

# Emission de CO<sub>2</sub> par les véhicules

Modélisation



Marc BASSELIER  
Thierry GONCALVES-NOVO  
Tanguy LALOUELLE  
Louise RIGAL

## Sommaire

---

<b>Sommaire</b>	<b>2</b>
<b>1. Étapes de réalisation du projet</b>	<b>3</b>
1.1 Classification du problème	3
1.2 Choix du modèle et optimisation	3
1.2.1. Algorithmes de Machine Learning de régressions linéaires	4
1.2.2. Autres algorithmes de Machine Learning	4
1.2.3. Algorithmes de Deep Learning	5
1.2.4. Réduction du jeu de données	6
1.2.5. Modèle bespoke multi-linéaire	6
1.2.6. Bilan comparatif	8
1.3 Interprétation des résultats	8
<b>2. Conclusion</b>	<b>9</b>

## 1. Étapes de réalisation du projet

---

### 1.1 Classification du problème

---

Notre projet s'apparente à un problème de régression avec plusieurs variables explicatives. Nous cherchons à faire une prédiction fiable de la quantité de CO<sub>2</sub> (g/km) émis par un véhicule selon ses caractéristiques techniques (consommation de carburant, type de carburant utilisé, puissance administrative, mesure qui est liée à la puissance moteur, et masse à vide du véhicule).

Pour comparer la performance de nos différents modèles, les métriques suivantes sont retenues :

- **Mean Squared Error (MSE) / Erreur quadratique moyenne** qui étudie l'écart entre les valeurs réelles et celles prédites par notre modèle. Plus celle-ci est proche de zéro, plus les écarts sont faibles.

Grâce à l'élévation au carré, la MSE pénalise fortement les écarts anormaux entre valeurs réelles et valeurs prédites. Cet aspect est pertinent pour notre problème. En effet, nous ne souhaitons pas, pour des raisons notamment environnementales, de normes juridiques ou encore industrielles, que notre future prédiction ne reflète pas précisément la réalité. Néanmoins, la valeur de la MSE n'est pas toujours évidente à interpréter si le nombre d'observations est grand, c'est pourquoi nous compléterons notre analyse des modèles avec d'autres indicateurs de performance.

Dans le cas particulier de notre modélisation en Deep Learning, nous avons chaque fois observé la MSE calculée sur les jeux d'entraînement et de validation, en fonction des epochs.

- **Mean Absolute Error (MAE) / Erreur absolue moyenne** car en passant par la valeur absolue, cette métrique est moins sensible que la MSE aux grands écarts. De même que pour la MSE, plus la MAE est proche de zéro, plus les écarts sont faibles. Ces deux métriques sont complémentaires pour une modélisation de régression.

Pour chacun des modèles, nous avons également observé la **proportion de valeurs prédites qui avaient moins d' 1%, 5% et 10 % d'écarts avec les valeurs réelles correspondantes**.

### 1.2 Choix du modèle et optimisation

L'objectif que nous avons fixé pour cette problématique très concrète est d'obtenir un modèle certes performant, mais aussi facilement interprétable et compréhensible. Nous avons constaté que le projet d'apprentissage est un problème de régression. Cela nous amènera à privilégier certains algorithmes de Machine Learning. Néanmoins, notre objectif premier au cours de ce travail étant aussi la manipulation des programmes d'optimisation, nous avons testé un large

éventail d'algorithmes pour comparer leurs résultats et trouver celui qui fonctionne le mieux pour l'objectif fixé.

Nous allons donc dans un premier temps décrire les différents modèles utilisés qui s'inscrivent dans 3 grandes catégories de modèles: les modèles de Machine Learning, de Deep Learning et un modèle bespoke à partir des objets du module tensorflow. Mais avant de les lister, notons que le code a été structuré afin de simplifier fortement l'appel des modèles, ce qui nous a permis de tester facilement un grand jeu de modèles. Nous avons pour ce faire défini la fonction `train_models` qui entraîne un `GridSearchCV` sur une liste de modèles donnée en entrée, chaque modèle étant couplé à un dictionnaire de paramètre à tester. En sortie, la fonction affiche les meilleurs paramètres ainsi que les métriques retenues et décrites dans le paragraphe précédent.

### 1.2.1. Algorithmes de Machine Learning de régressions linéaires

Nous avons entraîné les 3 modèles de régressions linéaires suivants:

- Ridge (norme euclidienne), Lasso (norme absolue avec possibilité de coefficients nuls) et Elastic Net (combinaison des 2 premiers)

Ces 3 modèles donnent des résultats satisfaisants mais largement améliorables. A noter que l'ElasticNet combine effectivement les 2 modèles avec un coefficient l1 optimal de 0.75

### 1.2.2. Autres algorithmes de Machine Learning

Nous avons également entraîné 3 modèles de Machine Learning légèrement plus complexes à méthodologie alternative.

- k-NN, le principe des plus proches voisins qui optimise sur un paramétrage à 2 voisins
- `DecisionTreeRegressor`: qui reste un modèle d'arbre facilement interprétable. Le paramètre `max_depth` optimal s'établit du reste à 'None'
- `XGBRegressor`: pour tester un modèle plus complexe mais réputé très performant

Ces 3 modèles donnent d'excellents résultats rapidement, sur un jeu de paramètre standard, sans avoir besoin d'optimiser particulièrement la méthodologie liée à `GridSearchCV`. Les résultats sont par ailleurs comparables entre eux, ce qui nous pousse à préférer au sein de l'ensemble des algorithmes de Machine Learning le modèle `DecisionTreeRegressor` qui est a priori plus facilement interprétable.

[En revanche, nous n'avons testé que des modèles de régression classique, et non les modèles de classification avancés de type Bagging ni même la validation croisée qui n'est pas nécessaire

ici compte tenu des très bons résultats obtenus sur les modèles simples, qui correspondent mieux à notre objectif.]

### 1.2.3. Algorithmes de Deep Learning

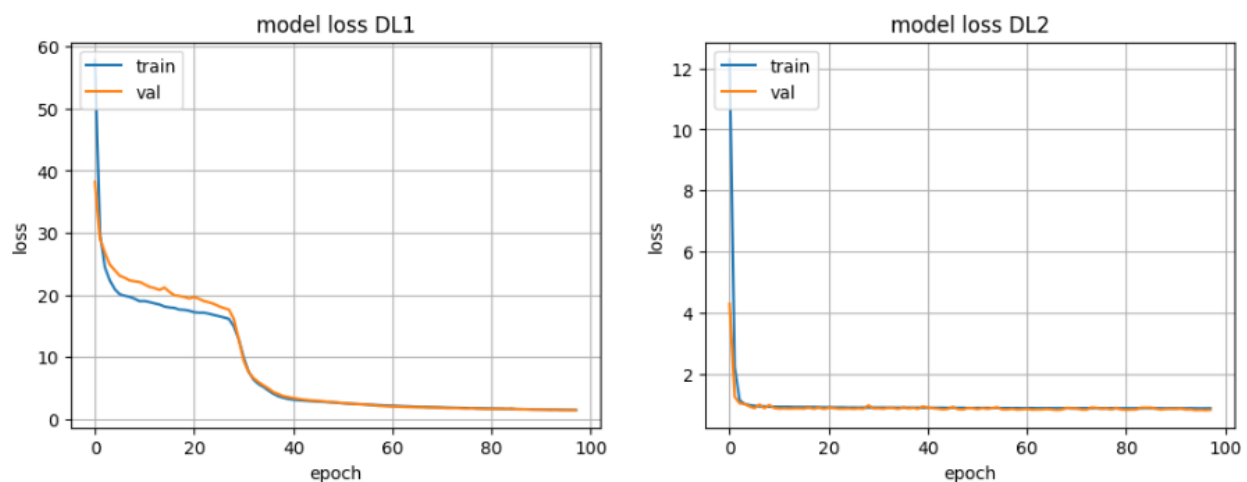
Les modèles utilisés ont pu être très divers, chacun s'étant essayé à construire son propre modèle. De manière générale, les modèles très simples donnent de très bons résultats. Nous illustrons pour cet exposé 2 versions de modèles qui partagent les éléments suivants:

- Une première couche dense de 16 neurones avec la fonction d'activation 'relu'
- Une seconde couche de sortie à 1 neurone
- Une fonction de perte 'mean square error' et l'optimizer 'adam'
- 100 epochs ainsi qu'un batch\_size de 32

La seule différence est l'entrée. Sur le premier modèle DL1, on entre les 4 variables (puissance administrative, consommation mixte, masse vide min, carburant) sans variables d'état pour le carburant. Ce modèle a 97 paramètres.

Le second modèle DL2 utilise la variable carburant après sa transformation en 5 variables d'état. Ce modèle a un total de 161 paramètres.

Les courbes respectives de model loss pour ces 2 modèles sont les suivantes:



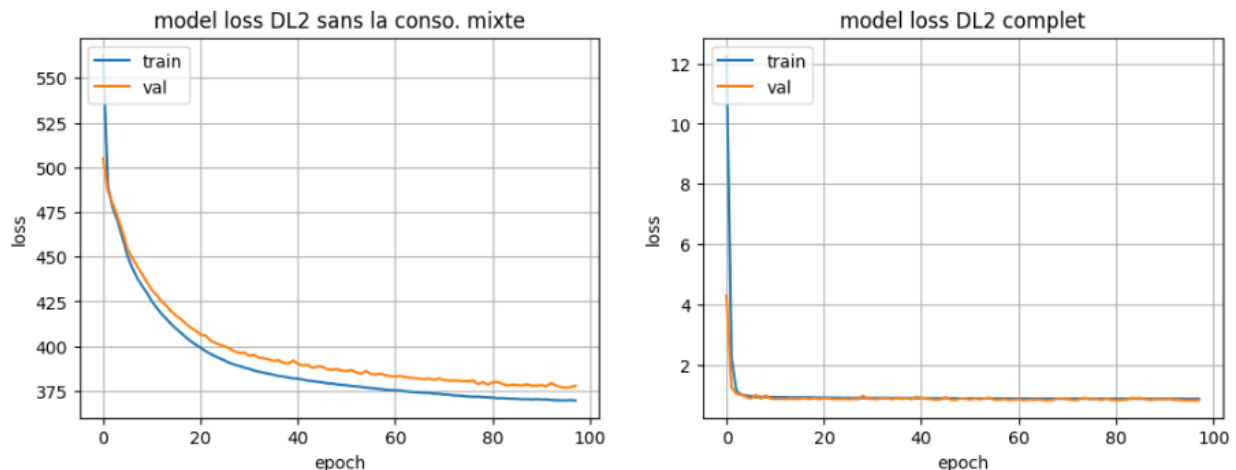
On constate l'intérêt d'utiliser les variables d'état malgré l'augmentation du nombre de paramètres. En revanche, il ne semble pas nécessaire de complexifier la structure du modèle davantage compte tenu des excellents résultats obtenus par le modèle DL2.

Les résultats de Deep Learning sont équivalents à ceux du Machine Learning. Ils sont généralement attendus comme étant inférieurs pour les problèmes de régression. Néanmoins, par souci d'interprétabilité, les modèles Machine Learning restent a priori à privilégier ici.

#### 1.2.4. Réduction du jeu de données

Une fois ces modèles simples stabilisés, nous nous sommes interrogés sur la pertinence ou non du jeu de variables descriptives. Pour l'ensemble des modèles, nous avons entraîné des versions avec des jeux de données partiels, afin d'analyser la perte de performances consécutives à ces choix.

Pour illustrer cette analyse, le plus simple est de présenter les résultats d'un modèle Deep Learning de type DL2 sur toutes les variables hormis la consommation mixte de la voiture et de les comparer au modèle DL2 présenté plus haut.



On constate que le nouveau modèle est très peu performant. Toutes les métriques se dégradent dramatiquement.

A l'inverse la question se pose de savoir comment réagirait un modèle avec pour seuls paramètres la consommation mixte et le carburant comme variables d'entrée. Et pour renforcer encore plus l'interprétabilité du modèle, on va forcer davantage la structure dans un modèle fait sur mesure.

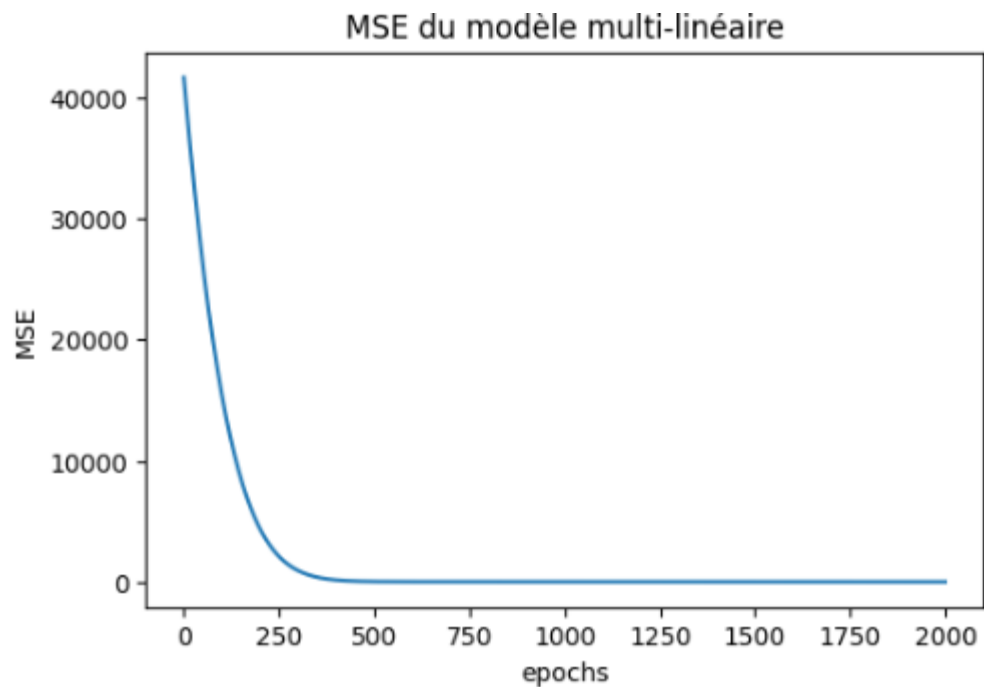
#### 1.2.5. Modèle bespoke multi-linéaire

Nous avons utilisé la bibliothèque tensorflow pour entraîner un modèle contraint à 5 régressions linéaires, une pour chacune des 5 catégories de carburant.

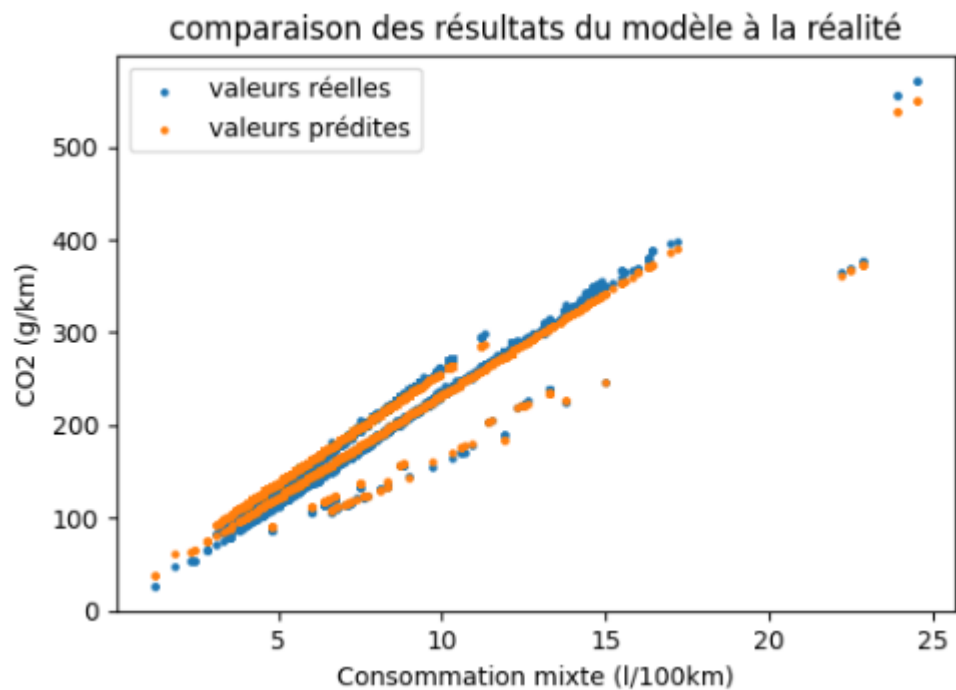
Nous avons ainsi créé une class CustomRegression pour définir ce modèle à 10 variables (la pente et l'ordonnée à l'origine des 5 régressions linéaires) ainsi qu'une fonction d'entraînement de ce modèle utilisant la méthode du gradient avec les éléments disponibles de tensorflow.

Les résultats du modèle sont très satisfaisants. Ils sont proches des meilleurs modèles vus précédemment. Et son interprétabilité est incomparable puisque le modèle ne nécessite que 10 paramètres et ceux-ci permettent de comprendre immédiatement les relations en jeu dans notre problématique.

La mesure MSE de notre modèle est illustré ci-dessous:























On retrouve par ailleurs les relations décrites lors de l'analyse des données:



Ce modèle est finalement celui que l'on retient à l'issue de notre travail.

### 1.2.6. Bilan comparatif

MODELE	Hyperparamètres	Métriques					Performance*	Interprétabilité*
		MSQE	MAE	Écart <1%	Écart <5%	Écart <10%		
Ridge	alpha: 0,01 sover: saga	32,6	2,3	57%	97,8%	99,6%		
Lasso	alpha: 0,01 selection: random	32,6	2,3	57%	97,8%	99,6%		
ElasticNet	l1_ratio: 0,75	32,6	2,3	57%	97,8%	99,6%		
k-NN	n_neighbors: 2	0,38	0,17	97,2%	100%	100%		
DecisionTree	max_depth: None	0,16	0,13	98,1%	100%	100%		
XGBRegressor	learning_rate: 0,2 max_depth: 5	0,42	0,30	96,3%	100%	100%		
DL1	-	2,1	0,81	92,1%	99,7%	99,9%		
DL2	-	0,86	0,60	92,9%	100%	100%		
DL2 sans 'conso. mixte'	-	376	14	8%	45%	77%		
<b>Bespoke</b>	-	0,86	0,62	93,4%	100%	100%		

\* : échelle relative à l'ensemble des modèles testés (vert: best, rouge: worst, orange: avg)

## 1.3 Interprétation des résultats

Nous avons réalisé le modèle TensorFlow pour obtenir la meilleure interprétabilité des résultats. Néanmoins, certains modèles ont une performance encore supérieure.

En regardant les erreurs du modèle TensorFlow, on remarque que la relation n'est pas parfaitement linéaire, on observe une certaine convexité entre la consommation mixte et le taux de CO2 émis. On pourrait évidemment corriger cette erreur, mais on perdrait immédiatement en lisibilité. Hors, la performance du modèle nous paraît suffisante pour privilégier ce modèle.

Nous avons également appliqué les techniques d'interprétabilité Shap à certains de nos modèles, notamment le Decision Tree Model. Il est intéressant de noter qu'il nous a paru difficile d'appréhender parfaitement les relations entre les variables en utilisant les outils (visualisation d'un arbre / par composant principal...). Ces outils indiquent simplement la forte dépendance à la consommation mixte, et la faible dépendance des autres paramètres. La multilinéarité n'est pas lisible à travers les résultats obtenus.



## 2. Conclusion

---

Au terme de ce projet, nous constatons sur notre échantillon la très forte dépendance linéaire entre le taux d'émission de CO<sub>2</sub> et la consommation mixte de la voiture. Néanmoins, cette linéarité est elle-même légèrement différente en fonction du carburant utilisé. Le gazole est le plus polluant avant l'essence, les 3 autres carburants ont des relations à peu de chose près équivalentes.

Ces résultats peuvent permettre au consommateur voulant réduire son empreinte carbone de choisir un véhicule, notamment en choisissant la consommation la plus faible suivant l'usage (citadin/route).

A la vue de ces résultats, il est aussi intéressant de s'interroger sur la législation en vigueur pour réduire la pollution atmosphérique, l'objectif européen étant de réduire les émissions de CO<sub>2</sub> pour le parc automobile neuf à zéro (voitures particulières et véhicules utilitaires) d'ici à 2035. La législation porte donc sur 2 volets:

- A long terme: la vente de véhicules électriques - et de fait l'interdiction de vente de véhicules thermiques à partir de 2035
- A court terme, un malus écologique pour les véhicules thermiques

Le malus écologique se déclenche à partir de 118 g/km de CO<sub>2</sub> pour un montant de 50€, franchit la barre de 200€ à 125 g/km, et celle de 1000€ pour 140 g/km (le malus est plafonné à 60 000 € à partir de 194 g/km).

En considérant une estimation très approchée de 200 000 kilomètres d'usage pour un véhicule neuf, et un prix d'un quota d'une tonne de CO<sub>2</sub> de 67€, on peut rapidement estimer le coût à vie d'une consommation unitaire d'1g/km à 13.4€. A partir du seuil de déclenchement, le malus joue plus que son rôle, il progresse plus que le coût actuel du CO<sub>2</sub>. Il est même excessif en relatif pour les véhicules les plus polluants, et peut jouer un rôle de régulation. En revanche, on peut légitimement s'interroger sur ce seuil de 118 g/km (une telle consommation représente un coût carbone de 15 800€ sur la durée de vie du véhicule avec nos hypothèses).

On connaît notamment l'engouement actuel des consommateurs pour les SUV. Leur consommation reste proche de ce seuil pour une majorité de véhicules, n'impactant pas la décision d'achat. Pourtant, les SUV consomment facilement 20% de plus que les citadines légères. On constate là une inefficience à la limitation des émissions de CO<sub>2</sub> à court terme.