08/09/2024

Project CO2 emission by vehicles

Report 2

Lenovo
DATA SCIENTEST

Table des matières

Introduction

The following activities have been conducted:

1. Classification of the problem
2. Reprocessing of the data
3. Implementation of several machine learning models
4. Implementation of deep learning model
5. Interpretation of the results and recommendations

This report aims to consolidate the various activities conducted during the modeling phase. The differing points of view among team members contribute to a richer, and more diverse approach

1 Classification of the problem

The goal of the project is to identify the vehicles that emit the most CO2 based on the technical characteristics that play a role in pollution. This problem can be solved using two different types of machine learning algorithms:

- **Regression**: The absolute value of the CO2 emission can be predicted using the technical features of the vehicle
- **Classification**: The CO2 emission category can be predicted. The car labels (Haq and Weiss 2016.) used in some European countries (e.g., Belgium, Germany, Spain, UK) can be used for our modeling. These models are presented hereafter:
    - A: CO2 emission < 100 g/km
    - B: CO2 emission between 100 g/km and 130 g/km
    - C: CO2 emission between 130 g/km and 160 g/km
    - D: CO2 emission between 160 g/km and 190 g/km
    - E: CO2 emission between 190 g/km and 220 g/km
    - F: CO2 emission between 220 g/km and 250 g/km
    - G: CO2 emission greater 250 g/km

The main performance metric differs based on the category of the problem. For regression, we used the **RMSE** as the main performance metric. RMSE is particularly suitable because (i) it gives higher weight to large errors, which is important when predicting CO2 emissions as large prediction errors could lead to significant misclassification of vehicle pollution levels (ii) it's in the same unit as the target variable (g/km), making it easily interpretable. For the classification problem, we use **F1-score** because
it provides a single score that encapsulates both false positives and false negatives, giving a more comprehensive view of the model's performance across all classes.

We also use some other quantitative metrics. For regression, we use **R-square**. For classification, we use **Confusion Matrix** as well as **Precision and Recall for each class**. For classification, these metrics are particularly important for understanding the model's performance on **individual CO2 emission classes**.

We added another qualitative metric to help select the best model **prediction speed** measured through the time taken to make predictions.

In this report, the numbering **of the classes will go from 0 to 6**. 0 will correspond to label A, and 6 will correspond to label G.

2 First modeling using the data from the first preprocessing

We started the modeling with the output data of the post-processing phase. The post-processing highlighted **4 main features** for the model:

- Fuel consumption
- Engine capacity
- Emissions reduction through Innovative Technology
- WLTP (Test) mass

During the first week, we run three regressions models (Linear Regression, Ridge Regression, and Lasso Regression) and two classification models (Logistic Regression and Decision Tree) to evaluate if we will need to **work again on the preprocessing** before moving to the modeling. We chose Linear Regression, Ridge Regression, and Lasso Regression because they are the most common ones. We chose Logistic Regression and Decision Tree for classification because they are fast to implement and don't need important computational resources. The objective once again here is to have a quick sense if we would need to modify the algorithm.

Since the fuel consumption was identified to be highly correlated with CO2 emissions, we decided to build two models: one with fuel consumption and another without fuel consumption. The best model should be able to have good performance with and without fuel consumption. Fuel consumption is not an intrinsic design feature but a consequence of design choices (e.g., emissions reduction through Innovative Technology, engine capacity, WLTP mass).

2.1 Regression models

2.1.1 Linear Regression

Table 1: Linear Regression for Dataset from 2022

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.92	0.25
R^2 test data	0.92	0.25
RMSE train data	11.15	34.41
RMSE test data	11.15	34.41

The prediction time is 53 ms for the model with fuel consumption and 30 ms for the model without fuel consumption.

Table 2: Linear Regression for Dataset from France 2022

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.985	0.343
R^2 test data	0.985	0.343
RMSE train data	1.946	2.23
RMSE test data	1.946	2.23

The prediction time is 0.026 ms for the model with fuel consumption and 0.014 ms for the model without fuel consumption.

Table 3: Linear Regression for Dataset from Germany 2022

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.996	0.312
R^2 test data	0.996	0.312
RMSE train data	1.864	2.08
RMSE test data	1.864	2.08

The prediction time is 0.017 ms for the model with fuel consumption and 0.008 ms for the model without fuel consumption.

2.1.2 Ridge Regression

Table 4: Ridge Regression for Dataset from 2022 – Ridge Regression with alpha=10

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.92	0.25
R^2 test data	0.92	0.25
RMSE train data	11.15	34.41
RMSE test data	11.15	34.41

The prediction time is 34 ms for the model with fuel consumption and 31 ms for the model without fuel consumption. Several values of alpha in the Ridge Regression between 0 and 10 are giving almost the same performance metrics

Table 5: Ridge Regression for Dataset from France 2022– Ridge Regression with alpha=10

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.985	0.343
R^2 test data	0.985	0.343
RMSE train data	1.946	2.23
RMSE test data	1.946	2.23

The prediction time is 0.021 ms for the model with fuel consumption and 0.012 ms for the model without fuel consumption. Several values of alpha in the Ridge Regression between 0 and 10 are giving almost the same performance metrics

Table 6: Ridge Regression for Dataset from Germany 2022– Ridge Regression with alpha=10

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.996	0.312
R^2 test data	0.996	0.312
RMSE train data	1.864	2.08
RMSE test data	1.864	2.08

The prediction time is 0.028 ms for the model with fuel consumption and 0.016 ms for the model without fuel consumption. Several values of alpha in the Ridge Regression between 0 and 10 are giving almost the same performance metrics.

2.1.3 Lasso Regression

Table 7: Lasso Regression for Dataset from 2022 – Lasso with alpha =35.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.13	0.0
R^2 test data	0.13	0.0
RMSE train data	37.06	39.74
RMSE test data	37.04	39.71

The prediction time is 34 ms for the model with fuel consumption and 30 ms for the model without fuel consumption. This value of alpha was chosen to demonstrate the impact of the GridSearch on the Lasso algorithm.

Table 8: Lasso Regression for Dataset from 2022 – Lasso with an optimized value for alpha 0.001.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.92	0.25
R^2 test data	0.92	0.25
RMSE train data	11.15	34.41
RMSE test data	11.15	34.41

The prediction time is 35 ms for the model with fuel consumption and 29 ms for the model without fuel consumption.

Table 9: Lasso Regression for Dataset from France 2022 – Lasso with an optimized value for alpha 0.001.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.964	0.33
R^2 test data	0.964	0.33
RMSE train data	0.375	2.31
RMSE test data	0.375	2.31

The prediction time is 0.0027 s for the model with fuel consumption and 0.0015 s for the model without fuel consumption.

Table 10: Lasso Regression for Dataset from Germany 2022 – Lasso with an optimized value for alpha 0.001.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.996	0.29
R^2 test data	0.996	0.29
RMSE train data	1.86	2.57
RMSE test data	1.86	2.57

The prediction time is 0.022 s for the model with fuel consumption and 0.017 s for the model without fuel consumption.

Key takeaways: First, all the 3 regression models, after using GridSearch for Lasso and Ridge, have the same performance. This can be explained by the fact that the relationship between the target variable and the features are strongly linear. This hypothesis is confirmed by the fact that the probability plot presented in Figure 1. In our case, the additional complexity of Ridge and Lasso did

not provide any benefit over simple linear regression. Second, the model is not performant when the fuel consumption is removed.

When we plotted the linear regression coefficient in the case of model without fuel consumption (Figure 1), we realized that the weight of the vehicle has a negative contribution to the CO2 emission. This does not make sense because more the vehicle is heavy more it will consume fuel and emit CO2.

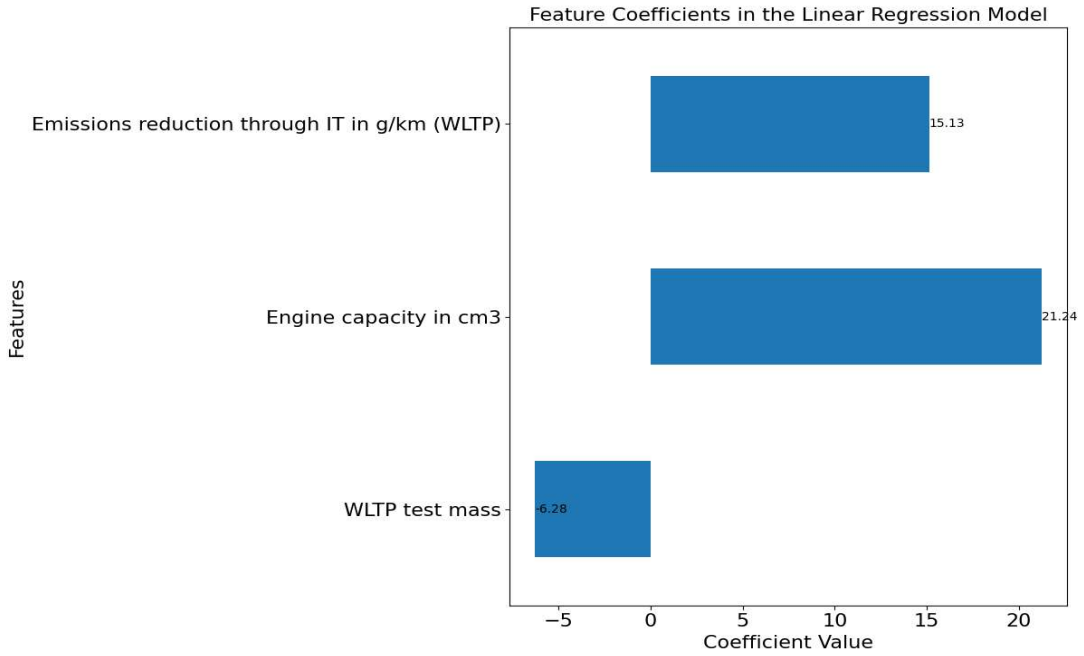


Figure 1: Features coefficients in the linear regression models

2.2 Classification models

2.2.1 Logistic regression

The confusion matrix and the classification report are presented below:

Table 11: Logistic Regression Confusion Matrix for Dataset from 2022.

		Predicted values													
		Model with fuel consumption							Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	124012	17534	706	180	21	10	1	104460	30472	22	112	6772	0	626
	1	3268	556675	79783	445	133	9	56	16526	513838	109459	404	79	0	63
	2	1381	95173	385234	6115	13	0	0	38877	187180	257593	4261	5	0	0
	3	88	1238	16399	55316	4028	2	2	15058	4440	47180	7997	2228	0	170
	4	9	0	236	6961	28336	1304	0	5404	1115	10569	5238	14508	0	12
	5	7	0	30	159	2735	12884	845	1872	67	1754	4750	7381	0	836
	6	3	0	1	9	46	1214	10061	535	2	1	1749	1120	0	7927

Table 12: Logistic Regression Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.96	0.87	0.91		0.57	0.73	0.64
1	0.83	0.87	0.85		0.7	0.8	0.75
2	0.8	0.79	0.79		0.6	0.53	0.56
3	0.8	0.72	0.76		0.33	0.1	0.16
4	0.8	0.77	0.79		0.45	0.39	0.42
5	0.84	0.77	0.8		0	0	0
6	0.92	0.89	0.9		0.82	0.7	0.76
Accuracy			0.83				0.64
Macro average	0.85	0.81	0.83		0.5	0.47	0.47

Weighted average	0.83	0.83	0.83		0.62	0.64	0.62

2.2.2 Decision Tree

The confusion matrix and the classification report are presented below:

Table 13: Decision Tree Confusion Matrix for Dataset from 2022.

		Predicted values													
		Model with fuel consumption							Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	124012	17534	706	180	21	10	1	139771	1899	168	0	0	0	626
	1	3268	556675	79783	445	133	9	56	31569	441445	167290	6	4	0	55
	2	1381	95173	385234	6115	13	0	0	44011	63607	379501	796	1	0	0
	3	88	1238	16399	55316	4028	2	2	18607	7550	40250	10279	290	0	97
	4	9	0	236	6961	28336	1304	0	10889	580	8613	7825	8938	0	1
	5	7	0	30	159	2735	12884	845	4886	62	5271	1322	4780	0	339
	6	3	0	1	9	46	1214	10061	908	47	1562	1935	1762	0	5120

Table 14: Decision Tree Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.92	0.99	0.95		0.56	0.98	0.71
1	0.95	0.89	0.92		0.86	0.69	0.76
2	0.88	0.91	0.89		0.63	0.78	0.7
3	0.7	0.94	0.8		0.46	0.13	0.21
4	0.48	0.46	0.47		0.57	0.24	0.34
5	0	0	0		0	0	0
6	0.99	0.87	0.93		0.82	0.45	0.58
Accuracy			0.83				0.7
Macro average	0.7	0.72	0.71		0.56	0.47	0.47
Weighted average	0.88	0.89	0.88		0.71	0.7	0.68

Key takeaways: The accuracy of the different classification models requires to rework on the data processing

3 Second modeling using a modified data set

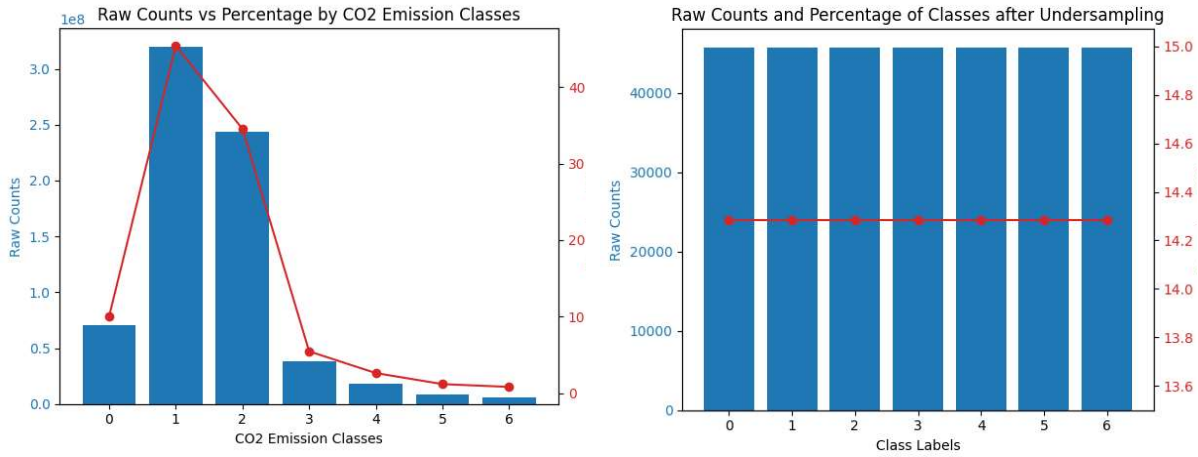
Based on the first modeling, we took two initiatives to improve the quality of the modeling (i) improve the quality of the data set (ii) ensure balanced classes.

- Improve the quality of the training data set

  First we added features that we added all the secondary features that first we do not think that will be useful for the modeling. Three new features related either to the nature of the fuel or the characteristics of the engine were added: electric energy consumption in Wh/km , engine power in KW, fuel type. We use the dummies function to preprocess the fuel type which is a categorical data

- Ensure that the classes are balanced

We use undersampling to ensure that we have the same number of elements in each classes. Before undersampling, more than 70% of the data was within class B and C. After undersampling, each of the 7 classes has around 14% of the total data (Fig. 2).



(a) Data distribution before under sampling

(b) Data distribution after under sampling

Figure 2: Data distribution across classes

After we reprocessed the data, we tried 3 regression models, and also 9 classification models.

We use several types of algorithms:

- **Regression:**
  - Linear regression
  - Ridge
  - Lasso
- **Classification:**
  - Logistic regression
  - SVM
  - KNN
  - Decision Tree
  - Decision Tree Boosting
  - Bagging
  - Random Forest
  - XgBoost
  - Deep Learning

The results of the different models are presented in the Table 1, and in Figure 2 (for the model with fuel consumption) and Figure 3 (without fuel consumption). The best models that will be retained will be the **ones that have the best metrics in terms of performance and complexity** in the case of the model without fuel consumption.

3.1 Regression

3.1.1 Linear Regression

Table 15: Linear Regression for Modified Dataset from 2022.

	Model with fuel consumption	**Model without fuel consumption**
R^2 train data	0.92	0.87
R^2 test data	0.92	0.87
RMSE train data	11.15	14.38
RMSE test data	11.15	14.38

Table 16: Linear Regression for Modified Dataset from France 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.986	0.891
R^2 test data	0.986	0.891
RMSE train data	1.94	8.32
RMSE test data	1.94	8.32

Table 17: Linear Regression for Modified Dataset from Germany 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.996	0.923
R^2 test data	0.996	0.923
RMSE train data	1.86	4.31
RMSE test data	1.86	4.31

Advantage: Linear regression is simple and easy to interpret. The prediction is fast ~73 ms, and the result is relatively good.

Constraint: In general linear regression can struggle with complex data patterns (e.g., multicollinearity)

Key takeaways: The model lost 0.05 point in R-square and had 30% higher when the fuel consumption was removed. However, the model without fuel consumption is still relatively good

3.1.2 Ridge

Table 18: Ridge for Modified Dataset from 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.92	0.87
R^2 test data	0.92	0.87
RMSE train data	11.15	14.38
RMSE test data	11.15	14.38

Table 19: Ridge for Modified Dataset from France 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.986	0.891
R^2 test data	0.986	0.891
RMSE train data	1.94	8.32
RMSE test data	1.94	8.32

Table 20: Ridge for Modified Dataset from Germany 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.996	0.923
R^2 test data	0.996	0.923
RMSE train data	1.86	4.31
RMSE test data	1.86	4.31

Advantage: Ridge regression addresses the multicollinearity through L2 regularization but can still be as fast as the linear regression**.**

Constraint: This model requires tuning of regularization parameter**.** It still assumes linearity.

Key takeaways: Because regularization did not significantly affect the performance, we can conclude that the collinearity is limited

3.1.3 Lasso Regression

Table 21: Lasso for Modified Dataset from 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.92	0.87
R^2 test data	0.92	0.87
RMSE train data	11.15	14.39
RMSE test data	11.15	14.39

Table 22: Lasso for Modified Dataset from France 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.986	0.891
R^2 test data	0.986	0.891
RMSE train data	1.94	8.32
RMSE test data	1.94	8.32

Table 23: Lasso for Modified Dataset from Germany 2022.

	Model with fuel consumption	Model without fuel consumption
R^2 train data	0.996	0.923
R^2 test data	0.996	0.923
RMSE train data	1.84	4.29
RMSE test data	1.84	4.29

For sake of clarity, a high level synthesis of all the regression models is presented below :

Table 24: Lasso for Modified Dataset from 2022.

	Model with Fuel Consumption	**Model without Fuel Consumption**	**Model with Fuel Consumption**	**Model without Fuel Consumption**
	Performance metrics	Prediction speed (ms)	Accuracy (RMSE for regression)	Prediction speed (ms)
Linear regression	R-square: 0.97 RMSE: 6.64	69	R-square: 0.87 RMSE: 14.38	70
Ridge regression	R-square: 0.97 RMSE: 6.64	72	R-square: 0.87 RMSE: 14.38	67
Lasso regression	R-square: 0.97 RMSE: 6.67	71	R-square: 0.87 RMSE: 14.39	67

Advantage: This model can perform feature selection through L1 regularization. This is not very helpful in our case since the number of features is limited and features were carefully chosen through an intense data preparation phase

Constraint: This model requires careful tuning to avoid the elimination of important features

Key takeaways: Lasso regression did improve the performance. This can be explained by the fact that we don't have high dimensional data.

3.2 Classification

3.2.1 Logistic regression

Characteristics of the model: Linear model with logistic regression. The parameter C, the inverse regularization strength is 0.1, small to avoid overfitting.

Table 25: Logistic regression Confusion Matrix for Dataset from 2022.

		Predicted values													
		Model with fuel consumption							**Model without fuel consumption**						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5530	84	1	8	0	0	0	4871	731	5	9	0	0	7
	1	135	4755	721	11	0	0	1	232	4005	1167	212	1	6	0
	2	8	458	4710	447	0	0	0	54	975	3824	720	45	5	0
	3	14	66	255	4737	544	4	3	5	63	635	3776	822	312	10
	4	2	0	4	316	4724	575	2	1	0	41	1382	2895	1276	28
	5	3	0	1	41	271	5176	131	3	0	0	114	1916	3071	519
	6	0	1	0	2	13	447	5160	0	0	1	5	152	1438	4027

Table 26: Logistic regression Classification Report for Dataset from 2022.

	Model with fuel consumption				**Model without fuel consumption**		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.97	0.98	0.98		0.94	0.87	0.9
1	0.89	0.85	0.87		0.69	0.71	0.7
2	0.83	0.84	0.83		0.67	0.68	0.68
3	0.85	0.84	0.85		0.61	0.67	0.64
4	0.85	0.84	0.85		0.5	0.51	0.51
5	0.83	0.92	0.88		0.5	0.55	0.52
6	0.97	0.92	0.95		0.88	0.72	0.79
Accuracy			0.88				0.67
Macro average	0.89	0.88	0.88		0.68	0.67	0.68
Weighted average	0.89	0.88	0.88		0.68	0.67	0.68

Advantage: It is a simple linear model with a limited risk of overfitting since C =in our case is very small

Constraint: The model requires an optimal selection of the parameter C

Key takeaways: The model without fuel consumption struggled to predict certain classes**,** and the overall accuracy dropped by 0.21 point

3.2.2 Logistic regression with Grid Search

Characteristics of the model: We explored several values for C (0.1, 1.0, 10), for the penalty (L1, L2), and the solver (saga). We chose the Saga solver because it fast and works well with large datasets. The optimization found that C=0.1, penalty='l1', and solver='saga' gave the best result.

Table 27: Logistic regression with Grid Search Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5573	41	1	8	0	0	0	5397	204	5	10	0	5	2
	1	109	4903	607	3	0	0	1	261	3967	1184	204	1	6	0
	2	6	295	4926	396	0	0	0	38	939	3886	711	44	5	0
	3	14	99	155	4869	478	5	3	5	63	627	3791	827	304	6
	4	2	0	6	169	4964	481	1	2	0	6	1393	2908	1283	31
	5	3	1	0	50	247	5236	86	3	0	0	170	1919	3053	478
	6	1	0	0	1	15	134	5472	0	0	1	5	152	1438	4027

Table 28: Logistic regression with Grid Search Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.98	0.99	0.98		0.95	0.96	0.95
1	0.92	0.87	0.89		0.77	0.71	0.73
2	0.86	0.88	0.87		0.68	0.69	0.69
3	0.89	0.87	0.88		0.6	0.67	0.64
4	0.87	0.88	0.88		0.5	0.52	0.51
5	0.89	0.93	0.91		0.5	0.54	0.52
6	0.98	0.97	0.98		0.89	0.72	0.79
Accuracy			0.91				0.69
Macro average	0.91	0.91	0.91		0.7	0.69	0.69
Weighted average	0.91	0.91	0.91		0.7	0.69	0.69

Advantage: In general, optimization helps find better parameters for the classic model

Constraint: Grid search is time consuming and increases computational complexity

Key takeaways: The optimization increased the accuracy by 0.02/0.03, which is limited when compared to the computational complexity

3.2.3 SVM

Characteristics of the model: LinearSVC() with default values. The penalty is 'l2' and the C=1.0

Table 29: SVM Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5463	153	0	0	0	7	0	4877	734	0	4	0	8	0
	1	124	5035	260	151	52	0	1	256	4537	582	130	113	3	2
	2	2	1825	3171	349	276	0	0	41	2157	2812	235	370	8	0
	3	14	45	2061	1571	1706	223	3	5	123	2213	1077	1595	443	167
	4	2	0	179	536	3116	1789	1	136	0	560	224	2964	1188	551
	5	3	0	1	91	2334	1907	1287	3	2	0	79	2162	1045	2332
	6	1	0	1	14	41	58	5508	0	0	2	0	415	208	4998

Table 30: SVM Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.97	0.97	0.97		0.92	0.87	0.89
1	0.71	0.9	0.79		0.6	0.81	0.69
2	0.56	0.56	0.56		0.46	0.5	0.48
3	0.58	0.28	0.38		0.62	0.19	0.29
4	0.41	0.55	0.47		0.39	0.53	0.45
5	0.48	0.34	0.4		0.36	0.19	0.25
6	0.81	0.98	0.89		0.62	0.89	0.73
Accuracy			0.88				0.57
Macro average	0.65	0.65	0.64		0.57	0.57	0.54
Weighted average	0.65	0.65	0.64		0.57	0.57	0.54

Advantage: SVM can be very effective in high-dimensional spaces

Constraint: It is memory intensive especially with large datasets. SVM can be very sensitive to the parameter choices

Key takeaways: SVM underperformed Logistic Regression which is supposed to be one of the basics classification algorithm**.**

3.2.4 SVM with Grid Search

Characteristics of the model: We tried different values of C (0.01, 0.1, 0.5), and also different values of the maximum iterations (2000, 5000). Changing the maximum iterations will allow the model to reach a better convergence state.

Table 31: SVM with Grid Search Confusion Matrix for Dataset from 2022.

		Predicted values														
		Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**		**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5449	167	0	0	0	7	0		4877	734	0	4	0	8	0
	1	124	5035	260	151	52	0	1		256	4537	582	130	113	3	2
	2	2	1825	3171	349	276	0	0		41	2157	2812	235	370	8	0
	3	14	45	2061	1571	1706	223	3		6	123	2213	1076	1595	443	167
	4	2	0	179	536	3116	1789	1		136	0	560	224	2964	1188	551
	5	3	0	1	91	2334	1907	1287		3	2	0	79	2162	1045	2332
	6	1	0	1	14	41	58	5508		0	0	2	0	415	208	4998

Table 32: SVM with Grid Search Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.97	0.97	0.97		0.92	0.87	0.89
1	0.71	0.9	0.79		0.6	0.81	0.69
2	0.56	0.56	0.56		0.46	0.5	0.48
3	0.58	0.28	0.38		0.62	0.19	0.29
4	0.41	0.55	0.47		0.39	0.53	0.45
5	0.48	0.34	0.4		0.36	0.19	0.25
6	0.81	0.98	0.89		0.62	0.89	0.73
Accuracy			0.65				0.57
Macro average	0.65	0.65	0.64		0.57	0.57	0.54
Weighted average	0.65	0.65	0.64		0.57	0.57	0.54

Advantage: Grid Search can improve the performance metrics.

Constraint: It can be time consuming.

Key takeaways: The fact that SVM with GridSearch underperformed Logistic Regression made us conclude that it was not adapted to the problem.

3.2.5 KNN

Characteristics of the model: number of neighbors=7, and 'minkowski' metric. Since we are in the presence of a large data set (~300 K), 7 is a relatively moderate number.

Table 33: KNN Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5621	1	0	1	0	0	0	5612	10	0	1	0	0	0
	1	8	5510	95	8	1	0	1	9	5312	286	14	0	2	0
	2	0	115	5414	92	2	0	0	1	293	5130	196	3	0	0
	3	13	11	47	5460	91	1	0	3	9	179	5222	201	9	0
	4	2	2	2	52	5481	81	3	1	0	2	262	5108	240	10
	5	2	1	1	3	86	5505	25	3	0	0	27	485	5025	83
	6	1	0	1	5	5	25	5586	1	0	2	4	4	224	5388

Table 34: KNN Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	1	1	1		1	1	1
1	0.98	0.98	0.98		0.94	0.94	0.94
2	0.97	0.96	0.97		0.92	0.91	0.91
3	0.97	0.97	0.97		0.91	0.93	0.92
4	0.97	0.97	0.97		0.88	0.91	0.89
5	0.98	0.98	0.98		0.91	0.89	0.9
6	0.99	0.99	0.99		0.98	0.96	0.97
Accuracy			0.98				0.93
Macro average	0.98	0.98	0.98		0.94	0.93	0.93
Weighted average	0.98	0.98	0.98		0.94	0.93	0.93

Advantage: This algorithm is simple and very intuitive with a limited number of parameters to tune

Constraint: Can be computationally intensive with large data sets. In our case the prediction time is 17300 ms v. 4 ms for Logistic Regression (x4300)

Key takeaways: With a moderate value of k, we were able to have a good performance: 0.93 even without fuel consumption.

3.2.6 Decision Tree

Characteristics of the model: criterion='entropy', max_depth=7. The entropy criteria. We chose a depth of 7 to have a good balance between underfitting and overfitting.

Table 35: Decision Tree Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5590	24	1	2	0	0	6	5582	20	5	0	0	5	11
	1	17	5453	147	3	2	0	1	257	3533	1752	54	25	2	0
	2	4	202	5333	80	4	0	0	1	278	4394	750	200	0	0
	3	14	137	94	5259	95	24	0	3	0	456	3699	1352	113	0
	4	2	0	28	62	5318	213	0	1	0	23	736	3884	963	16
	5	3	1	12	64	92	5443	8	3	0	0	157	629	4267	567
	6	0	0	0	4	59	99	5461	0	0	1	11	394	324	4893

Table 36: Decision Tree Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.99	0.99	0.99		0.95	0.99	0.97
1	0.94	0.97	0.95		0.92	0.63	0.75
2	0.95	0.95	0.95		0.66	0.78	0.72
3	0.96	0.94	0.95		0.68	0.66	0.67
4	0.95	0.95	0.95		0.6	0.69	0.64
5	0.94	0.97	0.95		0.75	0.76	0.76
6	1	0.97	0.98		0.89	0.87	0.88
Accuracy			0.96				0.77

Macro average	0.96	0.96	0.96		0.78	0.77	0.77
Weighted average	0.96	0.96	0.96		0.78	0.77	0.77

Advantage: This algorithm is easy to interpret and visualize and can also handle both numerical and categorical data.

Constraint: Decision Tree to overfitting and the max depth needs careful tuning**.**

Key takeaways: Compared to KNN, Decision has a higher difference between (0.23 vs. 0.05) between the model with and without fuel consumption**.**

3.2.7 Decision Tree Boosting

Characteristics of the model: The estimator is the previous decision tree with a maximum depth of 7. The n_estimators is 100, this is the number of weak learners to train iteratively. We chose 100 which is a moderate number to limit the computation time. It's possible to increase this number up to 500 in the case the model needs significant improvement.

Table 37: Decision Tree Boosting Confusion Matrix for Dataset from 2022.

		Predicted values													
		Model with fuel consumption								Model without fuel consumption					
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5598	20	0	2	0	0	3	5515	100	0	2	1	0	5
	1	22	5373	223	4	0	0	1	21	4800	793	7	0	2	0
	2	0	196	5397	30	0	0	0	1	1031	4258	333	0	0	0
	3	13	28	198	5304	80	0	0	3	7	1062	4021	483	43	4
	4	2	0	0	74	5389	158	0	1	1	2	587	4579	448	5
	5	2	1	0	1	131	5338	150	3	0	0	20	783	4536	281
	6	0	0	0	5	16	27	5575	0	0	2	0	13	371	5237

Table 38: Decision Tree Boosting Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.99	1	0.99		0.99	0.98	0.99
1	0.96	0.96	0.96		0.81	0.85	0.83
2	0.93	0.96	0.94		0.7	0.76	0.73
3	0.98	0.94	0.96		0.81	0.72	0.76
4	0.96	0.96	0.96		0.78	0.81	0.8
5	0.97	0.95	0.96		0.84	0.81	0.82
6	0.97	0.99	0.98		0.95	0.93	0.94
Accuracy			0.96				0.84
Macro average	0.96	0.96	0.96		0.84	0.84	0.84
Weighted average	0.96	0.96	0.96		0.84	0.84	0.84

Advantage: This algorithm by combining multiple weak learners improve accuracy over single tree

Constraint: It requires more parameters tuning

Key takeaways: Despite the boosting, the gap between the model with and without fuel consumption is 0.12 higher than the gap for KNN

3.2.8 Bagging

Characteristics of the model: n_estimators is 200. 200 is on the higher end of commonly used values which can provide a good performance.

Table 39: Bagging Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5614	2	2	2	0	0	3	5612	7	0	4	0	0	0
	1	1	5573	46	2	0	0	1	5	5336	270	9	1	2	0
	2	1	42	5537	42	1	0	0	1	200	5234	187	0	1	0
	3	13	6	20	5531	51	2	0	2	6	167	5233	201	14	0
	4	1	0	1	37	5529	55	0	1	1	1	188	5114	311	7
	5	2	1	0	2	59	5542	17	3	0	0	14	244	5213	149
	6	0	1	0	5	3	16	5598	0	1	2	4	5	151	5460

Table 40: Bagging Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	1	1	1		1	1	1
1	0.99	0.99	0.99		0.96	0.95	0.96
2	0.99	0.98	0.99		0.92	0.93	0.93
3	0.98	0.98	0.98		0.93	0.93	0.93
4	0.98	0.98	0.98		0.92	0.91	0.91
5	0.99	0.99	0.99		0.92	0.93	0.92
6	1	1	1		0.97	0.97	0.97
Accuracy			0.99				0.95
Macro average	0.99	0.99	0.99		0.95	0.95	0.95
Weighted average	0.99	0.99	0.99		0.95	0.95	0.95

Advantage: This algorithm reduces variance and helps prevent overfitting

Constraint: It computationally intensive due to multiple models and requires significant memory and processing power

Key takeaways: It was one the best models because it reached an accuracy of 0.95 in the case without fuel consumption

3.2.9 Random Forest

Characteristics of the model: We used the default number of estimators which 100. These parameters give good results with very good prediction time:

Table 41: Random Forest Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5617	1	1	1	0	0	3	5615	7	0	1	0	0	0
	1	2	5577	41	2	0	0	1	5	5345	261	9	1	2	0
	2	0	41	5538	43	1	0	0	1	195	5241	186	0	0	0
	3	13	5	22	5535	47	1	0	3	4	161	5253	191	11	0
	4	1	0	1	31	5535	55	0	1	1	1	183	5125	305	7
	5	2	1	0	1	60	5542	17	3	0	0	16	252	5209	143
	6	0	0	1	4	2	15	5601	0	0	2	4	6	147	5464

Table 42: Random Forest Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	1	1	1		1	1	1
1	0.99	0.99	0.99		0.96	0.95	0.96
2	0.99	0.98	0.99		0.92	0.93	0.93
3	0.99	0.98	0.98		0.93	0.93	0.93
4	0.98	0.98	0.98		0.92	0.91	0.92
5	0.99	0.99	0.99		0.92	0.93	0.92

6	1	1	1	0.97	0.97	0.97
Accuracy			0.99			0.95
Macro average	0.99	0.99	0.99	0.95	0.95	0.95
Weighted average	0.99	0.99	0.99	0.95	0.95	0.95

Advantage: It is robust to overfitting due to averaging

Constraint: It is less interpretable than individual decision tress

Key takeaways: This model performs well even without extensive hyperparameter tuning. The accuracy is 0.95 even without fuel classes. All classes have F1-score about 0.9

3.2.10 XgBoost

Characteristics of the model: n_estimators=100, learning_rate=0.1. Those numbers are typically the common starting point for XgBoost algorithm

Table 43: XgBoost Confusion Matrix for Dataset from 2022.

	Predicted values														
	Model with fuel consumption								Model without fuel consumption						
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5616	2	0	2	0	0	3	5614	9	0	0	0	0	0
	1	14	5466	139	2	1	0	1	41	4968	600	11	1	1	1
	2	0	104	5451	67	1	0	0	1	335	4972	315	0	0	0
	3	13	18	52	5465	74	1	0	3	1	260	5010	325	24	0
	4	2	0	3	22	5528	67	1	1	0	3	243	4986	390	0
	5	2	1	0	10	87	5513	10	3	0	0	10	348	5123	139
	6	0	0	0	4	4	33	5582	0	0	2	3	8	219	5391

Table 44: XgBoost Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.99	1	1		0.99	1	0.99
1	0.98	0.97	0.97		0.94	0.88	0.91
2	0.97	0.97	0.97		0.85	0.88	0.87
3	0.98	0.97	0.98		0.9	0.89	0.89
4	0.97	0.98	0.98		0.88	0.89	0.88
5	0.98	0.98	0.98		0.89	0.91	0.9
6	1	0.99	1		0.97	0.96	0.97
Accuracy			0.98				0.92
Macro average	0.98	0.98	0.98		0.92	0.92	0.92
Weighted average	0.98	0.98	0.98		0.92	0.92	0.92

Advantage: This algorithm is one of the best on Kaggle. It is capable of capturing complex patterns

Constraint: The hyperparameter tuning can be sometimes complex

Key takeaways: The default parameters provided strong results. Further tuning could improve performance

3.2.11 Deep Learning

Characteristics of the model: the best architecture found is 64-32-7, with epochs=50 and batch_size of 32. We used the 'relu' as activation function for the first two layers, and 'softmax' for the last layer. For the model we used categorical cross entropy, Adam optimizer, and accuracy as metric. The model gave very good results. An analysis of the loss function and the accuracy showed that overfitting is limited (Fig. 3).

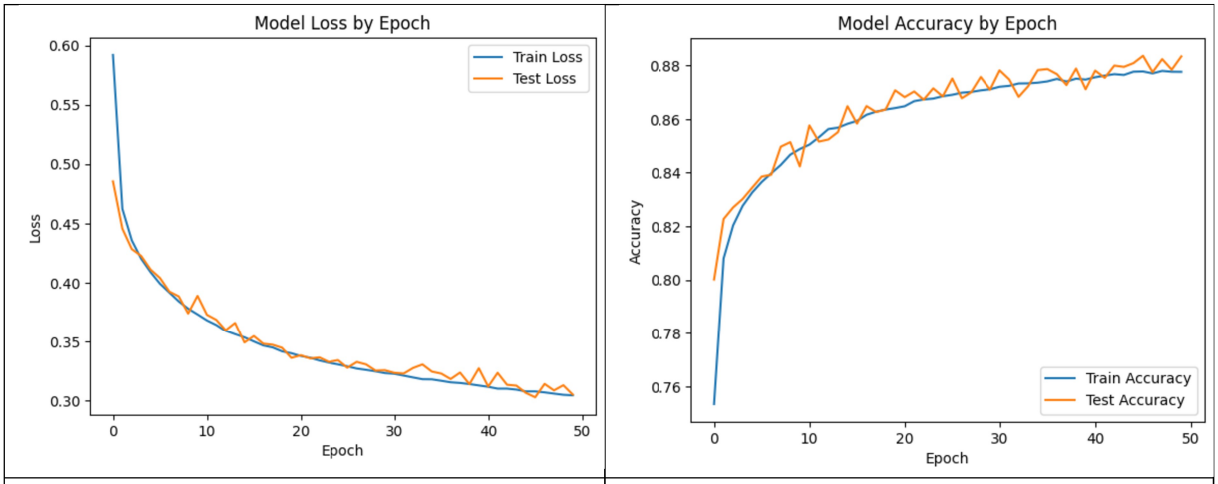
By curiosity we tried also another architecture 64-64-32-7 and realized that the validation loss was diverging (Fig. 4). Further analyses are needed to see how the dee learning model could be improved.

Table 45: Deep Learning Confusion Matrix for Dataset from 2022.

		Predicted values													
		Model with fuel consumption								Model without fuel consumption					
Real values	**Class**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0	5614	4	0	2	0	0	3	5613	10	0	0	0	0	0
	1	58	5422	133	8	1	0	1	79	4584	897	51	10	0	2
	2	0	258	5259	105	1	0	0	7	497	4732	364	23	0	0
	3	13	26	75	5421	83	5	0	5	19	557	4597	385	60	0
	4	3	3	14	28	5478	93	4	1	1	19	287	4985	309	21
	5	3	1	5	5	184	5394	31	3	0	3	12	647	4531	427
	6	0	0	0	3	13	36	5571	0	0	2	3	10	130	5478

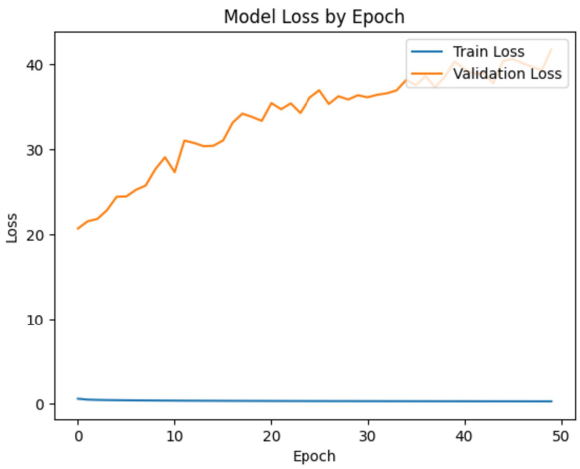
Table 46: Deep Learning Classification Report for Dataset from 2022.

	Model with fuel consumption				Model without fuel consumption		
Class	Precision	Recall	F1- Score		Precision	Recall	F1- Score
0	0.99	1	0.99		0.98	1	0.99
1	0.95	0.96	0.96		0.9	0.82	0.85
2	0.96	0.94	0.95		0.76	0.84	0.8
3	0.97	0.96	0.97		0.87	0.82	0.84
4	0.95	0.97	0.96		0.82	0.89	0.85
5	0.98	0.96	0.97		0.9	0.81	0.85
6	0.99	0.99	0.99		0.92	0.97	0.95
Accuracy			0.97				0.88
Macro average	0.97	0.97	0.97		0.88	0.88	0.88
Weighted average	0.97	0.97	0.97		0.88	0.88	0.88

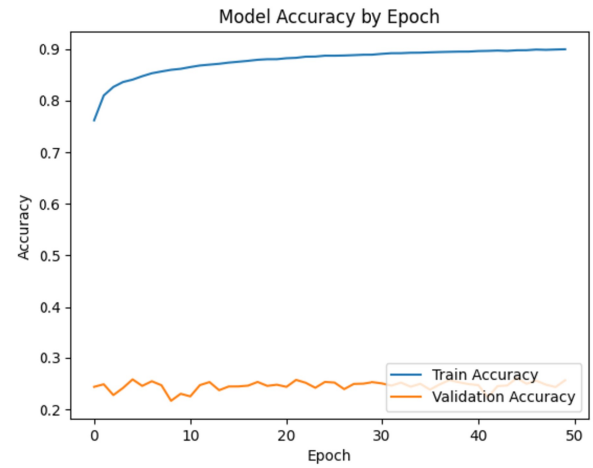


(a) Loss function (b) Accuracy

Figure 3: Loss and accuracy by epoch for for the 64-32-7 architecture



(a) Loss function



(b) Accuracy

Figure 4: Loss and accuracy by epoch for for the 64-64-32-7 architecture

Table 47: Comparison of various classification models Dataset from 2022.

	Model with Fuel Consumption	**Model without Fuel Consumption**	**Model with Fuel Consumption**	**Model without Fuel Consumption**
	Performance metrics	Prediction speed (ms)	Accuracy (RMSE for regression)	Prediction speed (ms)
Logistic regression	Average F1-score: 0.88	4	Average F1-score: 0.68	6
Logistic regression with Grid Search	Average F1-score: 0.91	8	Average F1-score: 0.69	5
SVM	Average F1-score: 0.64	4	Average F1-score: 0.54	4
SVM with Grid Search optimization	Average F1-score: 0.64	4	Average F1-score: 0.54	5
KNN	Average F1-score: 0.98	16 705	Average F1-score: 0.93	17 300
Decision Tree	Average F1-score: 0.96	4	Average F1-score: 0.77	4
Decision Tree Boosting	Average F1-score: 0.96	647	Average F1-score: 0.83	651
Bagging	Average F1-score: 0.99	1079	Average F1-score: 0.95	1434
Random Forest	Average F1-score: 0.99	89	Average F1-score: 0.95	99
XgBoost	Average F1-score: 0.98	70	Average F1-score: 0.91	77
Deep Learning	Average F1-score: 0.97	337 730	Average F1-score:	331 154

This table presents a comparison of various classification models applied to a dataset. The models are evaluated based on their average F1-score and prediction speed. The table includes models with and without fuel consumption as a predictor variable, as well as models optimized using grid search.

Key Findings

1. **Importance of Fuel Consumption:** Models incorporating fuel consumption consistently outperform those without, demonstrating its significance as a predictor.
2. **Ensemble Methods Dominate:** Ensemble methods like Decision Tree Boosting, Bagging, Random Forest, and XGBoost generally achieve the highest F1-scores, indicating their effectiveness in handling complex relationships within the data.
3. **Grid Search Optimization:** Grid search optimization often leads to improved performance, especially for models like Logistic Regression and SVM.
4. **KNN's High Accuracy:** KNN, despite its computational cost, achieves very high F1-scores, suggesting its suitability for this task.
5. **Deep Learning's Potential:** While Deep Learning models show promise, their computational overhead might limit their practicality in certain applications.

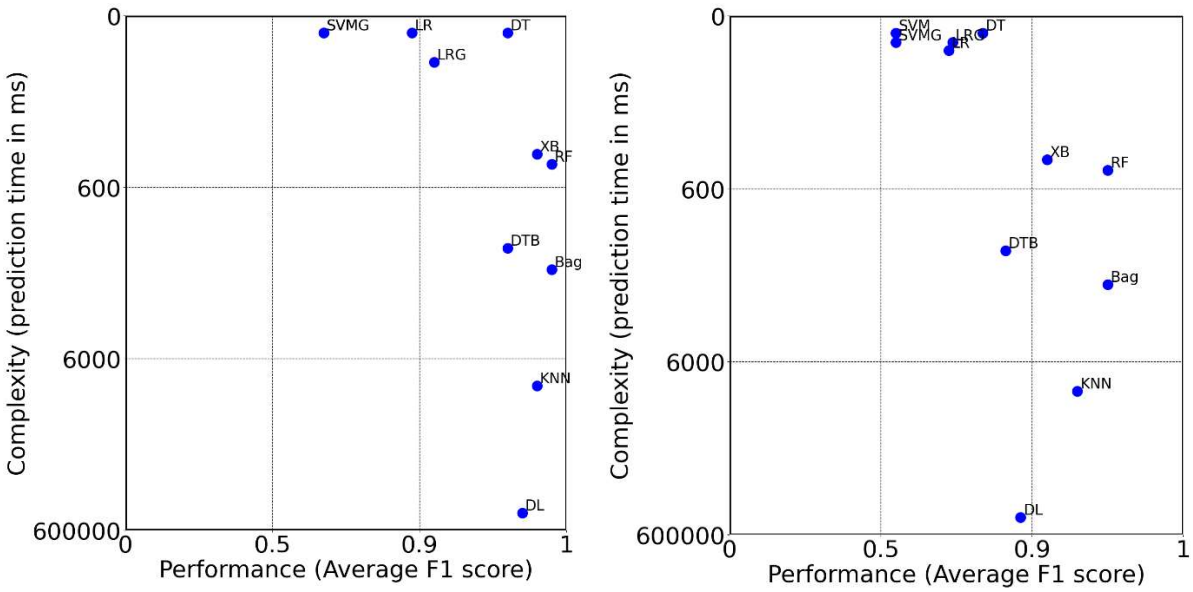
Grid Search Optimization

Grid Search is a hyperparameter tuning technique that explores different combinations of hyperparameters to find the optimal configuration for a model. By systematically testing various parameter values, Grid Search helps improve model performance.

Constraints and Advantages of Each Model

- **Logistic Regression:** Interpretable, efficient, but might struggle with complex non-linear relationships.
- **SVM:** Robust to outliers, effective for high-dimensional data, but can be computationally expensive for large datasets.
- **KNN:** Simple, non-parametric, but can be computationally expensive for large datasets and sensitive to the choice of distance metric.
- **Decision Trees:** Interpretable, handle non-linear relationships well, but can be prone to overfitting.
- **Ensemble Methods:** Combine multiple models to reduce overfitting and improve generalization, but can be computationally expensive.
- **Deep Learning:** Can learn complex patterns, but require significant computational resources and might be less interpretable.

Based on the analysis, ensemble methods like Decision Tree Boosting, Bagging, Random Forest, and XGBoost appear to be strong candidates for CO2 emission prediction in this context.



(a) Model with fuel consumption (b) Model without fuel consumption

Figure 5: Complexity vs. performance of all the classification models

Based on the results, Random Forest, XgBoost, Bagging, and Deep Learning appear to be the best models so far based on (i) the complexity which is measured through the prediction time and (ii) the performance score which is measured through F1 score for classification and R-square for regression.

3.3 Interpretation of the results

To understand the importance of each feature, we conducted an interpretability analysis using 3 tools: (i) the feature coefficients in the case of regression (ii) PCA, and (iii) SHAP

Interpretability of the model without fuel consumption:

The different analyses (Figures 6 to 8) identified 4 parameters as the most critical to CO2 emission:

- Factors that increase CO2 emission:
  - Engine power in KW
  - Mass of the vehicle
- Factors that decrease CO2 emission
  - Petrol/electric as fuel type
  - Electric energy consumption in KW

To meet the CO2 emission target, car manufacturers can follow two different strategies

Strategy 1: Car manufacturers can design light vehicles thus reducing the weight and the engine power

Strategy 2: They can do R&D to have better hybrid vehicles, vehicles with high electric engine power

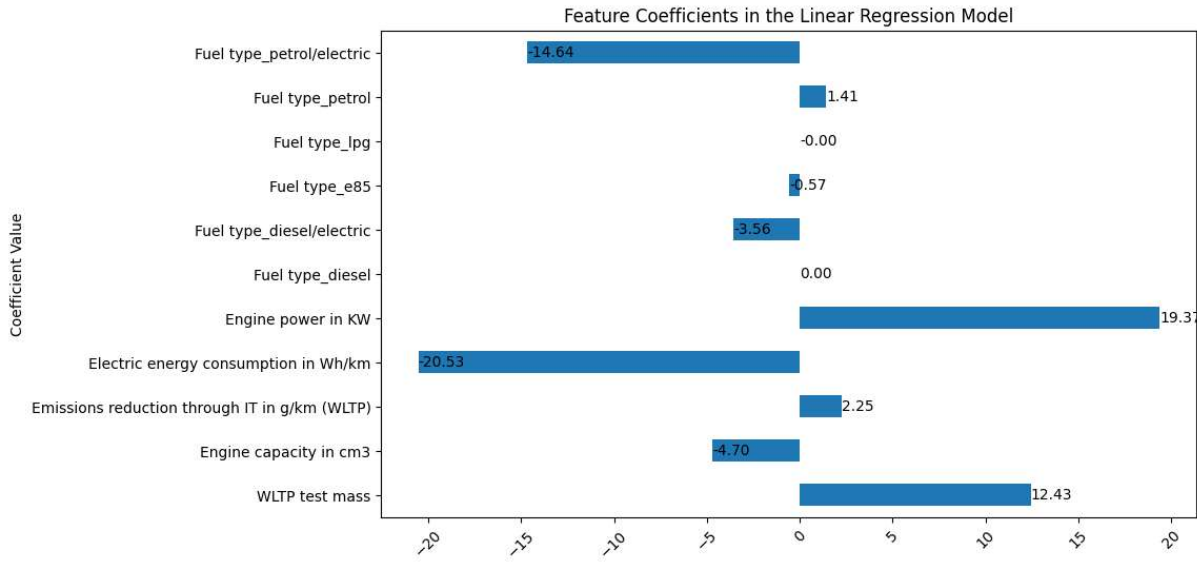


Figure 6: Model without Fuel consumption - Features coefficients in the Linear Regression model

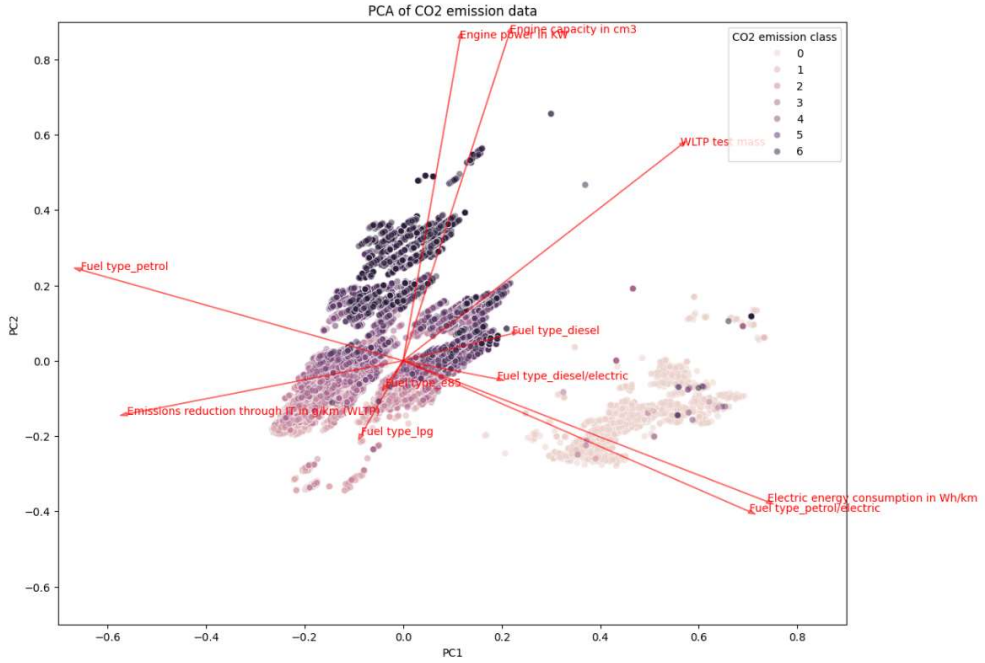


Figure 7: Model without Fuel consumption - PCA with the Random Forest model

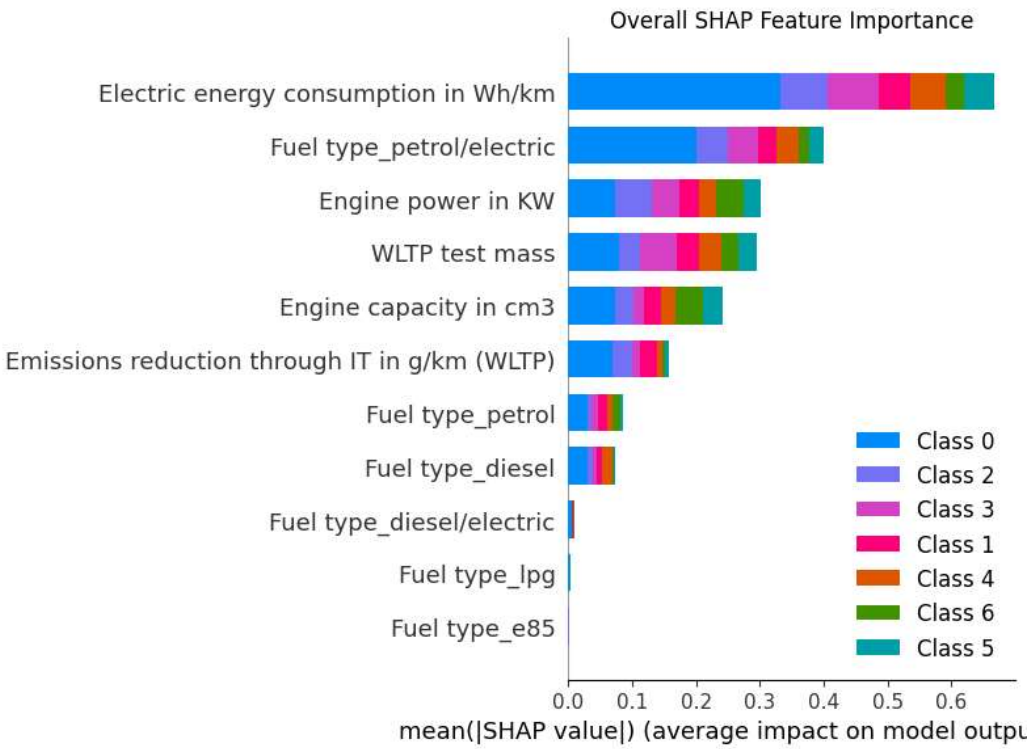


Figure 8: Model without Fuel consumption - SHAP analysis on the Random Forest model (best model)

Interpretability of the model with fuel consumption:

In the case of the model with the fuel consumption, the analyses conducted on figures 9 to 11 show a high impact of the fuel impact on the model. Since fuel consumption is a design output, it is definitely better to not keep in the model.

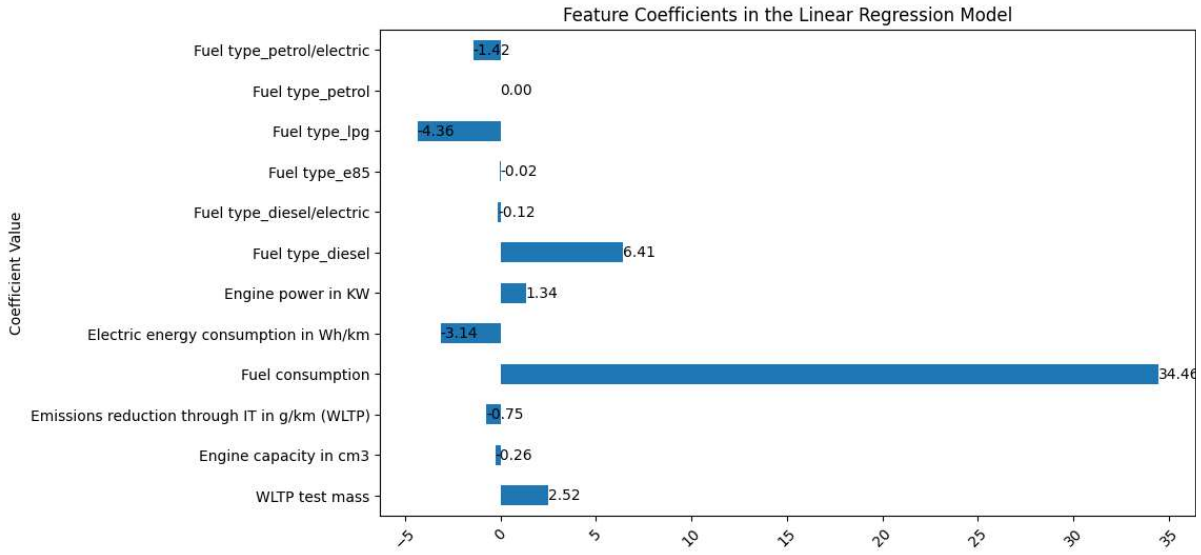


Figure 9: Model with Fuel consumption - Features coefficients in the Linear Regression model

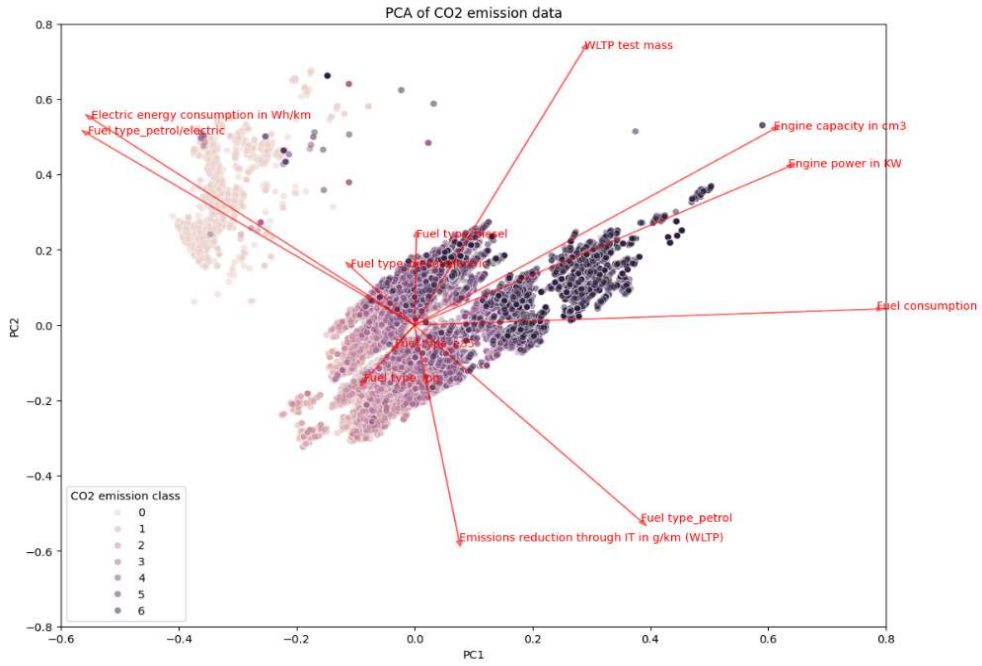


Figure 10: Model with Fuel consumption - PCA with the RF model

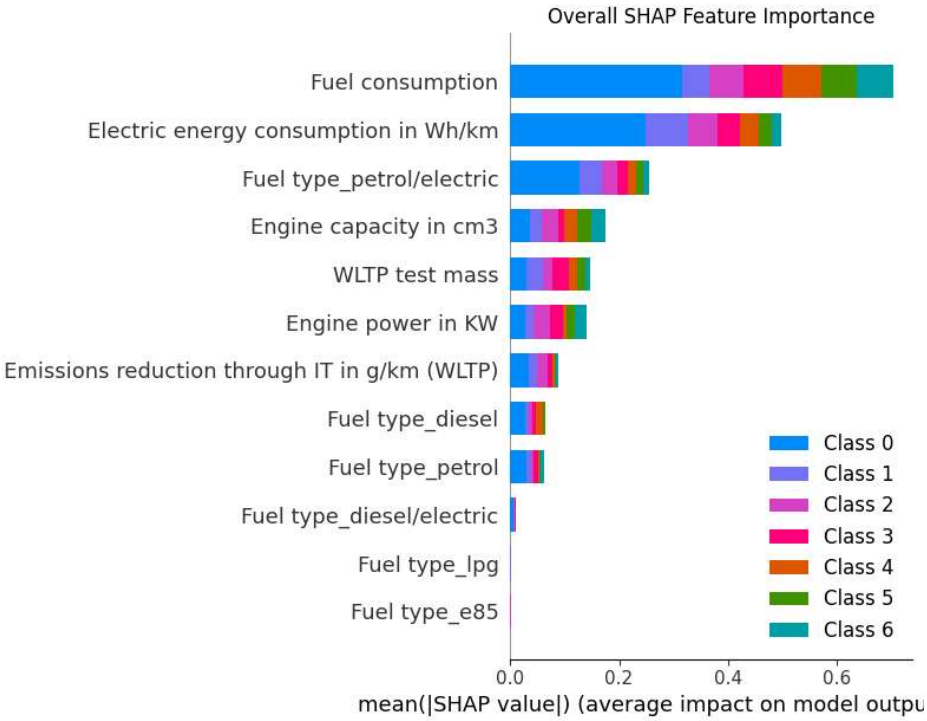


Figure 11: Model with Fuel consumption - SHAP analysis on the Random Forest model (best model)

Overall, we prefer the linear regression model for the regression, and Random Forest for the classification. Linear regression is fast and simple to interpret and has the same performance as Ridge and Lasso. Random Forest offers the optimal balance between performance and prediction speed, making it the most effective choice for classification.

4 Going further

The classification problem using EU labels gave too good results. Therefore, we have decided to execute another classification that would be slightly harder, which we enforced through a different discretization method as well as a slightly different preprocessing step. As we've done for the first classification, we will show the results for two scenarios: with and without considering the Fuel Consumption explanatory variable.

4.1 Pre-processing

- First of all, we've decided to remove all hybrid-electric vehicles (diesel/electric & petrol/electric), since these are quite easy to predict, as they don't have a reflection of their entire emission in the dataset.
- Removal of outliers on the target variable (the emissions).
- While adding some slightly-correlated preprocessing variables didn't have much impact on the results when the Fuel consumption variable was considered, adding them helped improve the results when it was disregarded.
- Including 'dummies' for Innovative technologies extraction, fuel types, countries, pools, etc.

4.2 Discretization

Instead of using the A-G standards, we've decided to make the problem harder by discretizing the target variable into more classes. For that purpose, we wrote a function called "discretize', which takes an integer **n** and splits the data into n equally populated classes. This "equally populated" choice also helped us solve the issue of the target variable being dispersed mainly around 2 classes (as shown in a graph on section 3 above), which absolved the need for over/under-sampling. We chose **n=10** as the number of classes we'd like to discretize our target variable into, but it'd be extremely easy to explore other discretizations.

4.3 Training Method

In order to find the ideal hyperparameters, all of the algorithms were first trained on a smaller dataset (under 100k records). Then, in order to be able to perform a fair comparison, all of the algorithms were trained on a 2 million records dataset.

We've tried multiple classifiers and combined them with hyperparameters, to eventually come up with the following winners:

4.4 Training Method

Random Forest Classifier

Table 48: Random Forest Classifier

	With Fuel Consumption	Without Fuel Consumption
Training Accuracy	0.9881	0.9423
Accuracy	0.9830	0.9421
F1-score	0.9829	0.9418
Precision	0.98	0.9419
Recall	0.98	0.9417

HyperParameters:

- With Fuel Consumption:
  criterion='entropy', *max_depth*=None, *max_features*='log2', *n_estimators*=350
- Without Fuel Consumption:
  criterion='entropy', *max_depth*=None, *max_features*='sqrt', n_estimators=200

Advantages: High accuracy, robust to noise, provides feature importance.

Constraints: Computationally expensive, less interpretable than individual trees.

Key Takeaways:

- Fuel consumption is crucial for CO2 prediction.
- Random Forest performs well, but requires careful hyperparameter tuning.
- Consider computational costs for large datasets

DTC+Adaptive Boosting

DTC alone:

Table 49: DTC

	With Fuel Consumption	Without Fuel Consumption
Training Accuracy	0.9788	0.9310
Accuracy	0.9772	0.9245
F1-score	0.9772	0.9241
Precision	0.9772	0.9242
Recall	0.9771	0.9240

HyperParameters:

- With Fuel Consumption:
  criterion='entropy', *max_depth*=20, *min_samples_leaf*=15, *min_samples_split*=15
- Without Fuel Consumption:
  criterion='gini', *max_depth*=25, *min_samples_leaf*=15, *min_samples_split*=15

Advantages: Interpretable, efficient, handles non-linear relationships.

Constraints: Prone to overfitting, sensitive to features, limited flexibility.

Key Takeaways:

- Fuel consumption is a key predictor.
- Hyperparameter tuning is crucial.
- Interpretable, but watch for overfitting.

DTC is a good choice for CO2 prediction, but requires careful tuning.

DTC with Adaptive Boosting:

Table 50: DTC with Adptive Boosting

	With Fuel Consumption	Without Fuel Consumption
Training	0.9861	0.9432

Accuracy		
Accuracy	0.9810	0.9310
F1-score	0.9810	0.9307
Precision	0.9810	0.9309
Recall	0.9811	0.9307

HyperParameters:

- *algorithm*='SAMME', *n_estimators*=100, *learning_rate*=1.01

Advantages: Improves accuracy, robust to noise, provides insights.

Constraints: Computationally expensive, depends on base classifier.

Key Takeaways:

- Fuel consumption is crucial.
- AdaBoost enhances performance.
- Hyperparameter tuning is essential.

Promising approach for CO2 prediction, but consider computational costs and base classifier quality.

XGBoost

Table 51: XGBoost

	With Fuel Consumption	Without Fuel Consumption
Training Accuracy	0.9822	0.9377
Accuracy	0.9791	0.9265
F1-score	0.9790	0.9263
Precision	0.9790	0.9265
Recall	0.9790	0.9262

HyperParameters:

- With Fuel Consumption:
  max_depth=20, *learning_rate*=0.2, *subsample*=0.7
- Without Fuel Consumption:
  max_depth=25, *learning_rate*=0.2, *subsample*=1

Advantages: Ensemble approach, gradient boosting, regularization.

Constraints: Computationally expensive, less interpretable.

Key Takeaways:

- Fuel consumption is crucial.
- XGBoost offers high accuracy and handles complexity.
- Hyperparameter tuning is essential.

XGBoost is a promising choice for CO2 prediction, but consider computational costs.

Bagging

Table 52: Bagging

	With Fuel Consumption	Without Fuel Consumption
Training Accuracy	0.9879	0.9573
Accuracy	0.9823	0.9385
F1-score	0.9823	0.9381
Precision	0.9823	0.9382
Recall	0.9823	0.9381

HyperParameters:

- With Fuel Consumption:
  max_features=0.92, *max_samples*=0.95, *bootstrap_features*=False, *oob_score*=True, *n_estimators*=200
- Without Fuel Consumption:
  max_features=0.92, *max_samples*=0.9, *bootstrap_features*=False, *oob_score*=True, *n_estimators*=200

Advantages: Ensemble approach, parallelizable, out-of-bag estimation.

Constraints: Computationally expensive, less interpretable.

Key Takeaways:

- Fuel consumption is crucial.
- Bagging offers strong performance and robustness.
- Hyperparameter tuning is essential.

Bagging is a promising choice for CO2 prediction, but consider computational costs.

K-Nearest Neighbors (KNN)

Table 53: KNN

	With Fuel Consumption	Without Fuel Consumption
Training Accuracy	0.9840	-
Accuracy	0.9798	-
F1-score	0.9798	-
Precision	0.9797	-
Recall	0.9798	-

HyperParameters:

- *n_neighbors*=3, *metric*='minkowski'

Advantages: Simple, non-parametric, lazy learning.

Constraints: Computationally expensive, sensitive to distance metric, suffers from curse of dimensionality.

Key Takeaways:

- Fuel consumption is crucial.
- KNN performs well, especially for smaller datasets.
- Consider computational costs and distance metric choice.

KNN is a good option for CO2 prediction, especially when interpretability and smaller datasets are priorities.

Voting Classifier

Table 54: Voting Classifier

	With Fuel Consumption	Without Fuel Consumption
Training Accuracy	0.9873	0.9557
Accuracy	0.9828	0.9388
F1-score	0.9827	0.9384
Precision	0.9827	0.9385
Recall	0.9828	9.9384

HyperParameters:

- With Fuel Consumption:
  estimators=[(‘ab_dt’, ab_dt_clf), (‘xgb’, xgb_clf), (‘rf’, rf_clf), (‘bagging’, bag_clf), (‘knn’, knn_clf)] , *voting*='soft'
- Without Fuel Consumption:
  estimators=[(‘ab_dt’, ab_dt_clf), (‘xgb’, xgb_clf), (‘rf’, rf_clf), (‘bagging’, bag_clf)] , *voting*='soft'

Advantages: Combines multiple models, reduces overfitting, can be interpretable.

Constraints: Computationally expensive, depends on base classifier quality.

Key Takeaways:

- Fuel consumption is crucial.
- Voting classifiers offer strong performance.
- Tune hyperparameters carefully.

Voting classifiers are promising for CO2 prediction, but consider computational costs and base classifier selection.

Crosstab (Voting - With Fuel Consumption):

Table 55: Crosstab Voting Classifier

Predicted Class / Real Class	0%-10%	10%+	20%+	30%+	40%+	50%+	60%+	70%+	80%+	90%+
0%-10%	40531	193	0	1	1	1	0	0	0	0
10%+	31	40220	64	10	1	0	0	0	0	0
20%+	0	92	41295	547	16	2	0	0	0	0
30%+	0	7	397	36626	210	1	6	0	0	0
40%+	0	9	48	546	43291	404	6	5	5	0
50%+	1	0	0	5	419	35606	506	9	0	3
60%+	0	0	1	3	13	235	38751	676	0	4
70%+	0	0	0	2	2	11	776	38607	458	31
80%+	0	0	0	0	1	7	2	404	42264	269
90%+	0	0	0	0	0	0	59	25	345	35939

The provided confusion matrix evaluates the performance of a classification model, likely a Voting Classifier, on a dataset with CO2 emission predictions. The rows represent the true classes (real CO2 emission levels), while the columns represent the predicted classes.

Classification Report (Voting - With Fuel Consumption):

Table 56: Classification Report Voting Classifier

	precision	recall	f1-score	support
0	1.00	1.00	1.00	40727
1	0.99	1.00	0.99	40326
2	0.99	0.98	0.99	41952
3	0.97	0.98	0.98	37247
4	0.98	0.98	0.98	44314
5	0.98	0.97	0.98	36549
6	0.97	0.98	0.97	39683
7	0.97	0.97	0.97	39887
8	0.98	0.98	0.98	42947
9	0.99	0.99	0.99	36368
accuracy			0.98	400000
macro avg	0.98	0.98	0.98	400000
weighted avg	0.98	0.98	0.98	400000

Overall Performance:

- **High Accuracy:** The overall accuracy of 0.98 indicates that the model is performing well in general.
- **Balanced Metrics:** The macro and weighted averages of precision, recall, and F1-score are all close to 0.98, suggesting balanced performance across classes.

Class-Specific Performance:

- **Class 0:** Perfect precision, recall, and F1-score, indicating excellent performance for this class.
- **Classes 1-9:** All classes exhibit high precision, recall, and F1-scores, with minor variations.
- **Class 1:** The model has perfect recall for Class 1, indicating it correctly identifies all instances of this class.

Key Takeaways:

- The Voting Classifier with fuel consumption as a predictor demonstrates strong overall performance.
- The model excels in classifying most instances accurately.
- Minor variations in performance across classes might be due to class imbalance or inherent difficulty in distinguishing certain classes.

The classification report indicates that the Voting Classifier is an effective model for CO2 emission prediction. Its high accuracy and balanced performance across classes suggest that it can reliably classify instances into different CO2 emission categories. However, further analysis of specific classes and error cases can provide valuable insights for potential improvements

5 Bibliography

1. Gary Haq and Martin Weiss, *CO2 labelling of passenger cars in Europe: Status, challenges, and future prospects,* Energy Policy, Volume 95, August 2016, Pages 324-335.

2. Datascientest training.