated, as can be seen in Figure 6 and 7, so that one of them can be omitted. We dropped the feature with more missing data. (R = correlation coefficient)

* m (kg) and Mt (R = 0.99)
* At1 (mm) and At2 (mm) (R = 0.97)
* Mp, Mh, T, Cn and Mk (R>0.87)
* ec (cm3) and ep (KW) (R = 0.82), ec (cm3) has 16% missing data
* Fm and Ft (R=0.82), both have no missing data, but we need the more detailed classes of Ft to differentiate the cars, so we remove Fm

Both m (kg) and Mt are also strongly correlated with W (mm), At1 (mm), and At2 (mm) (with R ~ 0.8) and ec (cm3) and ep (KW) (with R ~ 0.7).

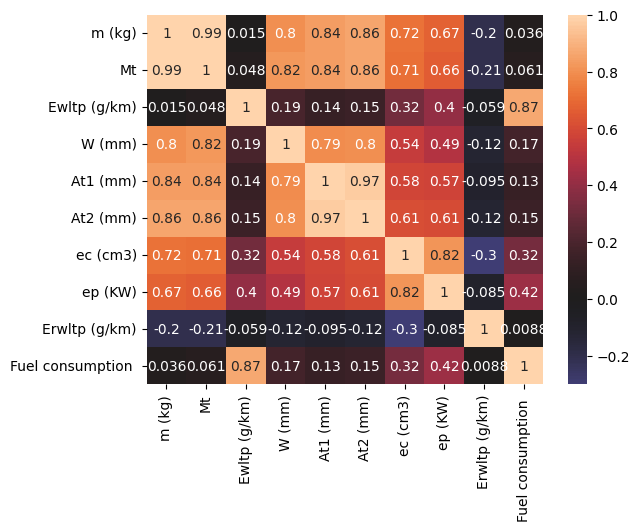
****

Figure 6: Heat map of the correlation matrix for the numerical features.

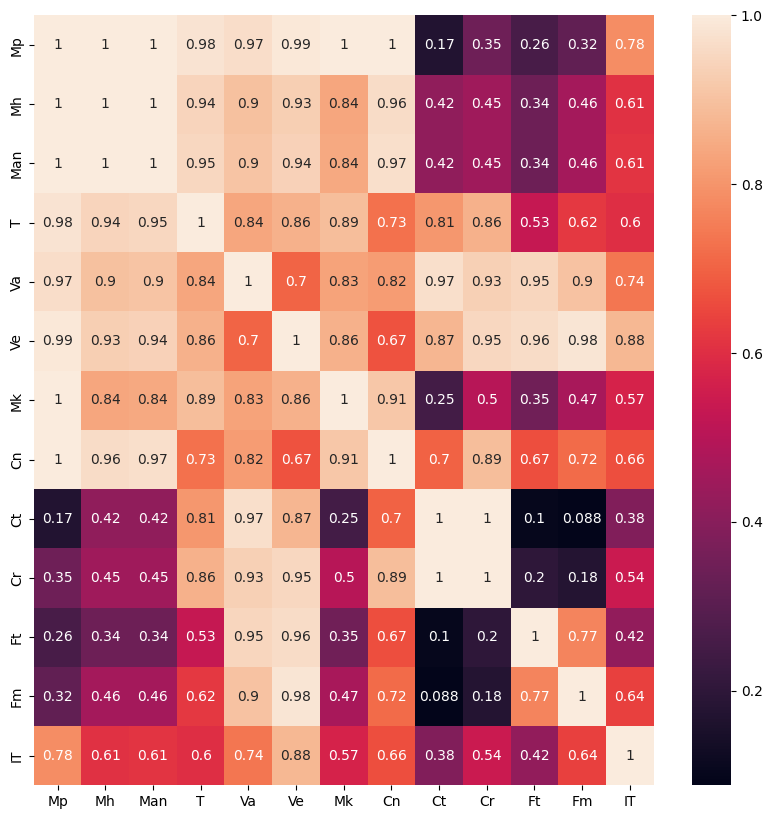
****

Figure 7: Heat map of the correlation matrix for the categorical features determined with Cramers V method.

## Selected features

After the feature selection process, the following 9 features were kept for further analysis:

Table 1: Data type of the remaining features

|  | **Feature** | **Dtype** |
| --- | --- | --- |
| 0 | Man | object |
| 1 | Cr | object |
| 2 | m (kg) | float64 |
| 3 | Ewltp (g/km) | float64 |
| 4 | Ft | object |
| 5 | ep (KW) | float64 |
| 6 | IT | object |
| 7 | Erwltp (g/km) | float64 |
| 8 | Fuel consumption | float64 |

## Handling missing data

### Emissions (Ewltp (g/km))

We dropped all 6365 NaN rows of the emissions column, as they are only a small portion (0.18%) of the data.

### Fuel Consumption

After dropping the emissions NaN rows, we are left with 0.74% of the values of Fuel Consumption being NaN values, which again is a small percentage and therefore we decided to drop them entirely.

### Engine Power (ep (KW))

There were two entries with no data, not only in the ep (KW) column but also in most other columns including the most important emission column, so these entries were dropped from the dataset.

### Emission Reduction from Innovative Technologies (Erwltp (g/km) ) and Innovative Technologies (IT)

Firstly we drop all entries where no innovative technology was specified and no emission reduction was given, as we can not estimate any values for those cases.

After that, we investigate entries where there is no value for the emission reduction, although an innovative technology was specified. We found entries of the innovative technology "e2 29" with no emission reduction values, which were then filled by taking the mean over all the entries of the "e2 29" technology.

With this, all missing values were taken care of.

## Other changes made to the data

### Renaming

We found that the column name for Fuel Consumption had an extra space at the end, which we rectified. Furthermore, after displaying the unique values of the Fuel Type feature we could see that a label was incorrectly labeled as "NG" instead of "NG-BIOMETHANE", which we also changed.

### One hot-encoding

The IT feature was changed into a binary feature with 1 if any innovative technology was specified and 0 if none was specified.

The remaining categorical features Man, Cr, and Fuel Type were changed into indicator variables.

# Modeling

We study the performance of several models, divided into two groups: Regression and Classification.

## Standardization of the data

### Target encoding

First, we sort the numerical values of the target variable into 7 bins according to the European Union energy labels (see Figure 8) to have a categorical target for the classification modeling.

## 

Figure 8: European Union energy labels according to emissions*.*

After counting how many cars each of the 7 bins contains we obtain the following (Table 2):

Table 2: The number of samples falling into each bin of CO2 emissions groups.

| **Label** | **Class** | **Count** |
| --- | --- | --- |
| A | 0 | 1212500 |
| B | 1 | 1214250 |
| C | 2 | 518359 |
| D | 3 | 177188 |
| E | 4 | 156811 |
| F | 5 | 143041 |
| G | 6 | 73948 |

By a closer look at Table 2, one can observe that the first two bins are mostly populated, containing approximately 1.2 Mio registrations.

### Splitting the data into train, test, and validation sets

Namely because of the imbalanced distribution of the explanatory variable we do stratify the sampling by using the option stratify to keep a mix of all classes for each set.

On 10% of the data, we will do hyperparameter tuning, i.e. validation, while for the training we use 20% of all the data. The rest of the instances will be used as our test set. Simultaneously we perform the splitting for both, for the regression and the classification model.

### Data normalization

As part of the standardization process, we normalize all numerical variables using a Standard Scaler.

## Regression

### Models

We fit the following models with our train sets and evaluate the data on the test sets to determine the best-performing model for our problem.

* Linear Regression
* Elastic Net
* Decision Tree Regressor
* XGBoost Regressor
* Neural Network Regressor

We also made predictions on the test sets to determine if our models overfit the data or not.

The Neural Network Regressor had the following layers:

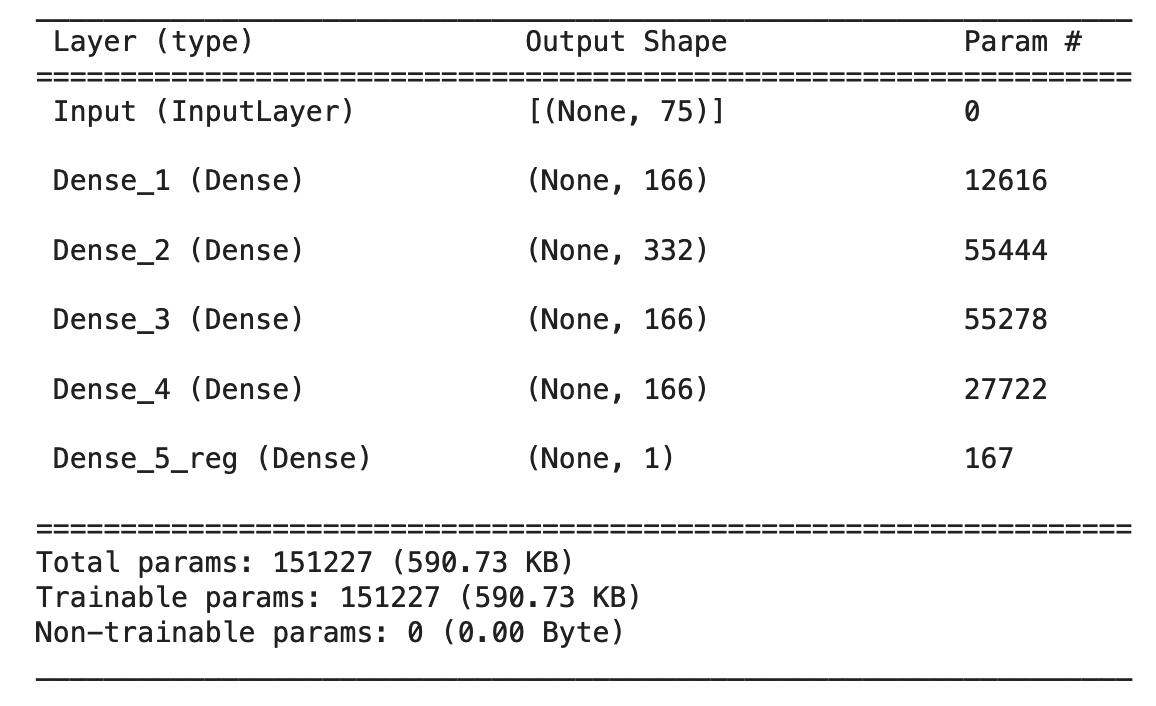


Figure 9: Layers of the Neural Network Regressor.

Although we initially implemented a neural network with dropout layers it resulted in no improvement and therefore, we chose not to use dropout layers in our final version, to keep the layout simple. Additionally, we implemented an early stopping and a learning rate reduction callback to prevent overfitting of the neural network. The neural network was run over 15 epochs with a batch size of 512 and evaluated on the validation set.

### Evaluation metrics

We used the metrics R² (R-squared) and RMSE (Root Mean Square Error) for interpreting the performance of the models.

We chose R² to have a standardized and simple-to-interpret metric for comparing our models and RMSE as a more intuitive, quantifiable metric.

### Results

As shown in the following table (Table 3), the XGBRegressor and Neural Networks Regressor give us the best performance with the lowest variance and error, with the Neural Network having the best test score, with the lowest error.

Table 3: Accuracy of the trained regression models for train and test sets

| **Model** | **Set** | **Metric** | **Score** |
| --- | --- | --- | --- |
| LinearRegression | Train | R^2 | 0.9423 |
| LinearRegression | Test | R^2 | 0.9424 |
| LinearRegression | Train | RMSE | 11.3035 |
| LinearRegression | Test | RMSE | 11.2885 |
| ElasticNet | Train | R^2 | 0.7518 |
| ElasticNet | Test | R^2 | 0.7513 |
| ElasticNet | Train | RMSE | 23.4402 |
| ElasticNet | Test | RMSE | 23.4459 |
| DecisionTreeRegressor | Train | R^2 | 0.9628 |
| DecisionTreeRegressor | Test | R^2 | 0.9629 |
| DecisionTreeRegressor | Train | RMSE | 9.0765 |
| DecisionTreeRegressor | Test | RMSE | 9.0613 |
| XGBRegressor | Train | R^2 | 0.9998 |
| XGBRegressor | Test | R^2 | 0.9998 |
| XGBRegressor | Train | RMSE | 0.7104 |
| XGBRegressor | Test | RMSE | 0.7168 |
| NN Regression | Train | R^2 | 0.9998 |
| NN Regression | Test | R^2 | 0.9998 |
| NN Regression | Train | RMSE | 0.7183 |
| NN Regression | Test | RMSE | 0.6729 |

Elastic Net on the other hand performs very poorly. This is most likely due to underfitting, as we already removed almost all irrelevant features and the model is probably penalizing the important features too much. This could probably be rectified by tuning the hyperparameters, but our other models already perform very well, so we chose to use them instead.

The Linear Regression and Decision Tree models seem to be too simple for our complex dataset as their errors seem to be still a bit too high to give a good performance.

Because of the nature of XGBooster, we had to be careful about overfitting, but the results show that both train and test sets perform the same, which does not indicate any over- or underfitting. Indeed, for all the models we could observe that the performance on train and test scores were quite similar, meaning that overfitting did not take place.

For the neural network, we plotted the change of the mean absolute error and loss per epoch.

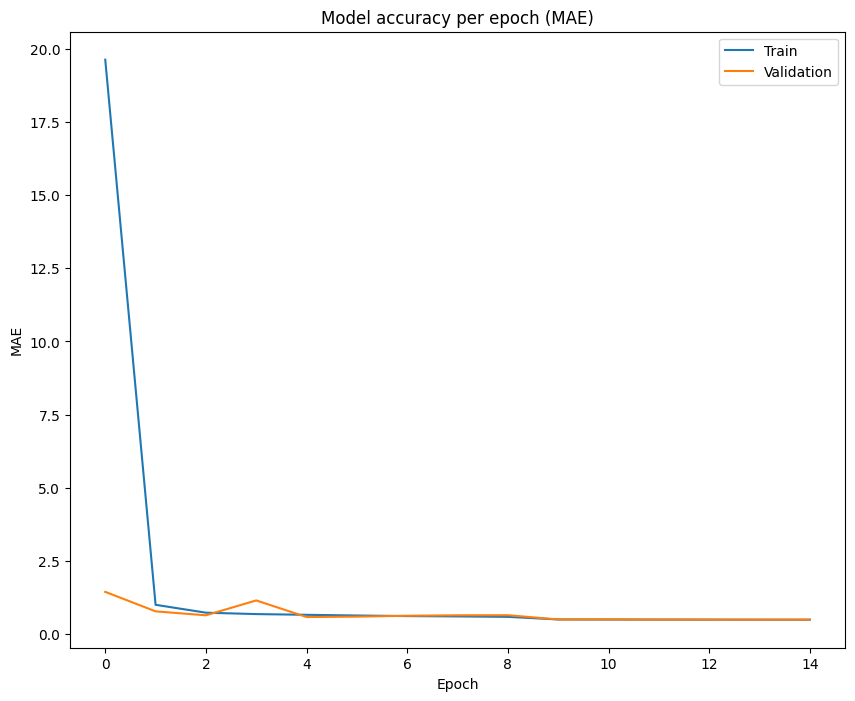


Figure 10: Neural network models mean absolute error per epoch.

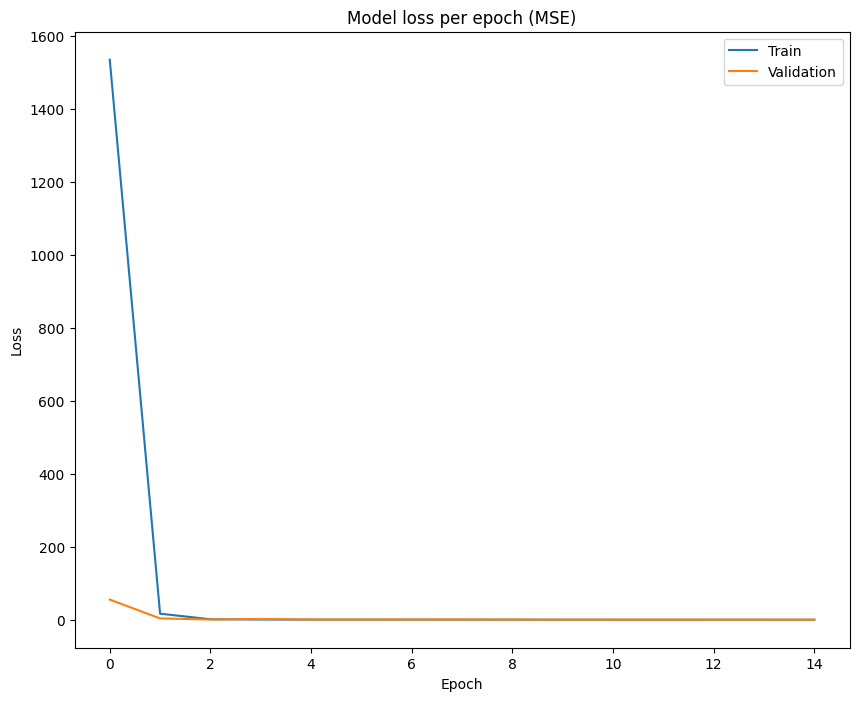


Figure 11: Neural network models mean squared error per epoch.

In the plots, we can see that the models MAE and loss function were already quite low after the first epoch and mostly stabilized after the fourth. On the other hand, the model improved until the last epoch, without stopping early or reducing the learning rate, which means it might be optimized further if we let it run for more epochs.

### Optimization

Firstly we tried to optimize the XGBRegressor even further by varying the learning rate, maximal depth, and the number of estimators.

The best parameters were determined to be:

| learning rate: | 0.1 |
| --- | --- |
| maximal depth: | 7 |
| number of estimators: | 1000 |

We then trained a new XGBRegressor with these parameters and as a result, the error was even smaller than with our previous regressor, which shows that our optimization was successful.

Table 4: Accuracy of the unoptimized and optimized XGBoost regression model.

| **Model** | **Set** | **Metric** | **Score** |
| --- | --- | --- | --- |
| XGBRegressor | Train | R^2 | 0.9998 |
| XGBRegressor | Test | R^2 | 0.9998 |
| XGBRegressor | Train | RMSE | 0.7104 |
| XGBRegressor | Test | RMSE | 0.7168 |
| XGBRegressor optimized | Train | R^2 | 0.9999 |
| XGBRegressor optimized | Test | R^2 | 0.9999 |
| XGBRegressor optimized | Train | RMSE | 0.544 |
| XGBRegressor optimized | Test | RMSE | 0.5539 |

Although our results for the Neural Network Regression model are already very good, we wanted to see if we can optimize them even more. For the optimization we used a KerasRegressor and a grid search and the used the following parameters for tuning:

| learning rate: | 0.001,0.0001 |
| --- | --- |
| optimizers: | SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam |
| batch size: | 512, 256, 128 |

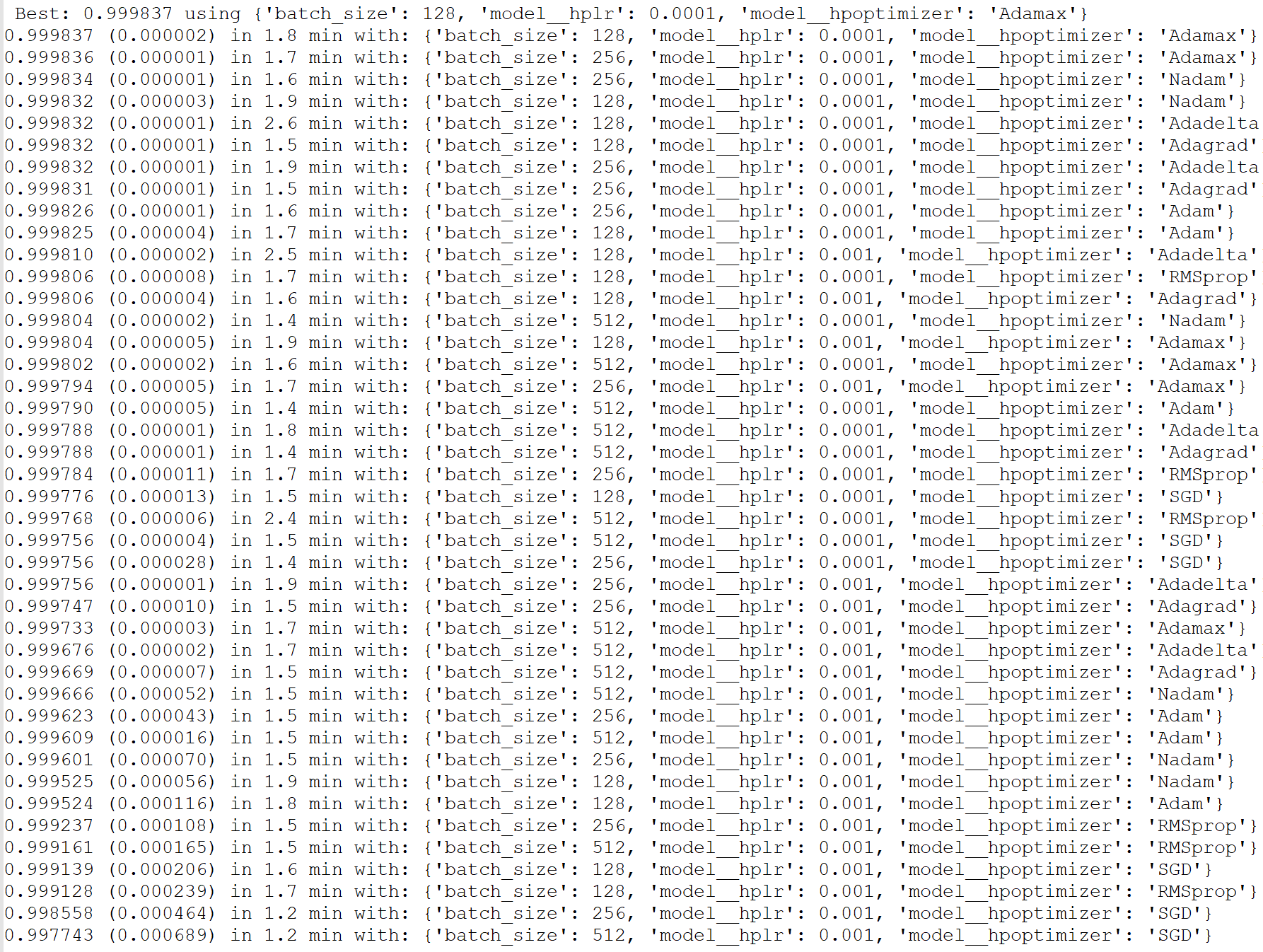


Figure 12: Optimization results of neural network regression.

With the best parameters shown in Figure 12 we fit a new neural network with the same layer structure as in Figure 9.

Table 5: Accuracy of the unoptimized and optimized Neural Network Regression.

| **Model** | **Set** | **Metric** | **Score** |
| --- | --- | --- | --- |
| NN Regression | Train | R^2 | 0.9998 |
| NN Regression | Test | R^2 | 0.9998 |
| NN Regression | Train | RMSE | 0.7183 |
| NN Regression | Test | RMSE | 0.6729 |
| NN Regression optimized | Train | R^2 | 0.9998 |
| NN Regression optimized | Test | R^2 | 0.9998 |
| NN Regression optimized | Train | RMSE | 0.594 |
| NN Regression optimized | Test | RMSE | 0.6163 |

The results are shown in Table 5. We can see that with the optimized model we get much better values than with our random neural network regression model. Interestingly the XGBoost regression algorithm still gave us a slightly better model (see Table 4)

### Interpretation

In the feature importance plot for our XGBRegressor, we can see that the fuel type petrol was the single most important feature for determining the emissions of a car. This is rather surprising because this feature is not a numerical feature but a categorical one. This clearly shows how important it is for the emission value if the car uses petrol as a fuel or not. It would be interesting to see if we would still get similar good predictions if we remove or penalize the fuel type.

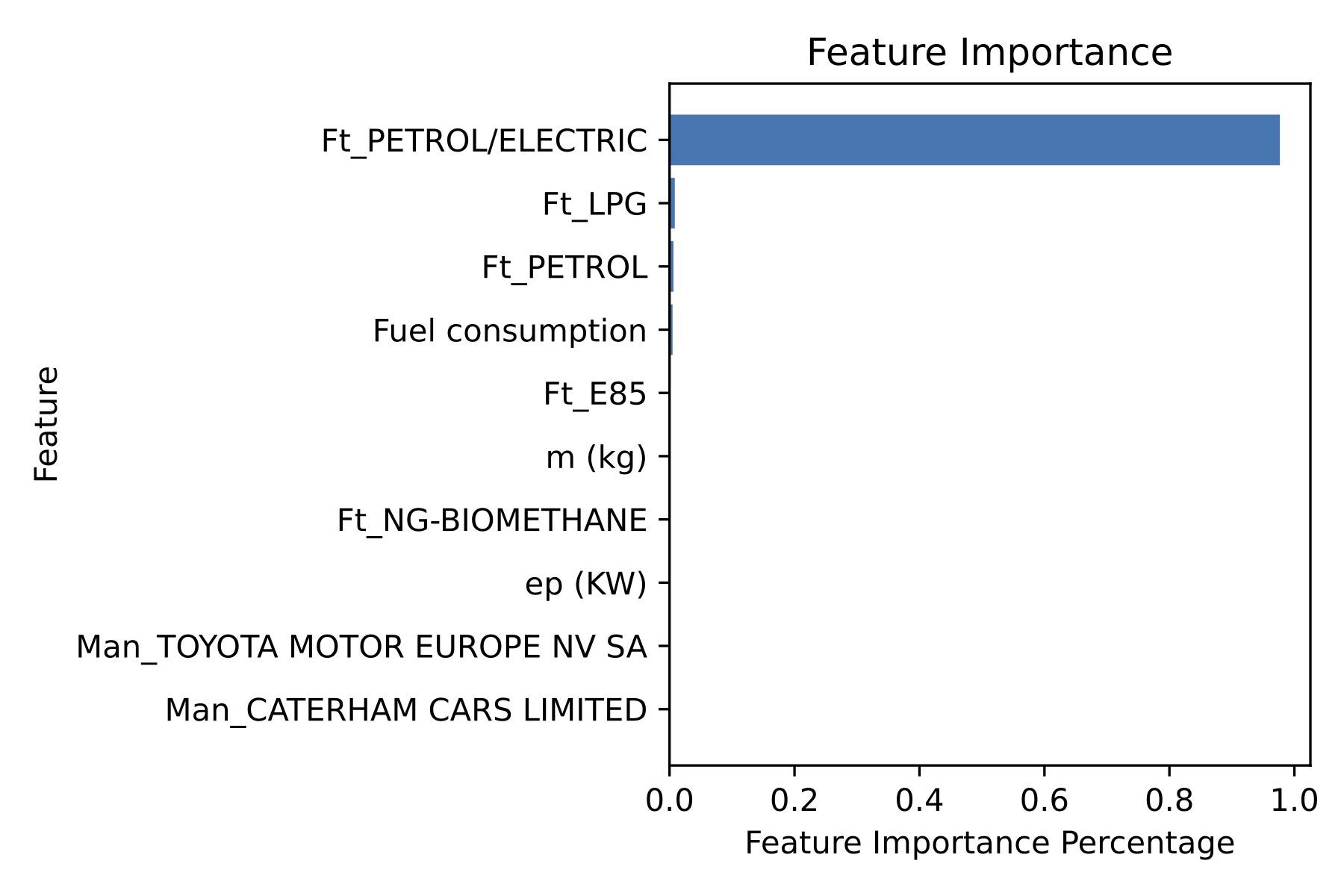


Figure 13: The 10 most important features of the optimized XGBRegressor.

## Classification

### Models

We fit the following models with our test sets and perform predictions on our train sets to determine the best-performing model for our problem.

* Logistic Regression
* K-Nearest Neighbors
* Decision Tree Classifier
* XGBoost Classifier
* Neural Network Classifier

We also made predictions on the test sets to determine if our models overfit the data or not.

The Neural Network Classifier had the following layers:

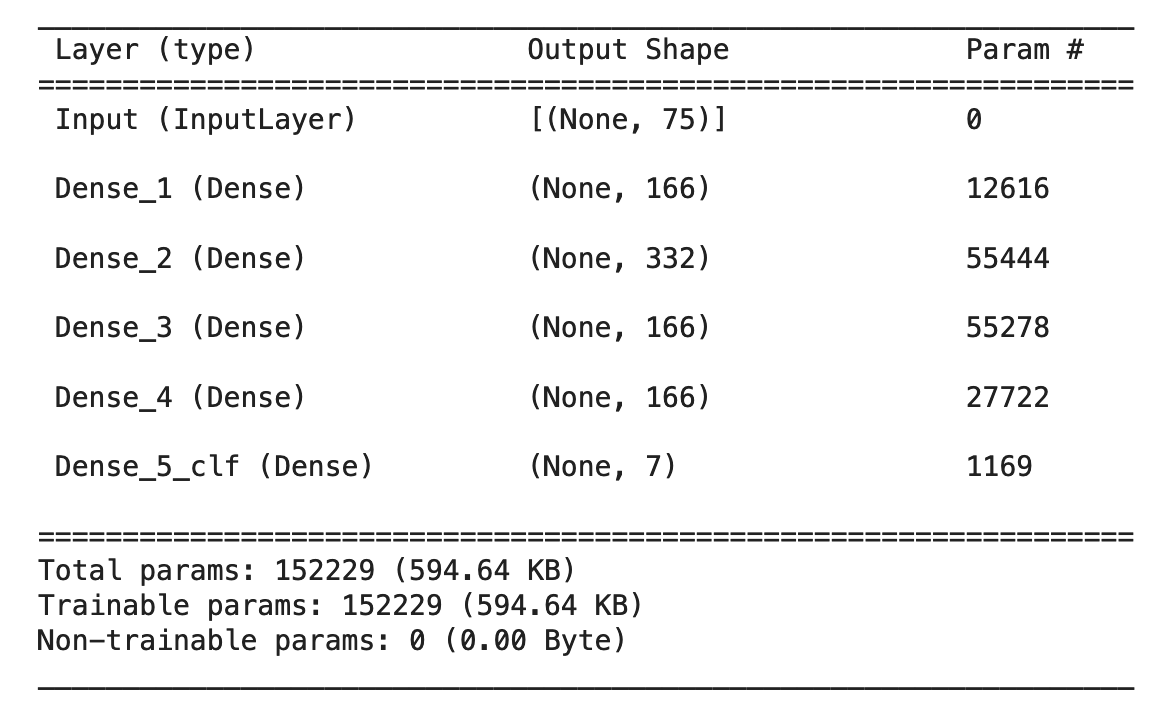


Figure 14: Layers of the Neural Network Regressor.

As with the Neural Network Regressor, we implemented an early stopping and a learning rate reduction callback to prevent overfitting of the neural network and it was run over 15 epochs with a batch size of 512.

### Evaluation metrics

We used the metrics Accuracy and the F1-score for interpreting the performance of the models.

We choose accuracy to have a straightforward metric for comparing our models, but also choose the F1-score because we have imbalanced data and have to take that into account.

### Results

As shown in the following table the XGBClassifier and Neural Networks Classifier give us the best performance with the highest accuracy and F1-score. Surprisingly the XGBooster gave us an even higher score than the neural network, although both models are yet to be optimized.

Table 6: Accuracy of the trained classification models for train and test sets

| **Model** | **Set** | **Metric** | **Score** |
| --- | --- | --- | --- |
| LogisticRegression | Train | Accuracy | 0.9414 |
| LogisticRegression | Test | Accuracy | 0.9408 |
| LogisticRegression | Train | F1 | 0.9414 |
| LogisticRegression | Test | F1 | 0.9407 |
| KNeighborsClassifier | Train | Accuracy | 0.9832 |
| KNeighborsClassifier | Test | Accuracy | 0.9825 |
| KNeighborsClassifier | Train | F1 | 0.9832 |
| KNeighborsClassifier | Test | F1 | 0.9825 |
| DecisionTreeClassifier | Train | Accuracy | 0.8382 |
| DecisionTreeClassifier | Test | Accuracy | 0.8378 |
| DecisionTreeClassifier | Train | F1 | 0.8165 |
| DecisionTreeClassifier | Test | F1 | 0.8161 |
| XGBClassifier | Train | Accuracy | 0.9858 |
| XGBClassifier | Test | Accuracy | 0.9856 |
| XGBClassifier | Train | F1 | 0.9858 |
| XGBClassifier | Test | F1 | 0.9856 |
| NN Classification | Train | Accuracy | 0.9832 |
| NN Classification | Test | Accuracy | 0.9830 |
| NN Classification | Train | F1 | 0.9831 |
| NN Classification | Test | F1 | 0.9830 |

It seems that the Logistic Regression and one Decision Tree were too simple models for your data, as they did not perform as well as the others, especially the Decision Tree. On the other hand, the K-Nearest Neighbors also performed rather well, which is likely due to some kind of clustering of the data. In the graph “Comparison of Vehicle Mass and CO2 Emissions” on page 3, it looks like the vehicles could be clustered by their fuel type, although there could be some problems distinguishing the petrol types.

Same as for the regression model, we observe that the train and test scores are quite similar, so we do not seem to have any over- or underfitting.

For the neural network, we plotted the change in the accuracy and loss per epoch.

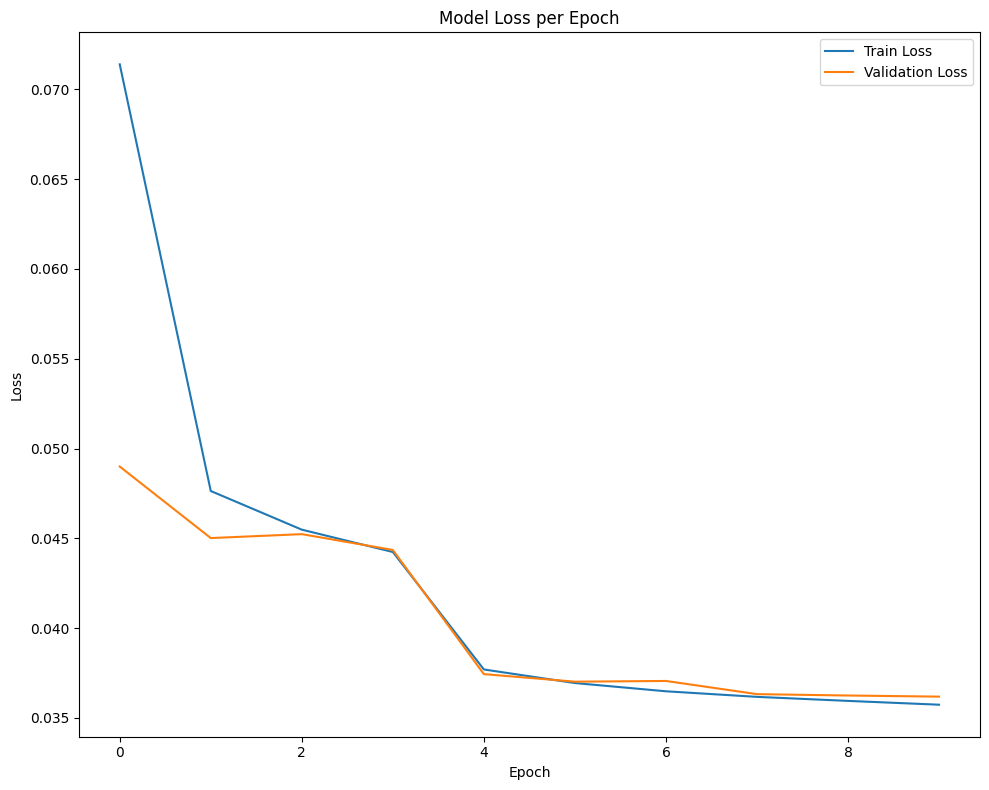


Figure 15: Neural network models accuracy per epoch.

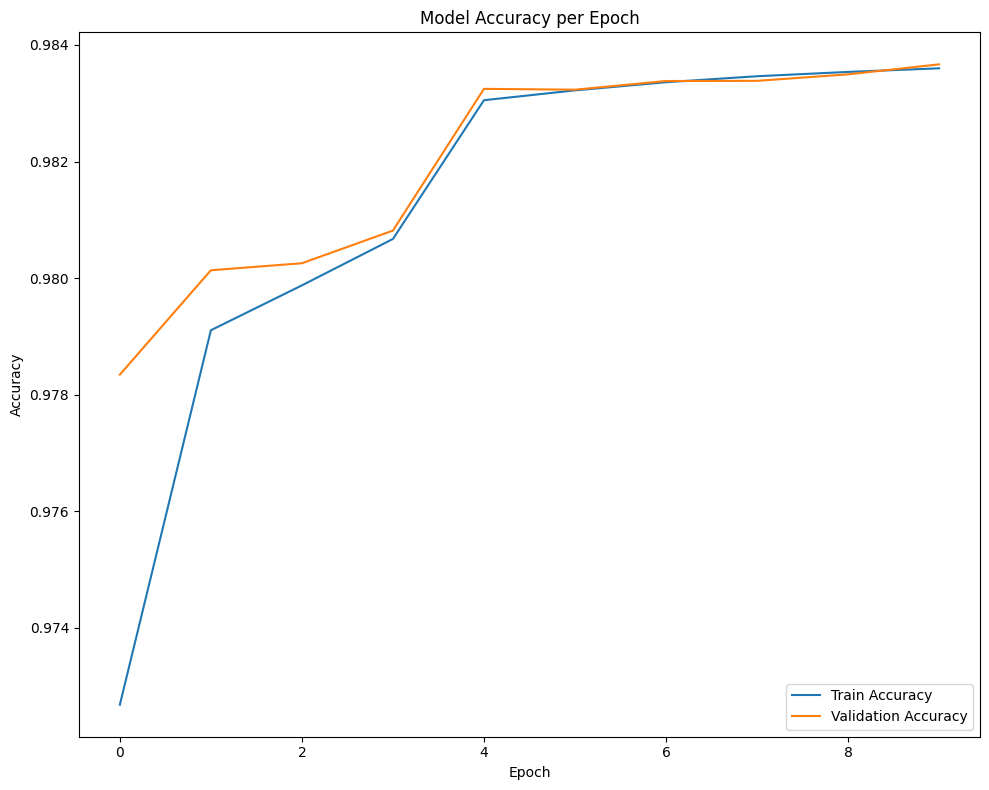


Figure 16: Neural network models loss per epoch.

We can see that the accuracy improved and the loss decreased rapidly until the fourth epoch. At that point, the model reduced the learning rate and we could see another improvement/reduction. After the 8th epoch, the model reduced the learning rate but could not improve the model further and thus stopped early at the 9th epoch.

### Optimization

We try to optimize the XGBClassifier even further by varying the learning rate, maximal depth, and the number of estimators, as we did with our regressor.

The best parameters were determined to be the same as the regressor, namely:

| learning rate: | 0.1 |
| --- | --- |
| maximal depth: | 7 |
| number of estimators: | 1000 |

We then trained a new XGBClassifier with these parameters, but could only see a small improvement in the accuracy, as our model was already very good.

Table 7: Accuracy of the unoptimized and optimized XGBClassifier model.

| **Model** | **Set** | **Metric** | **Score** |
| --- | --- | --- | --- |
| XGBClassifier | Train | Accuracy | 0.9858 |
| XGBClassifier | Test | Accuracy | 0.9856 |
| XGBClassifier | Train | F1 | 0.9858 |
| XGBClassifier | Test | F1 | 0.9856 |
| XGBClassifier optimized | Train | Accuracy | 0.9861 |
| XGBClassifier optimized | Test | Accuracy | 0.9858 |
| XGBClassifier optimized | Train | F1 | 0.9861 |
| XGBClassifier optimized | Test | F1 | 0.9858 |

We also studied the classification report to see if all the classes were equally well determined. As we can see in the following table, there is a slight imbalance, as the middle classes (2,3,4) could not be determined as well as the high and low polluters, so there could still be some improvement done here if the goal is to accurately determine each class. For us, this accuracy is enough as we are more interested in the highest polluters.

Table 8: Classification report for the XGBClassifier model.

| **class** | **precision** | **recall** | **f1-score** | **support** |
| --- | --- | --- | --- | --- |
| **0** | 0.99 | 0.99 | 0.99 | 121250 |
| **1** | 0.99 | 0.98 | 0.99 | 121425 |
| **2** | 0.98 | 0.99 | 0.98 | 51836 |
| **3** | 0.98 | 0.97 | 0.97 | 17719 |
| **4** | 0.99 | 0.97 | 0.98 | 15681 |
| **5** | 0.98 | 0.99 | 0.99 | 14304 |
| **6** | 1 | 0.99 | 0.99 | 7395 |
| **accuracy** |  |  | 0.99 | 349610 |
| **macro avg** | 0.98 | 0.98 | 0.98 | 349610 |
| **weighted avg** | 0.99 | 0.99 | 0.99 | 349610 |

To see if we can get better performance with an optimized neural network, we varied the batch size and the optimizer using a grid search.

| learning rate: | 0.01, 0.001, 0.0001 |
| --- | --- |
| optimizers: | SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam |
| batch size: | 512, 256, 128 |

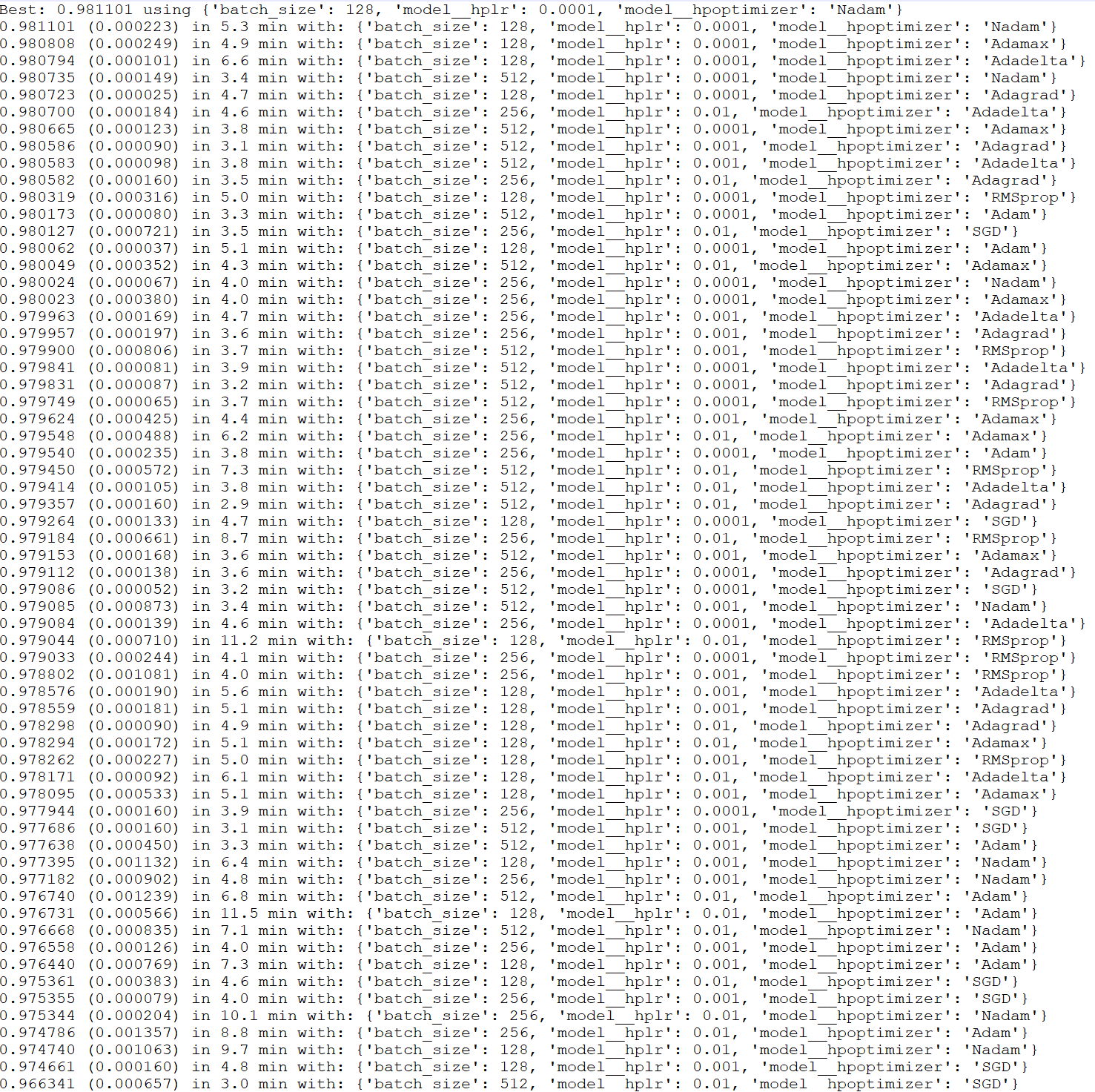


Figure 17: Optimization results of neural network classification.

Using the best parameters shown in Figure 17 we fitted a new neural network. The results are shown in Table 9.

Table 9: Accuracy of the unoptimized and optimized Neural Network Regression.

| **Model** | **Set** | **Metric** | **Score** |
| --- | --- | --- | --- |
| NN Classification | Train | Accuracy | 0.9832 |
| NN Classification | Test | Accuracy | 0.9830 |
| NN Classification | Train | F1 | 0.9831 |
| NN Classification | Test | F1 | 0.9830 |
| NN Classification optimized | Train | Accuracy | 0.9837 |
| NN Classification optimized | Test | Accuracy | 0.9834 |
| NN Classification optimized | Train | F1 | 0.9837 |
| NN Classification optimized | Test | F1 | 0.9834 |

If we compare the results of the optimized neural network classification model we can see that there was a slight improvement to the accuracy, but we could not improve the model to give us better results than XGBClassifier.

### Interpretation

In the feature importance plot, we can see that the fuel types having petrol in their name were important in determining the label of a car and therefore their pollution, as we would have expected. Also, fuel consumption seems to be an important feature, which makes sense, as the amount of fuel a car consumes is directly linked to the amount of emissions, even if the fuel type is not petrol.

Interestingly LPG was also considered important by the model for the calculation of the class, most probably as a detrimental feature for high polluters and a beneficial feature for lower polluters.

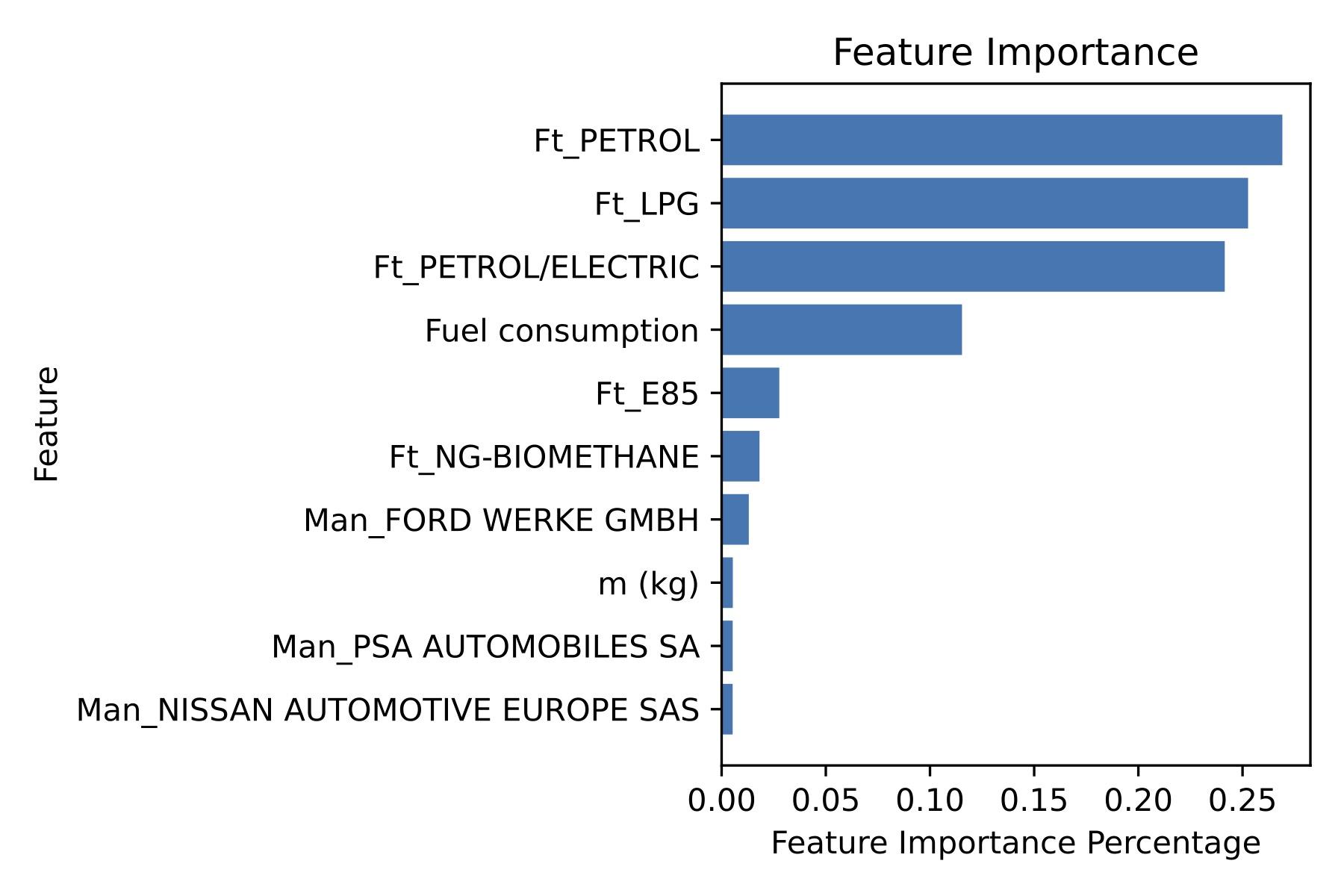


Figure 18: The 10 most important features of the optimized XGBClassifier.

Figure 19 shows the interpretation of a specific sample or instance of our dataset. We can see which feature contributes to the probability of the instance belonging to class 1, which corresponds to a car with the label B (120-140 g/km CO2 emissions). Here we see that again petrol and fuel consumption are the main contributors to determining the probability of the instance being class 1. On the other hand, The manufacturer Subaru and the fuel type biomethane contribute most to the probability that the class is not 1. Again we can see that surprisingly many manufacturers are important features for the model.

Additionally in this figure, we can see that it is difficult in some cases to determine the class correctly as some of the instances have a high probability to belong to two classes, here class 0 and class 1. This might be the case if a car has an emission value close to the border of two classes.

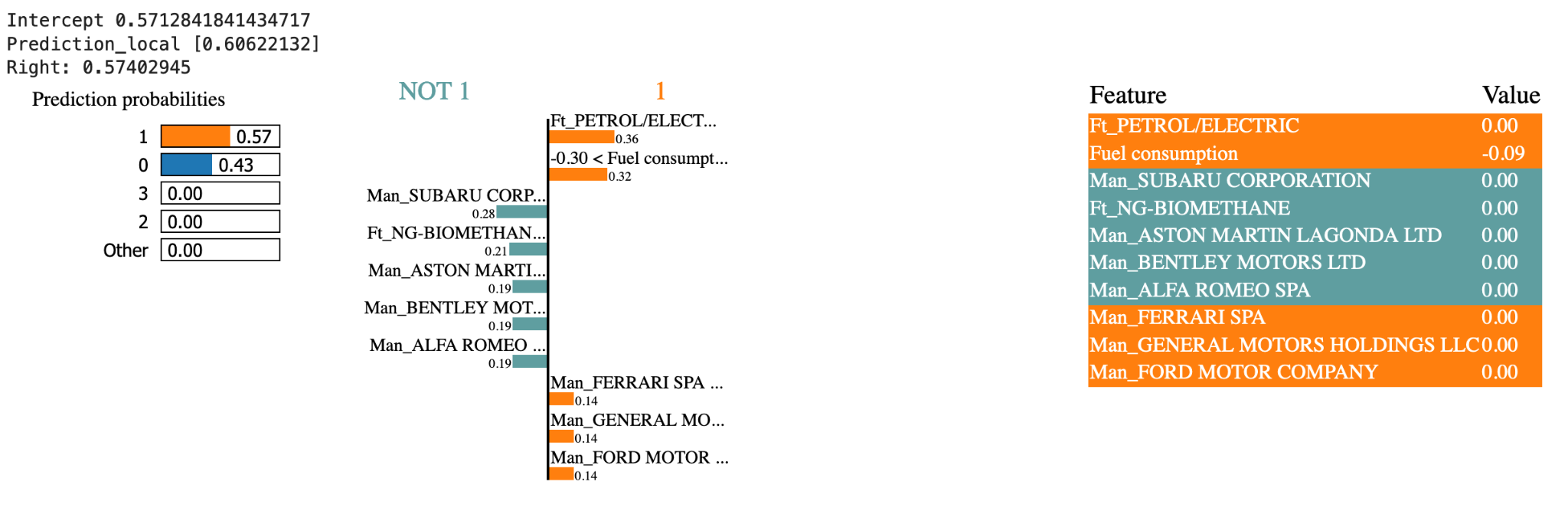
Figure 19: A random instance explanation using the LIME library.

Figure 20 shows the local interpretation of neural network regression with SHAP for a sample where the class was determined to be class 6, which is equivalent to a vehicle with the emission label G.

We can observe that the fuel consumption and the fuel types are the main contributors to the prediction of the CO2 emissions, while for this sample the fuel consumption has a negative impact, meaning that it indicates that this sample might not belong to class 6. The fuel types and manufacturers on the other hand indicate that the sample might belong to class 6.

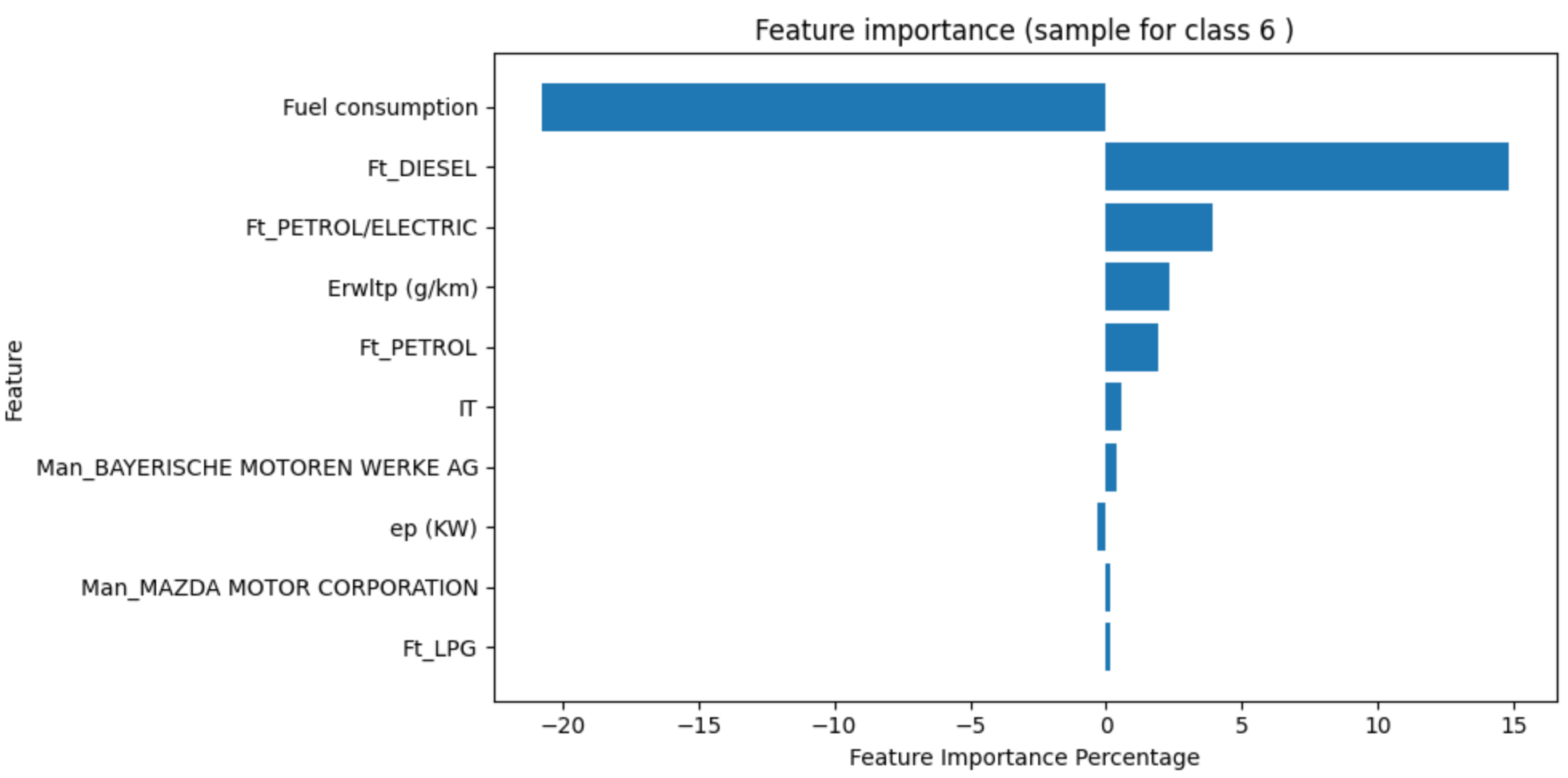


Figure 20: The 10 most important features of the Neural Network Regression.