

SIMPLON



MARCO CASION

Résumé

La société « Marco Casion » achète des voitures d'occasion en Inde. Marco avait pour habitude de faire des offres de reprise via un questionnaire et une réponse par email.

Pour gagner du temps et rendre son offre plus attractive, il a décidé de moderniser son process en donnant une réponse immédiate via son site internet, à l'aide d'un formulaire qui prend en compte un certain nombre de critères.

Thomas Cassagne

E1 – Rapport chef d'œuvre – Marco Casion

Table des matières

1	Référentiel des compétences abordées	3
2	Première partie	4
2.1	Compréhension du besoin client	4
2.2	Etat de l'art	5
2.2.1	Langage de programmation	5
2.2.2	Bibliothèque Python pour la manipulation des données.....	5
2.2.3	Bibliothèques Python permettant la visualisation des données.....	6
2.2.4	Bibliothèques Python permettant l'enregistrement des objets Python	6
2.2.5	Bibliothèques Python concernant la mise en place d'algorithmes d'intelligence artificielle.....	6
2.2.1	Bibliothèques Python concernant la mise en place de techniques de monitoring.....	6
2.2.2	Mise en place d'une API compatible avec Python	7
2.2.3	Gestion de base de données	7
2.3	Éléments de conception technique	8
2.3.1	Présentation de la base de données relationnelle	8
2.4	Choix techniques liés au projet	9
2.4.1	Programmation	9
2.4.2	Librairies Python	9
2.4.3	Gestion de base de données.....	10
2.5	Réponse finale apportée : ce qui a été réalisé.....	10
2.5.1	Import des données.....	10
2.5.2	Nettoyage des données	11
2.5.3	Analyse Originale - Distribution.....	11
2.5.4	Analyse Originale - Visualisation	12
2.5.5	Matrice des corrélations	15
2.5.6	Normalisation des données (voir Annexe 3).....	15
2.5.7	15
2.5.8	Mise en place des algorithmes de Machine Learning.....	16
2.5.9	Mise en place de techniques de monitoring	18
2.5.10	Création d'une API	19
3	Seconde partie	20
3.1	Organisation technique et l'environnement de développement tout au long de la production.....	20
3.1.1	Environnement de développement	20
3.2	Gestion du projet	20
3.3	Retour d'expérience sur les outils, techniques et compétences à l'œuvre tout au long du projet	20
4	Troisième partie	21

4.1	Bilan du projet.....	21
4.2	Améliorations envisageables.....	21
4.3	Conclusion.....	21
5	Annexes.....	22
5.1	Annexe 1 : Qu'est-ce qu'un DataFrame ?	22
5.2	Annexe 2: Machine Learning avec Scikit_Learn par Aurélien Guéron.....	22
5.3	Annexe 3 : Normalisation des données	23
5.4	Annexe 4 : MinMaxScaler.....	23
5.5	Annexe 5 : OneHotEncoder	23
5.6	Annexe 6 : Choisir le bon modèle	23
5.7	Annexe 7 : GridSearchCV.....	24
5.8	Annexe 8 : Hyperparamètres.....	24
5.9	Annexe 9 : Score (MAE, RMSE, R2).....	24

1 Référentiel des compétences abordées

- C1.** Qualifier les données grâce à des outils d'analyse et de visualisation de données en vue de vérifier leur adéquation avec le projet.
- C2.** Concevoir une base de données analytique avec l'approche orientée requêtes en vue de la mise à disposition des données pour un traitement analytique ou d'intelligence artificielle.
- C3.** Programmer l'import de données initiales nécessaires au projet en base de données, afin de les rendre exploitables par un tiers, dans un langage de programmation adapté et à partir de la stratégie de nettoyage des données préalablement définie.
- C4.** Préparer les données disponibles depuis la base de données analytiques en vue de leur utilisation par les algorithmes d'intelligence artificielle.
- C5.** Concevoir le programme d'intelligence artificielle adapté aux données disponibles afin de répondre aux objectifs fonctionnels du projet, à l'aide des algorithmes, outils et méthodes standards, notamment de machine learning et de deep learning.
- C6.** Développer le programme d'intelligence artificielle selon les données du projet et les éléments de conception définis, en exploitant les algorithmes et les outils standards couramment utilisés dans le domaine.
- C7.** Développer l'interaction entre les fonctionnalités de l'application et l'intelligence artificielle dans le respect des objectifs visés et des bonnes pratiques du domaine.
- C8.** Modifier les paramètres et composants de l'intelligence artificielle afin d'ajuster aux objectifs du projet les capacités fonctionnelles de l'algorithme à l'aide de techniques d'optimisation.
- C9.** Analyser un besoin en développement d'application mettant en œuvre des techniques d'intelligence artificielle afin de produire les éléments de réponses techniques.
- C10.** Concevoir une base de données relationnelle à l'aide de méthodes standards de modélisation de données.
- C11.** Développer les requêtes et les composants d'accès aux données dans un langage adapté afin de persister et mettre à jour les données issues de l'application en base de données.
- C12.** Développer le back-end de l'application d'intelligence artificielle dans le respect des spécifications fonctionnelles et des bonnes pratiques du domaine.
- C13.** Développer le front-end de l'application d'intelligence artificielle à partir de maquettes et du parcours utilisateur, dans le respect des objectifs visés et des bonnes pratiques du domaine.
- C15.** Maintenir l'application d'intelligence artificielle à l'aide des techniques de monitoring afin de détecter et corriger les éventuels dysfonctionnements.

2 Première partie

2.1 Compréhension du besoin client

La société « Marco Casion » achète des voitures d'occasion en Inde.

Marco (gérant de la société) avait pour habitude de faire des offres de reprise via un questionnaire et une réponse par email.

Pour gagner du temps et rendre son offre plus attractive, il a décidé de moderniser son process en donnant une réponse immédiate via son site internet à l'aide d'un formulaire qui prend en compte un certain nombre de critères. Afin d'attirer les vendeurs, Marco propose des prix de rachat 9% plus élevés que la moyenne observée pour ce même véhicule en Inde.

C'est dans ce cadre qu'il vous sollicite pour réaliser l'application qui permettra à tout vendeur de véhicule d'occasion de connaître le tarif de rachat de son véhicule par « Marco Casion ».

En termes de données, cette société met à disposition une base de données relationnelle qui contient les caractéristiques des véhicules d'occasion en Inde et leur prix de vente.



2.2 Etat de l'art

Les recherches portent sur les outils et techniques potentiellement utilisables pour réaliser ce projet :

- Le langage de programmation à utiliser pour programmer les fonctions qui seront utilisées dans mon travail et les algorithmes d'Intelligence artificielle
- Les librairies ou bibliothèques nécessaires concernant les :
 - Manipulations de données
 - Techniques d'intelligence artificielle, permettant de prédire quel serait le niveau de y pour des valeurs particulières de x
- Mise en place de techniques de monitoring
- Mise en place d'une API compatible avec Python

2.2.1 Langage de programmation

Sources :

- <https://docs.python.org/fr/3/tutorial/>
- <https://www.lebigdata.fr/python-langage-definition>

Python est un langage de programmation puissant et facile à apprendre. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes. Il est aussi **le langage le plus utilisé pour la Data Science**. Pour cause, ce langage est simple, lisible, propre, flexible et compatible avec de nombreuses plateformes et bibliothèques.

Le langage de programmation **Python** est connu pour :

- Être open source
- Simplicité d'apprentissage
- Large choix de librairies natives
- Large choix de librairies additionnelles
- Traitement et analyse des données
- Visualisation avancée des données
- Intelligence artificielle
- Polyvalence



2.2.2 Bibliothèque Python pour la manipulation des données

Sources :

- <http://www.python-simple.com/python-pandas/panda-intro.php>
- <https://www.lebigdata.fr/python-langage-definition>

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles.

Pandas est l'une des bibliothèques de science des données les plus populaires ; elle est aujourd'hui utilisée par un grand nombre de scientifiques et d'analystes.

Elle offre de nombreuses fonctionnalités natives très utiles. Il est notamment possible de lire des données en provenance de nombreuses sources, de créer de larges DataFrames (**Annexe 1**) à partir de ces sources, et d'effectuer des analyses agrégées basées sur les questions auxquelles on souhaite obtenir des réponses.

Pandas est une bibliothèque open source. Son nom est dérivé du terme « Panel Data » (en français « données de panel »), il est également un jeu de mots de l'expression « Python Data Analysis ».

2.2.3 Bibliothèques Python permettant la visualisation des données

Sources :

- <https://matplotlib.org/>
- <https://seaborn.pydata.org/>

Matplotlib est une bibliothèque complète permettant de créer des visualisations statiques, animées et interactives en Python. Matplotlib facilite des opérations qui pourraient apparaître complexes ou délaçâtes à réaliser.

- Créer des tracés de qualité de publication .
- Créer des figures interactives qui peuvent zoomer, faire un panoramique, mettre à jour.
- Personnaliser le style visuel et la mise en page .
- Exporter vers de nombreux formats de fichiers .
- Intégrer dans JupyterLab et les interfaces utilisateur graphiques .
- Utiliser un riche éventail de packages tiers construits sur Matplotlib.

Seaborn est une bibliothèque permettant de créer des graphiques statistiques en Python. Il s'appuie sur **Matplotlib** et s'intègre étroitement aux structures de données **Pandas** .

2.2.4 Bibliothèques Python permettant l'enregistrement des objets Python

Source :

- <https://docs.python.org/3/library/pickle.html>

Le module **Pickle** permet la **sérialisation** des objets mémoire en chaîne (et réciproquement). Il est utile pour la persistance et le transfert des données sur un réseau. La manipulation des données utilise un fichier par sérialisation, où les données stockées sous forme de chaînes.

2.2.5 Bibliothèques Python concernant la mise en place d'algorithmes d'intelligence artificielle

Source :

- <https://www.statsmodels.org/stable/index.html>
- <https://www.data-transitionnumerique.com/scikit-learn-python/>
- Machine Learning avec Scikit_Learn par Aurélien Guéron (**Annexe 2**)

Statsmodels est un module Python qui fournit des classes et des fonctions pour l'estimation de nombreux modèles statistiques différents, ainsi que pour la réalisation de tests statistiques et l'exploration de données statistiques.

Scikit-learn, encore appelé **Sklearn**, est la bibliothèque la plus puissante et la plus robuste **pour le Machine Learning en Python**. Elle fournit une sélection d'outils efficaces pour l'**apprentissage automatique** et la modélisation statistique, notamment la classification, la **régression** et le clustering via une interface cohérente en Python. Cette bibliothèque, qui est en grande partie écrite en Python, s'appuie sur **NumPy**, **SciPy** et **Matplotlib**.

2.2.1 Bibliothèques Python concernant la mise en place de techniques de monitoring

Sources :

- <https://www.actuia.com/actualite/mlflow-une-plateforme-open-source-pour-faciliter-la-creation-de-modeles-de-machine-learning/#:::text=MLflow%20est%20un%20outil%20permettant,et%20le%20d%C3%A9ploiement%20des%20mod%C3%A8les>
- <https://mlflow.org/>

MLflow est un outil permettant d'industrialiser de bout en bout les process de mise au point de projets Machine Learning. Il se donne pour ambition de simplifier le développement de projets de Machine Learning en entreprise en facilitant le suivi, la reproduction, la gestion et le déploiement des modèles.

2.2.2 Mise en place d'une API compatible avec Python

Sources :

- <https://www.section.io/engineering-education/choosing-between-django-flask-and-fastapi/>
- <https://www.djangoproject.com/>
- <https://flask.palletsprojects.com/en/2.0.x/>
- <https://fastapi.tiangolo.com/>

En informatique, une interface de programmation d'applications (API) ou interface de programmation applicative est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

Concernant la mise en place d'une API, à l'issue de mes recherches trois librairies / Framework se distinguent. Chacune d'entre elles possède sa liste d'avantages et d'inconvénients :

Tableau 1 : comparaison de 3 Frameworks sur différents critères

LIBRAIRIES / FRAMEWORK	RAPIDITE D'EXECUTION	OPTIONS	SIMPLICITE D'APPRENTISSAGE	MISE EN PLACE D'ALGORITHMES	DOCUMENTATION, EXEMPLES, AVIS
Django	★☆☆	★★★★	★★★☆☆	★★★☆☆	★★★★
Flask	★★★☆☆	★★★★	★★★★☆	★★★☆☆	★★★★
FastAPI	★★★★	★★★☆☆	★★★★	★★★★	★★★★☆

- **Django** est un framework de développement Web Python gratuit et open source utilisé dans la création de sites Web.
- **Flask** est un **micro framework** web écrit en Python. Un micro framework Web est un framework de développement Web avec une configuration facile et peut être utilisé pour développer des applications Web minimalistes.
- **FastAPI** est un **framework** Web Python moderne, open source, rapide et hautement performant, utilisé pour créer des API Web et est basé sur les astuces de type standard Python 3.6+. Il s'adapte également parfaitement au déploiement de modèles d'apprentissage machine prêts pour la production, car les modèles ML fonctionnent mieux en production lorsqu'ils sont enrôlés autour d'une API REST et déployés dans un microservice.

2.2.3 Gestion de base de données

Sources :

- <https://sqlitebrowser.org/>
- <https://db-engines.com/en/system/MariaDB%3BPostgreSQL%3BSQLite>
- <https://mariadb.org/>
- <https://www.postgresql.org/>

Afin de pouvoir acquérir et visualiser l'ensemble des données fournies (sous forme de base de données relationnelles) par la société « Marco Casion », il faut un système de gestion de base de données relationnelles (SGBDR) me permettant de travailler de manière locale, simple à mettre en place et peu gourmand en ressources.

« **SQLite** est une bibliothèque écrite en langage C qui propose un moteur de base de données relationnelle accessible par le langage SQL. Contrairement aux serveurs de base de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme. »

« **MariaDB Server** est l'une des bases de données relationnelles open source les plus populaires. Il est créé par les développeurs originaux de MySQL et garanti pour rester open source. Il fait partie de la plupart des offres cloud et est la valeur par défaut de la plupart des distributions Linux. »

« **PostgreSQL** est un puissant système de base de données relationnelle objet open source avec plus de 30 ans de développement actif qui lui a valu une solide réputation de fiabilité, de robustesse des fonctionnalités et de performances. »

Tableau 2 : Comparaison de 3 SGBDR de données sur 4 critères

LIBRAIRIES / FRAMEWORK	SIMPLICITE DE MISE EN PLACE	OPEN SOURCE / COUT	DOCUMENTATION	COMPATIBILITE PYTHON
SQLite	★★★★	★★★★	★★★☆☆	★★★★
MariaDB	★★★☆☆	★★★★	★★★★	★★★★
PostgreSQL	★★★☆☆	★★★★	★★★★	★★★★

2.3 Éléments de conception technique

2.3.1 Présentation de la base de données relationnelle

Source :

- <https://drawio-app.com/>

La société « MARCO CASON » a fourni une base de données relationnelles comportant un échantillon des voitures d'occasion vendues en Inde. L'année de construction de ces voitures varie de 1998 à 2019. Cette base est au format **.db**. Une fois importée dans DB Browser for SQLite je peux prendre connaissance des données et créer une structure visuelle des données via l'outil en ligne **Drawio**.

Schéma N° 1 : Structure visuelle de la base de données relationnelle

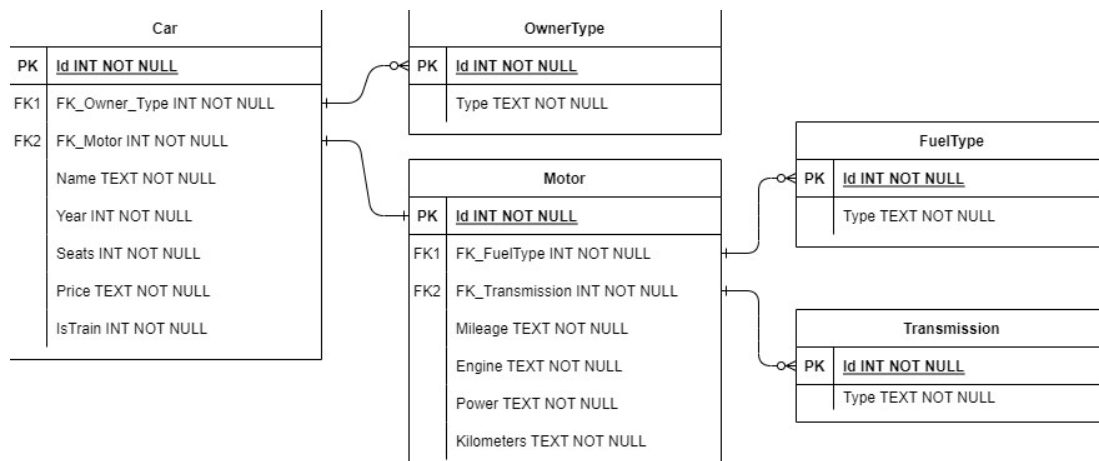


Tableau 3 : Les données relationnelles sont composées de 5 tables

TABLE N°1	TABLE N°2	TABLE N°3	TABLE N°4	TABLE N°5
Car : Voiture	Owner Type : Nombre de "mains" de la voiture	Motor : Moteur	Fuel Type : Type de carburant	Transmission : Type de transmission
PK → ID : Identification	PK → ID : Identification	PK → ID : Identification	PK → ID : Identification	PK → ID : Identification
Name : Nom	Type : Nombre de "mains" de la voiture	FK_FuelType : Type de carburant	Type : Type de carburant	Type : Type de transmission
Year : Année de fabrication		FK_Transmission : Type de transmission		
FK_OwnerType : Foreign key - Nombre de "mains"		Mileage : Consommation		
FK_Motor : Foreign key - Moteur		Kilometers : Kilomètres parcourus		
Seats : Nombre de places		Engine : Cylindrée (cm3)		
Price : Prix		Power : Puissance (chevaux din)		

2.4 Choix techniques liés au projet

2.4.1 Programmation

J'ai choisi d'utiliser le langage de programmation **Python**. En effet Python est spécialisé dans l'analyse de donnée et de mise en place d'algorithmes d'intelligence artificielle.

2.4.2 Bibliothèques Python

Manipulation des données :

- Concernant la manipulation de données le choix porte sur la librairie **Pandas**. Elle est de loin la librairie la plus répandue, documentée et simple d'utilisation.

Visualisation des données :

- Seaborn** sera la bibliothèque utilisée pour afficher les différents éléments de visualisation du projet.

Enregistrement des objets Python :

- Pickle** étant une librairie déjà déployée lors d'anciens projets, minimaliste, simple et fiable, je fais le choix de l'utiliser.

Mise en place d'algorithmes d'intelligence artificielle :

- Cette librairie est très utilisée auprès de la communauté data scientist.
J'ai aussi l'habitude d'utiliser la librairie **Sklearn** lors de mes projets IA. Je peux compter sur l'excellente documentation « *Machine Learning avec Scikit-Learn par Aurélien Guéron* » (**Annexe 2**).


Mise en place de techniques de monitoring :

- Ayant déjà pratiqué la librairie MLflow au sein de mon organisme de formation Simplon, je décide d'opter pour cette librairie.

Mise en place d'une API :

- **FastAPI** est un framework Web Python moderne, open source, rapide, performant, utilisé pour créer des API Web. **Il s'adapte également parfaitement au déploiement de modèles d'apprentissage machine prêts pour la production.** Pour l'ensemble de ces raisons mon API sera construite avec le framework FastAPI.

2.4.3 Gestion de base de données

Concernant la gestion de base de données **DB Browser for SQLite (DB4S)** est un outil visuel open source, léger, capable de créer, rechercher et modifier des fichiers de base de données compatibles avec SQLite. J'opte donc pour ce logiciel. 

2.5 Réponse finale apportée : ce qui a été réalisé

Voici les différents éléments qui ont permis de répondre au cahier des charges mis en place par l'entreprise « MARCO CASION » :

2.5.1 Import des données

Source :

- https://github.com/tomcdev63/ML/blob/main/cars/src/app/00_CLEAN_BDD.ipynb

Après avoir importé les données fournies par la société « Marco Casion » grâce à la librairie **Sqlite3**, je peux constater qu'il n'y a pas de valeur manquante au sein de celles-ci. Ces données sont complètes ce qui est un point fort pour mettre en place une prédiction correcte.

1	DATABASE = r"../../data/data.db"
---	----------------------------------

1	sql_manager = SqlManager(DATABASE)
2	data_global = sql_manager.select_dataset()
3	
4	data = data_global.copy()
5	print(data.info())
6	data.head()

Connexion réussie à SQLite
BDD importée

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5872 entries, 0 to 5871  
Data columns (total 11 columns):  
#   Column                Non-Null Count  Dtype  
---  ---  
0   Name                   5872 non-null   object  
1   Year                   5872 non-null   int64  
2   Owner_Type             5872 non-null   object  
3   Seats                  5872 non-null   int64  
4   Kilometers_Driven      5872 non-null   float64  
5   Fuel_Type              5872 non-null   object  
6   Transmission           5872 non-null   object  
7   Mileage                 5872 non-null   object  
8   Engine                 5872 non-null   float64  
9   Power                  5872 non-null   float64  
10  Price                  5872 non-null   object  
dtypes: float64(3), int64(2), object(6)  
memory usage: 504.8+ KB
```

	Name	Year	Owner_Type	Seats	Kilometers_Driven	Fuel_Type	Transmission	Mileage	Engine	Power	Price
0	Suzuki Wagon R LXi CNG	2010	First	5	72000.0	CNG	Manual	26.6 km/kg	998.0	58.16	1.75
1	Hyundai Creta 1.6 CRDi SX Option	2015	First	5	41000.0	Diesel	Manual	19.67	1582.0	126.20	12.5
2	Honda Jazz V	2011	First	5	46000.0	Petrol	Manual	18.2	1199.0	88.70	4.5
3	Suzuki Ertiga VDI	2012	First	7	87000.0	Diesel	Manual	20.77	1248.0	88.76	6.0
4	Audi A4 New 2.0 TDI Multitronic	2013	Second	5	40670.0	Diesel	Automatic	15.2	1968.0	140.80	17.74

2.5.2 Nettoyage des données

Source :

- https://github.com/tomcdev63/ML/blob/main/cars/src/app/00_CLEAN_BDD.ipynb

Un premier **notebook**, grâce à la librairie **SQLite3** et **Pandas**, permet d'extraire les données de la base de données fournie par la société, de les concaténer en un **DataFrame** et d'effectuer un premier nettoyage dit « classique ».

En observant mon jeu de données de plus près, je constate que la feature **Mileage** possède plusieurs unités (km/kg et kmpl), celles-ci diffèrent en fonction du type de carburant utilisé :

• Diesel --> kmpl*	1 data["Fuel_Type"].value_counts()
• Petrol --> kmpl*	Diesel 3152
• CNG --> km/kg*	Petrol 2655
• LPG --> km/kg*	CNG 55
	LPG 10
	Name: Fuel_Type, dtype: int64

Il y a très peu de voitures roulantes au CNG (55) et au LPG (10). Au total seulement 65 véhicules (lignes) sont présents pour ces deux catégories. Pour ne pas risquer de biaiser la prédiction sur les autres types de véhicules mais aussi parce que le nombre de données est trop faible pour obtenir une prédiction fiable sur ces deux catégories, je fais le choix de supprimer ces lignes (65), ce qui affecte globalement peu mon **Dataframe** qui possède 5872 lignes.

Pour les autres features les unités sont identiques ; de fait à l'aide de **listes compréhension** et de la fonction « **replace()** » les unités derrière les valeurs des features « **Power** », « **Engine** » et « **Mileage** » ont pu être retirées et les séries concernées transformées en type « **FLOAT** ».

Finalement notre **DataFrame** représente un total de **5807 lignes pour 11 colonnes**.

```
1 data.head()
```

	Name	Year	Owner_Type	Seats	Kilometers_Driven	Fuel_Type	Transmission	Mileage	Engine	Power	Price
1	Hyundai Creta 1.6 CRDi SX Option	2015	First	5	41000.0	Diesel	Manual	19.67	1582.0	126.20	12.5
2	Honda Jazz V	2011	First	5	46000.0	Petrol	Manual	18.2	1199.0	88.70	4.5
3	Suzuki Ertiga VDI	2012	First	7	87000.0	Diesel	Manual	20.77	1248.0	88.76	6.0
4	Audi A4 New 2.0 TDI Multitronic	2013	Second	5	40670.0	Diesel	Automatic	15.2	1968.0	140.80	17.74
6	Nissan Micra Diesel XV	2013	First	5	86999.0	Diesel	Manual	23.08	1461.0	63.10	3.5

```
1 data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 5807 entries, 1 to 5871
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Name                   5807 non-null   object
1   Year                   5807 non-null   int64
2   Owner_Type             5807 non-null   object
3   Seats                  5807 non-null   int64
4   Kilometers_Driven      5807 non-null   float64
5   Fuel_Type              5807 non-null   object
6   Transmission           5807 non-null   object
7   Mileage                 5807 non-null   object
8   Engine                 5807 non-null   float64
9   Power                  5807 non-null   float64
10  Price                  5807 non-null   object
dtypes: float64(3), int64(2), object(6)
memory usage: 544.4+ KB
```

2.5.3 Analyse Originale - Distribution

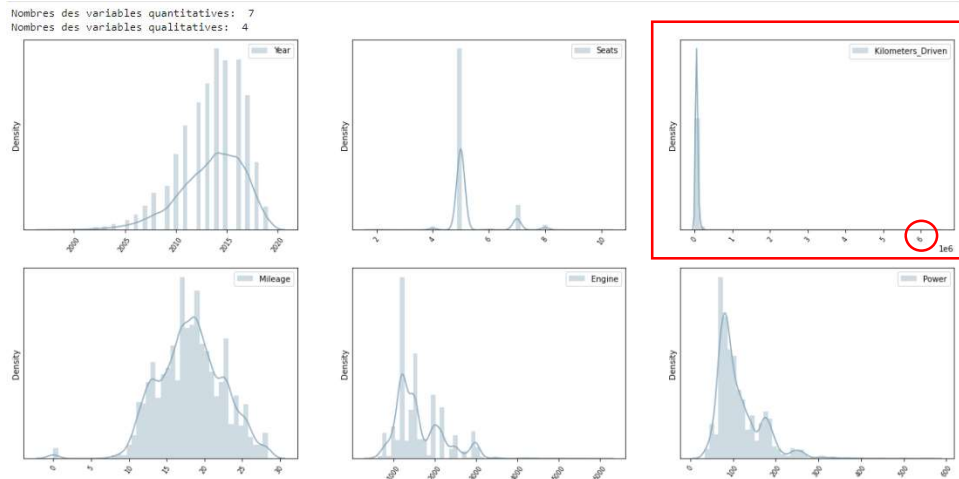
Source :

- https://github.com/tomcdev63/ML/blob/main/cars/src/app/00_MAIN_BDD.ipynb

Après avoir importé mes données et effectué un premier nettoyage « général » voici un échantillon des données ainsi que leur distribution.

Tableau 4 : Les données

	Name	Year	Owner_Type	Seats	Kilometers_Driven	Fuel_Type	Transmission	Mileage	Engine	Power	Price
0	Hyundai Creta 1.6 CRDi SX Option	2015	First	5	41000.0	Diesel	Manual	19.67	1582.0	126.20	12.50
1	Honda Jazz V	2011	First	5	46000.0	Petrol	Manual	18.20	1199.0	88.70	4.50
2	Suzuki Ertiga VDI	2012	First	7	87000.0	Diesel	Manual	20.77	1248.0	88.76	6.00
3	Audi A4 New 2.0 TDI Multitronic	2013	Second	5	40670.0	Diesel	Automatic	15.20	1968.0	140.80	17.74
4	Nissan Micra Diesel XV	2013	First	5	86999.0	Diesel	Manual	23.08	1461.0	63.10	3.50



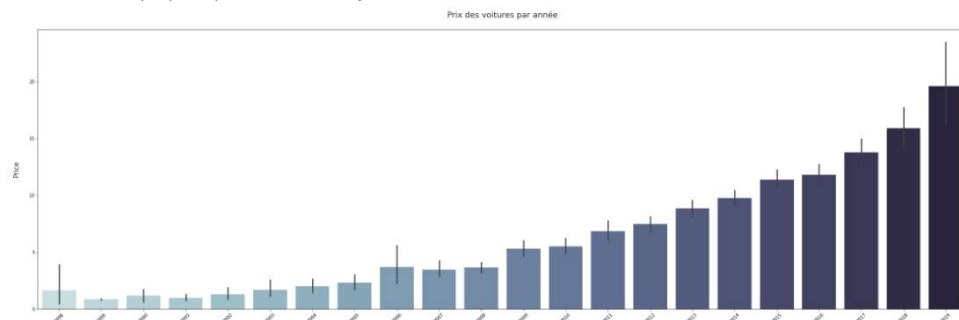
Après avoir observé la distribution de mes features, je m'aperçois qu'il y a clairement une ou des valeurs outlier dans la feature Kilometers_Driven. En effet, après analyse une voiture possède plus de 6 000 000 de kilomètres ! Cette valeur aberrante m'amène à supprimer cette ligne.

2.5.4 Analyse Originale - Visualisation

Afin de pouvoir mieux appréhender un jeu de données, émettre des hypothèses et en tirer des conclusions, il est important de passer par une étape de visualisation. A l'aide de la librairie Seaborn, j'ai pu :

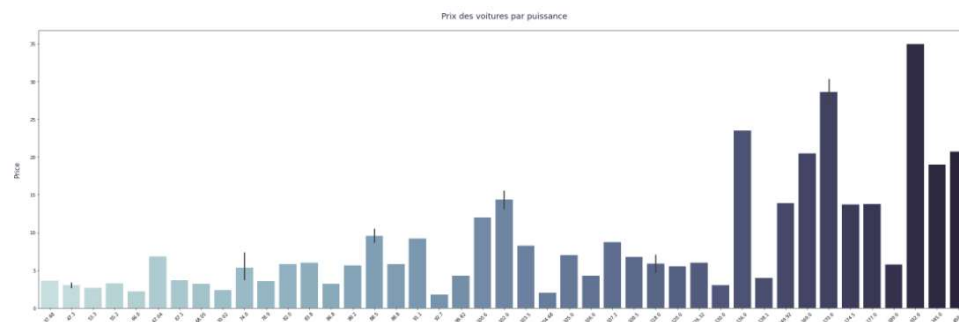
- Mieux comprendre les données
- Observer la distribution des features
- Observer des tendances générales, par exemple :
 - Plus une voiture est récente plus son prix est élevé ; cette tendance est très marquée et est très linéaire

Graphique2 : prix du véhicule en fonction de son ancienneté



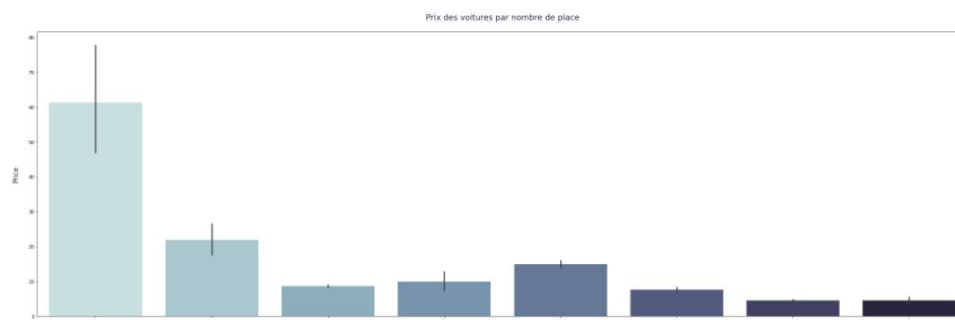
- Plus la voiture est puissante, globalement, plus elle est chère, même si l'on observe quelques variations à cette tendance

Graphique 3 : prix du véhicule en fonction de sa puissance



- Moins il y a de places dans la voiture, plus celle-ci est chère, mais cette liaison est surtout marquée pour les véhicules n'ayant que 2 places par rapport aux autres véhicules

Graphique 4 : prix du véhicule en fonction des places



Si l'on essaie de comprendre ce qui explique cette observation, on s'aperçoit que les voitures 2 places sont pour la plupart des voitures sportives, avec donc de grosses cylindrées (Engine) et beaucoup de chevaux (Power), donc logiquement plus chères (voir ci-dessous les voitures 2 places et leurs puissances respectives).

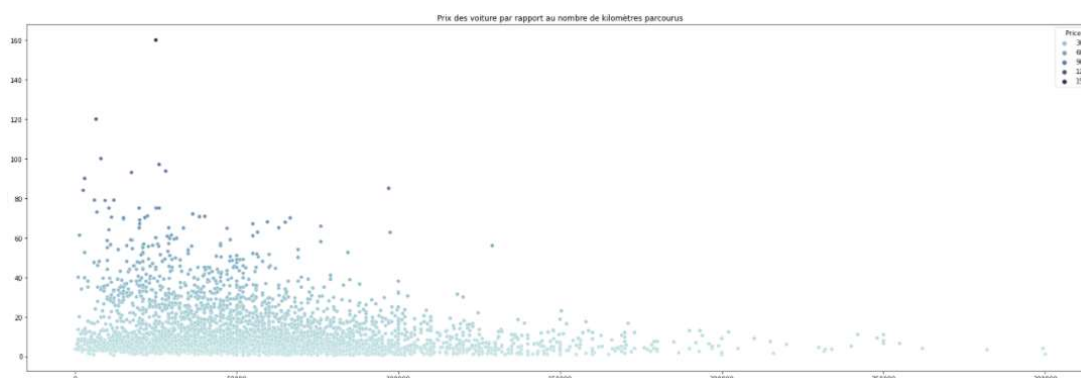
Tableau 5 : caractéristiques des véhicules de 2 places

```
1 sportives = data["Seats"]==2
2 data[sportives]
```

	Name	Year	Owner_Type	Seats	Kilometers_Driven	Fuel_Type	Transmission	Mileage	Engine	Power	Price
127	Mercedes-Benz SLC 43 AMG	2017	First	2	13372.0	Petrol	Automatic	19.00	2996.0	362.07	54.00
534	Audi TT 2.0 TFSI	2013	First	2	12100.0	Petrol	Automatic	9.90	1984.0	207.80	29.50
666	Mercedes-Benz SLK-Class SLK 350	2016	First	2	22732.0	Petrol	Automatic	18.10	3498.0	306.00	55.54
767	Mercedes-Benz SLK-Class SLK 350	2015	First	2	10000.0	Petrol	Automatic	18.10	3498.0	306.00	55.00
1038	Porsche Boxster S tiptronic	2015	First	2	10512.0	Petrol	Automatic	8.60	2706.0	265.00	64.00
1242	Audi TT 2.0 TFSI	2014	First	2	14262.0	Petrol	Automatic	9.90	1984.0	207.80	27.35
2024	Mercedes-Benz SLC 43 AMG	2019	First	2	2526.0	Petrol	Automatic	19.00	2996.0	362.07	83.96
4523	Mercedes-Benz SLK-Class 55 AMG	2014	Second	2	3000.0	Petrol	Automatic	12.00	5461.0	421.00	90.00
4550	Mercedes-Benz SL-Class SL 500	2010	First	2	35000.0	Petrol	Automatic	8.10	5461.0	387.30	29.50
4717	BMW Z4 2009-2013 Roadster 2.5i	2018	First	2	9952.0	Petrol	Automatic	10.37	2979.0	306.00	58.54
5105	BMW Z4 2009-2013 35i	2011	First	2	25000.0	Petrol	Automatic	10.37	2979.0	306.00	30.00
5580	Lamborghini Gallardo Coupe	2011	Third	2	6500.0	Petrol	Automatic	6.40	5204.0	560.00	120.00
5712	Jaguar F Type 5.0 V8 S	2015	First	2	8000.0	Petrol	Automatic	12.50	5000.0	488.10	100.00

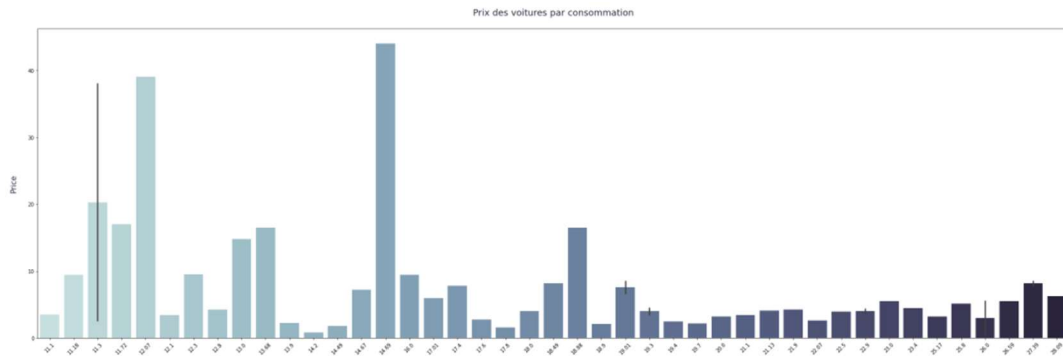
- Plus une voiture a réalisé de kilomètres moins son prix est élevé ; une certaine variabilité est observée sur cette tendance générale avec pour un même kilométrage des prix très différents

Graphique 5 : prix du véhicule en fonction du kilométrage



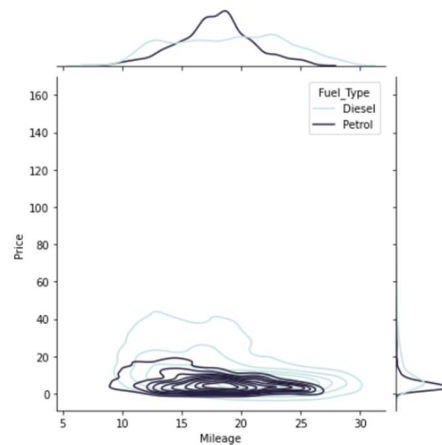
- D'après ce graphique, moins une voiture consomme plus elle est chère. L'explication vient du fait que, plus une voiture est récente, moins elle consomme de carburant. Mais plus elle est récente, plus son prix est élevé.

Graphique 6 : prix du véhicule en fonction de sa consommation



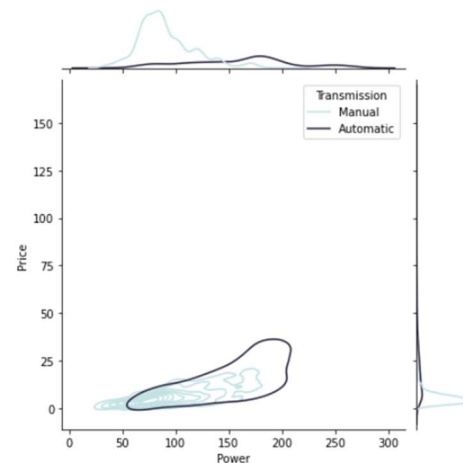
- Ce Jointplot permet de mettre en évidence qu'au sein du jeu de données certaines voitures Diesel coûtent plus chères et consomment légèrement plus que les voitures Essence.

Graphique 7 : prix et consommation du véhicule en fonction de son type de carburant



- Ce Jointplot permet de mettre en évidence qu'au sein du jeu de données les voitures à boîte automatique ont tendance à être plus chères et plus puissantes que les voitures à boîte manuelle.

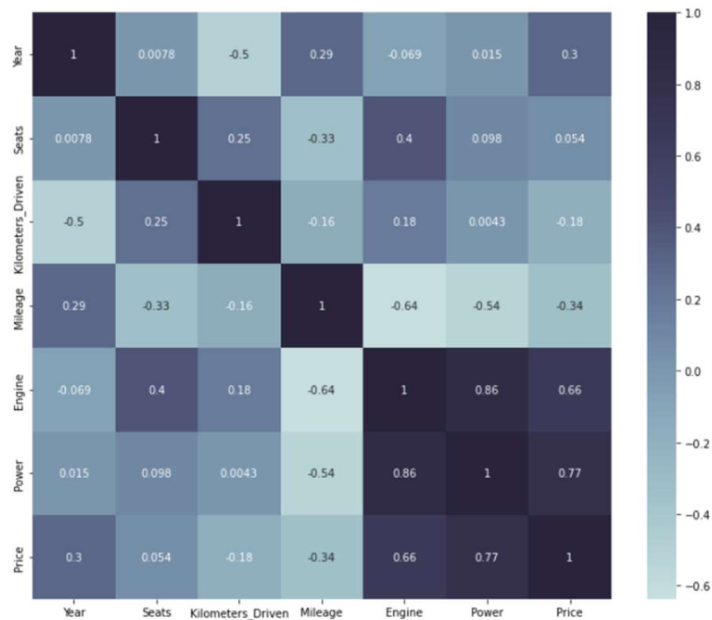
Graphique 8 : prix et puissance du véhicule en fonction de son type de boîte (automatique ou manuelle)



2.5.5 Matrice des corrélations

La matrice des corrélations entre variables permet de visualiser les liens entre les variables utilisées. Il est notamment intéressant de s'assurer de corrélations élevées ou au contraire très faibles entre variables.

Tableau 5 : matrice des corrélations



On note des corrélations positives assez marquées entre prix et puissance, ou encore des corrélations négatives entre prix et consommation de carburants et la cylindrée du véhicule.

2.5.6 Normalisation des données (voir Annexe 3)

En intelligence artificielle il est primordial de travailler avec des valeurs contenues sur des échelles similaires ou proches.

Mon jeu de données possède des features **numériques** et **catégorielles**. Je décide donc d'instancier un objet « **preprocessor** » issu de la classe « Preprocessor » créée en amont. Ce « preprocessor » a pour but d'appliquer un « **MinMaxScaler** (Voir annexe 4) » sur les colonnes de type **numériques** et un « **OneHotEncoder** (Voir annexe 5) » sur les colonnes de types **catégorielles**.

Le DataFrame regroupant les données possède à la suite de ce traitement 14 colonnes, dont toutes les valeurs sont comprises entre 0 et 1.

Tableau 6 : Les données après normalisation

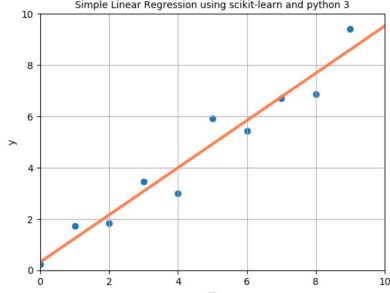
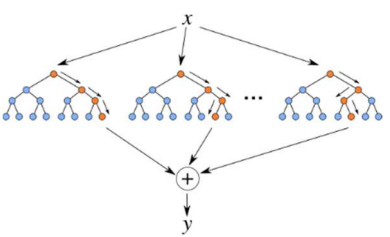
	Year	Seats	Kilometers_Driven	Mileage	Engine	Power	Owner_Type*First	Owner_Type*Second	Owner_Type*Fourth & Above	Owner_Type*Third	Fuel_Type*Diesel	Fuel_Type*Petrol
0	0.809524	0.375	0.136174	0.692606	0.178266	0.174971	1.0	0.0	0.0	0.0	1.0	0
1	0.619048	0.375	0.152850	0.640845	0.106997	0.103652	1.0	0.0	0.0	0.0	0.0	1
2	0.666667	0.625	0.289595	0.731338	0.116115	0.103766	1.0	0.0	0.0	0.0	1.0	0
3	0.714286	0.375	0.135074	0.535211	0.250093	0.202739	0.0	0.0	1.0	0.0	1.0	0
4	0.714286	0.375	0.289592	0.812676	0.155750	0.054964	1.0	0.0	0.0	0.0	1.0	0

2.5.8 Mise en place des algorithmes de Machine Learning

Après avoir consulté la documentation de SKlearn permettant de choisir le bon modèle (Voir annexe 6) je décide de mettre en place deux algorithmes de Machine Learning utilisant le principe de régression :

- **Sklearn LinearRegression** afin de m'assurer de la capacité à obtenir un résultat correct. Il s'agit ici d'une forme de test avant d'expérimenter un algorithme plus complexe.
- **Sklearn RandomForestRegressor**

Tableau 7: présentation des algorithmes utilisés

Sklearn LinearRegression	Source : https://fr.wikipedia.org/wiki/R%C3%A9gression_lin%C3%A9aire#:~:text=En%20statistiques%2C%20en%20%C3%A9conom%C3%A9trie%20et,ou%20plusieurs%20variables%2C%20dites%20explicatives
	<p>En statistique, en économie et en apprentissage automatique, un modèle de régression linéaire est un modèle de régression qui cherche à établir une relation linéaire entre une variable, dite expliquée, et une ou plusieurs variables, dites explicatives. On parle aussi de modèle linéaire ou de modèle de régression linéaire. Comme les autres modèles de régression, le modèle de régression linéaire est aussi bien utilisé pour chercher à prédire un phénomène que pour chercher à l'expliquer. Après avoir estimé un modèle de régression linéaire, on peut prédire quel serait le niveau de y pour des valeurs particulières de x.</p>
Sklearn RandomForestRegressor	Sources : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f
	<p>La forêt aléatoire est un algorithme d'apprentissage supervisé qui utilise une méthode d'apprentissage d'ensemble pour la classification et la régression. Les arbres en forêts aléatoires sont exécutés en parallèle. Il n'y a pas d'interaction entre ces arbres lors de la construction des arbres. Il fonctionne en construisant une multitude d'arbres de décision au moment de l'apprentissage et en produisant la classe qui est le mode des classes (classification) ou la prédiction moyenne (régression) des arbres individuels. Une forêt aléatoire est un méta-estimateur (c'est-à-dire qu'il combine le résultat de plusieurs prédictions) qui agrège de nombreux arbres de décision, avec quelques modifications utiles :</p> <ul style="list-style-type: none"> • Le nombre d'entités pouvant être fractionnées à chaque nœud est limité à un certain pourcentage du total (appelé hyperparamètre). Cela garantit que le modèle d'ensemble ne repose pas trop sur une caractéristique individuelle et fait un usage équitable de toutes les caractéristiques potentiellement prédictives. • Chaque arbre tire un échantillon aléatoire de l'ensemble de données d'origine lors de la génération de ses divisions, ajoutant un élément supplémentaire de caractère aléatoire qui empêche le surajustement.

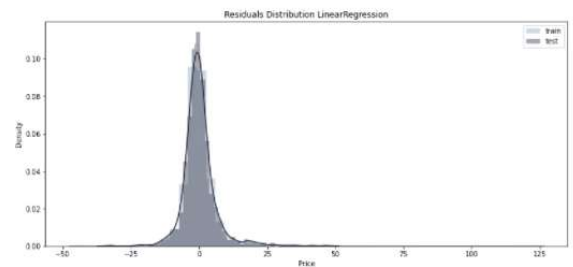
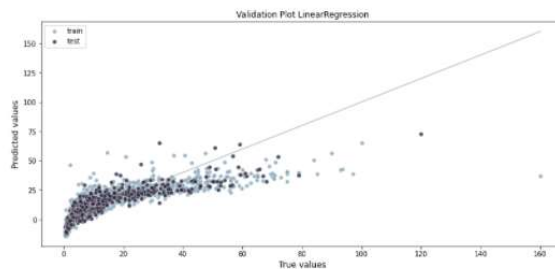
Lors du développement de mon projet j'ai décidé d'appliquer l'algorithme LinearRegression à des fins de test. Cette approche me permet d'étudier les résultats obtenus avec ce modèle dit « classique » et de les comparer à des modèles plus évolués.

Création d'un modèle de regression linéaire avec LinearRegression

```
1 model_ll = trainmodel(LinearRegression, X_train, y_train, X_test, y_test)
2
3 plot_ll = plot_validation_bis(model_ll, X_train, y_train, X_test, y_test, TARGET, "LinearRegression", COLORS[2], COLORS[6])
```

Score du jeu TRAIN
MAE: 3.7448482516397785
RMSE: 6.185174510894601
Median abs err: 2.4868089990973723
R2: 0.69732649799762

Score du jeu TEST
MAE: 3.8231591018670574
RMSE: 5.979414943306584
Median abs err: 2.483653307505286
R2: **0.7247163194631815**



Le score R2 de l'algorithme LinearRegression est de 0.72%. Comme le montre le premier graphique l'ensemble des points ne suit pas l'équation donnée par la régression linéaire issue de l'algorithme LinearRegression de la librairie SKlearn. On observe en particulier un éloignement des valeurs réelles par rapport aux valeurs prédites pour les valeurs réelles élevées ou très faibles.

Création d'un modèle de regression avec RandomForestRegressor 🌲

```

1 model_rfr = trainmodel(RandomForestRegressor, X_train, y_train, X_test, y_test)
2
3 param_grid = [
4     {
5         "max_depth" : [80, 90, 100, None],
6         "n_estimators" : [20, 50, 100],
7         "max_features": ["auto", "sqrt"],
8         "criterion" : ["squared_error", "absolute_error", "poisson"]
9     }
10 ]
11
12 grid_rfr = trainmodelGSCV(model_rfr, X_train, y_train, param_grid)
13
14 model_rfr = grid_rfr.best_estimator_
15
16 plot_rfr = plot_validation_bis(model_rfr, X_train, y_train, X_test, y_test, TARGET, "RandomForestRegressor", COLORS[2], COLO

```

Score du jeu TRAIN

MAE: 0.6186984909058257

RMSE: 1.5054327317568839

Median abs err: 0.2732499999999969

R2: 0.9820694671649806

Score du jeu TEST

MAE: 1.6259364978448279

RMSE: 3.69731636443902

Median abs err: 0.7038499999999985

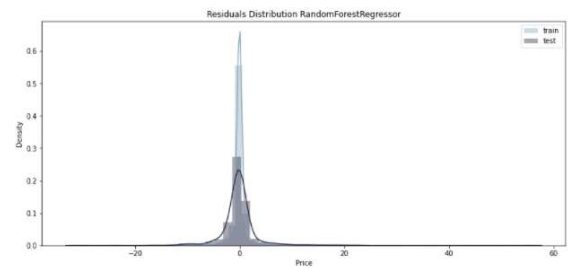
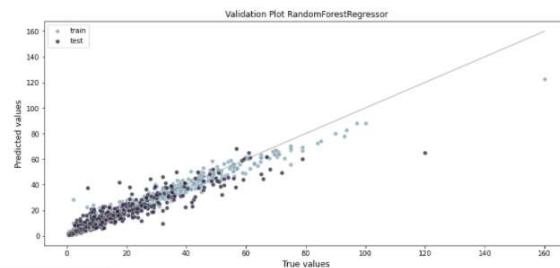
R2: 0.8947465579597457

Score après GridSearchCV

CV Mean: 0.8673514064356065

STD: 0.03281903043838012

Le meilleur score est de: 0.8385841350703653 avec {'criterion': 'poisson', 'max_depth': 80, 'max_features': 'sqrt', 'n_estimators': 50}



Il faut noter que de façon générale et pour une très large majorité des situations, la prédiction est d'un bon niveau. La plupart des points sont très proches de la droite qui symbolise les valeurs attendues. Les écarts entre les valeurs attendues et les valeurs prédites sont donc corrects dans ce modèle. On remarque cependant encore une légère « dérive » surtout pour les valeurs les plus élevées. En revanche la prédiction est très efficace en dessous de \$40 000, principalement dû au fait que le nombre de données ayant permis de déterminer la prédiction est important. Cela se traduit par le score R2 (du jeu de TEST) de l'algorithme RandomForestRegressor qui est de 0.83% après avoir effectué un GridSearchCV (**Annexe 7**). C'est un score correct.

2.5.9 Mise en place de techniques de monitoring

Grâce à la librairie **MLFlow** les divers tests concernant les algorithmes employés ainsi que leurs hyperparamètres (**voir Annexe 8**) ont pu être sauvegardés au sein d'une interface graphique. Cela a pour but d'avoir une représentation visuelle et graphique de l'ensemble des essais effectués ainsi que de détecter et corriger les éventuels dysfonctionnements et anomalies pouvant survenir.

Après avoir fait tourner le notebook regroupant les différents algorithmes d'IA, il suffit alors de lancer le serveur local MLFlow et de se rendre dans le dossier src/app/, puis d'exécuter la commande :

- mlflow ui

				Metrics			Parameters >		
<input type="checkbox"/>	↓ Start Time	Duration	Rt	CV Mean	R2	STD	criterion	max_depth	max_feature
<input type="checkbox"/>	🟢 2 minutes ago	122ms		0.867	0.839	0.033	poisson	100	sqrt
<input type="checkbox"/>	🟢 7 minutes ago	234ms		0.867	0.839	0.033	poisson	80	sqrt

2.5.10 Création d'une API

Afin de répondre au mieux à la demande du client, un site internet intégrant un module d'estimation (indiquant le prix moyen d'une voiture d'occasion reprise en Inde et celui appliqué par la société "MARCO CASION" : 9% de plus) a été mis en place grâce au **framework FastAPI**.

Ainsi les futurs vendeurs désireux de connaître le prix de rachat de leur véhicule par la société Marco Casion n'auront plus qu'à se rendre sur le site de Marco et indiquer les différents critères de leur véhicule afin d'avoir une estimation du prix de rachat de celui-ci.

Ce site est accessible en ligne à l'adresse (serveur personnel Ubuntu) :

- <http://ml.car.tomdev.ovh/>

Ou de manière locale, en se rendant dans le dossier src/app/ et en exécutant la commande :

- `uvicorn main:app`

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

PS G:\VAF\TOMDEV\EXP\U2\src\app> uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['G:\VAF\TOMDEV\EXP\U2\src\app']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reload process [6766] using statreload
INFO: Started server process [7188]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:62675 - "GET / HTTP/1.1" 200 OK
INFO: 127.0.0.1:62675 - "GET /static/css/main.css HTTP/1.1" 200 OK
INFO: 127.0.0.1:62676 - "GET /static/images/25231.png HTTP/1.1" 200 OK
INFO: 127.0.0.1:62676 - "GET /static/images/vendeur_veitures.jpg HTTP/1.1" 200 OK
INFO: 127.0.0.1:62676 - "GET /static/images/logo.png HTTP/1.1" 200 OK
```

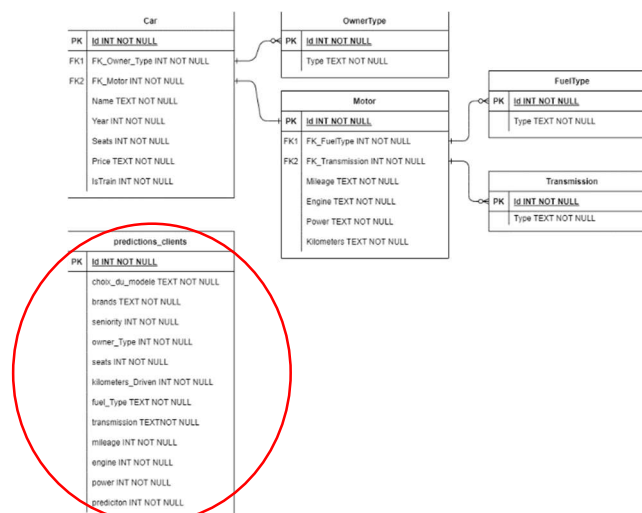


De plus, afin de permettre à Marco de **conserver un historique** des différentes demandes des éventuels futurs vendeurs, une nouvelle table a été créée au sein de la base de données relationnelles fournie :

- Schéma 2 : la base de données relationnelle

Source :

- https://github.com/tomdev63/ML/blob/main/cars/src/static/images/data_diagram.drawio.png



3 Seconde partie

3.1 Organisation technique et l'environnement de développement tout au long de la production

3.1.1 Environnement de développement

Afin de pouvoir installer les différentes librairies requises et de choisir une version stable du langage python ainsi que des diverses bibliothèques utilisées, j'ai choisi d'installer un nouvel environnement :

```
conda create --name nom_de_l'environnement python=3.8
```

De ce fait mon environnement de travail est plus léger et surtout lié uniquement au projet « Marco Casion »

Concernant l'IDE (environnement de développement intégré) j'ai choisi **VSCODE**, avec lequel j'ai l'habitude de travailler (script python et notebook).



3.2 Gestion du projet

Concernant la gestion du projet « Marco Casion », j'ai utilisé l'outil **Github** afin de sauvegarder mon travail et pouvoir travailler de manière « nomade » sur celui-ci (ordinateur fixe, portable ...).



De plus j'ai utilisé un système de **Kanban** afin de pouvoir **lister** les différentes étapes de mon projet ainsi que leurs avancements respectifs.



- La première étape a consisté à rechercher les outils nécessaires pour mener à bien ce travail et notamment les outils permettant de prédire une valeur donnée à partir de plusieurs variables explicatives.
- A partir de là, il a été nécessaire d'étudier les données fournies à partir desquelles il faut créer l'appli demandé.
- Cette étude a permis d'abord d'écarter des données aberrantes ou des situations qui ne pourraient pas être traitées sur ce travail.
- Différents traitements ont été appliqués aux données pour utiliser toute l'information disponible via les traitements statistiques utilisés

Des points d'étapes ont été faits régulièrement avec un de mes formateurs de Simplon pour m'assurer de la validité de mes démarches et de la progression de mon travail.

3.3 Retour d'expérience sur les outils, techniques et compétences à l'œuvre tout au long du projet

Ce projet m'a permis de construire de A à Z une application accessible, utilisant différents algorithmes de Machine Learning et générant de nouvelles données en fonction des différentes requêtes émises par les futurs vendeurs se dirigeant sur le site web de MARCO CASION.

Grâce à des outils simples et complets tel que FASTapi, nous pouvons facilement réaliser une application bénéficiant des dernières « technologies » tel que le HTML, CSS, Python et y intégrer un ou plusieurs modèles d'intelligence artificielle...

Ce projet montre aussi que les algorithmes récents tel que RandomForestRegressor, XGBoost etc, permettent d'obtenir de bons résultats sans avoir pour autant effectué une phase de Feature Engineering ou un nettoyage approfondi de nos données.

Il sera alors intéressant d'améliorer ce premier score obtenu par l'algorithme RandomForestRegressor : $R^2 = 0.83\%$ (voir rapport E2 : amélioration d'un modèle).

4 Troisième partie

4.1 Bilan du projet

Finalement, le résultat obtenu est satisfaisant car l'algorithme est performant et permet, avec un nombre de variables relativement limité, de proposer un prix de rachat cohérent avec le marché de l'occasion indien.

Ce travail m'a permis de me familiariser avec différents outils statistiques et plus particulièrement les outils de régression. L'analyse des données via différentes techniques, leur qualification est aussi un apport intéressant sur ce projet.

La préparation des données (normalisation) en vue de leur utilisation par les algorithmes a également été nécessaire.

Ce projet a ainsi permis d'utiliser la plupart des compétences à valider dans le cursus de formation, de la gestion des données, à la conception et au développement d'un programme d'intelligence artificielle pour aboutir à une API et à un serveur web permettant d'utiliser le modèle.

4.2 Améliorations envisageables

Plusieurs améliorations sont envisageables pour rendre le modèle prédictif plus fiable et plus complet, par exemple en utilisant :

- Un plus grand nombre de données permettrait d'obtenir une prédiction plus fiable sur le périmètre actuel cela pour les catégories de véhicules les moins représentées
- L'ajout de véhicules plus anciens à la base de données
- De nouvelles features pourraient aussi certainement apporter des informations supplémentaires pour améliorer cette prédiction :
 - Etat du véhicule
 - Poids du véhicule
 - Type de véhicule (citadine, berline, 4X4...)
 - Options du véhicule
 - Finitions du véhicule
 - Le continent et pays du siège social de la marque du véhicule
 - La prise en compte de la marque du véhicule

Voir rapport E2 : amélioration d'un modèle

4.3 Conclusion

Un négociant en voitures d'occasion, Marco Casion, a souhaité proposer sur son site une application permettant de donner à des vendeurs de véhicules d'occasion le prix de rachat proposé par sa société pour ces véhicules.

A partir de données fournies par cette société, l'étude et le traitement de ces informations puis l'application d'un modèle statistique ont permis d'aboutir à une prédiction correcte et à un algorithme qui donne la valeur de rachat proposée par la société ; ce module a été intégré au site via une API répondant ainsi à la demande du négociant.

De plus, la société a désormais accès à une nouvelle table (prédiction client) au sein de la base de données déjà existante.

Celle-ci a pour but de **conserver un historique des estimations réalisées, et d'alimenter la base de données.**

5 Annexes

5.1 Annexe 1 : Qu'est-ce qu'un DataFrame ?

Source :

- http://www.xavierdupre.fr/app/ensae_teaching_cs/helpsphinx/notebooks/td1a_cenonce_session_10.html

Un Data Frame est un objet qui est présent dans la plupart des logiciels de traitements de données, c'est une **matrice**, chaque colonne est de même type (nombre, dates, texte), elle peut contenir des valeurs manquantes. On peut considérer chaque colonne comme les variables d'une table.

En bref un DataFrame c'est :

- La structure de données la plus utilisée dans un projet de Data Science
- Un tableau avec des lignes et des colonnes, comme une table SQL ou une feuille de calcul Excel
- Une table, qui peut stocker en mémoire des données dans des formats différents
- Une ou plusieurs Series
- Mutable

Exemple d'un DataFrame :

	state	color	food	age	height	score
Jane	NY	blue	Steak	30	165	4.6
Niko	TX	green	Lamb	2	70	8.3
Aaron	FL	red	Mango	12	120	9.0
Penelope	AL	white	Apple	4	80	3.3
Dean	AK	gray	Cheese	32	180	1.8
Christina	TX	black	Melon	33	172	9.5
Cornelia	TX	red	Beans	69	150	2.2

5.2 Annexe 2: Machine Learning avec Scikit_Learn par Aurélien Guéron



5.3 Annexe 3 : Normalisation des données

Source :

- <https://dataanalyticspost.com/Lexique/normalisation/#:~:text=Normalisation%20%3A%20La%20normalisation%20est%20une,l'aapplication%20de%20certains%20algorithmes.&text=Cette%20m%C3%A9thode%20a%20en%20outre,dans%20la%20fouille%20de%20donn%C3%A9es.>

Normalisation : La normalisation est une méthode de prétraitement des données qui permet de réduire la complexité des modèles. C'est également un préalable à l'application de certains algorithmes. ... Cette méthode a en outre de nombreuses applications dans la fouille de données.

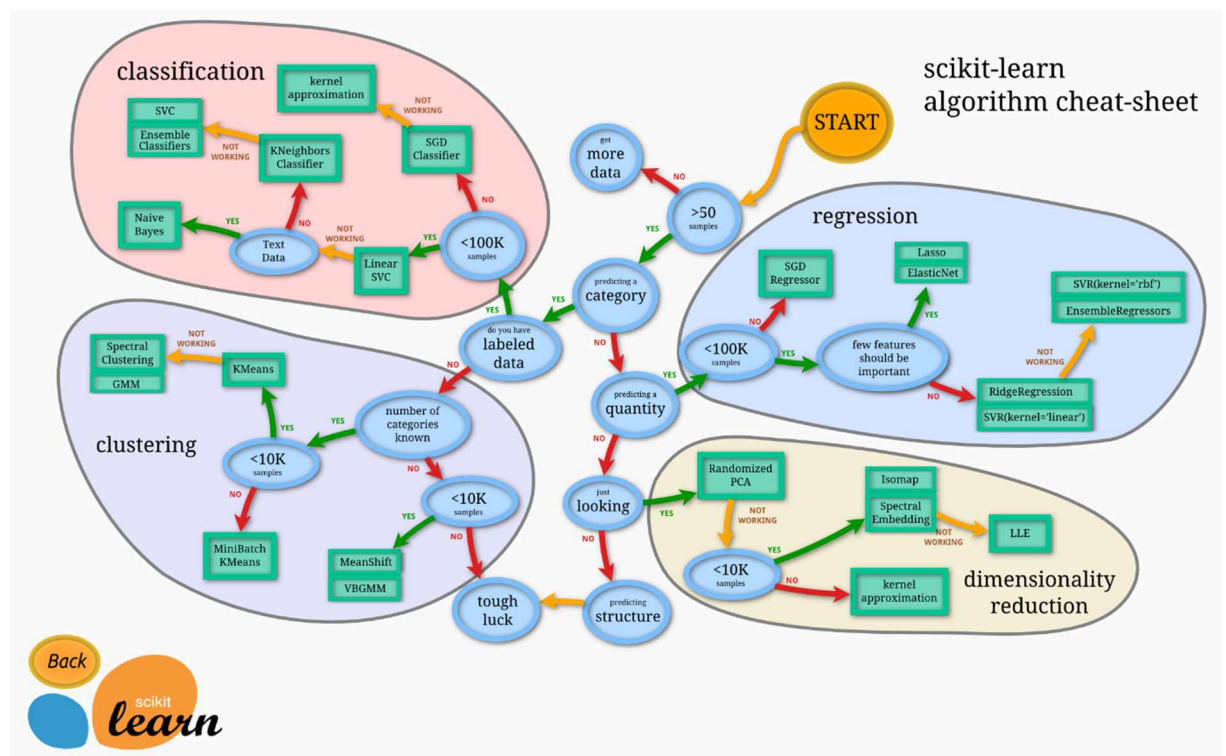
5.4 Annexe 4 : MinMaxScaler

Transforme les entités en adaptant chaque entité à une plage donnée. Cet estimateur met à l'échelle et traduit chaque caractéristique individuellement de telle sorte qu'elle se situe dans la plage donnée sur l'ensemble d'apprentissage, par exemple entre zéro et un.

5.5 Annexe 5 : OneHotEncoder

Encode les caractéristiques catégorielles sous la forme d'un tableau numérique unique. L'entrée de ce transformateur doit être un tableau d'entiers ou de chaînes, indiquant les valeurs prises par les caractéristiques catégorielles (discrètes). Les caractéristiques sont encodées à l'aide d'un schéma d'encodage one-hot (alias 'one-of-K' ou 'dummy'). Cela crée une colonne binaire pour chaque catégorie et renvoie une matrice clairsemée ou un tableau dense.

5.6 Annexe 6 : Choisir le bon modèle



5.7 Annexe 7 : GridSearchCV

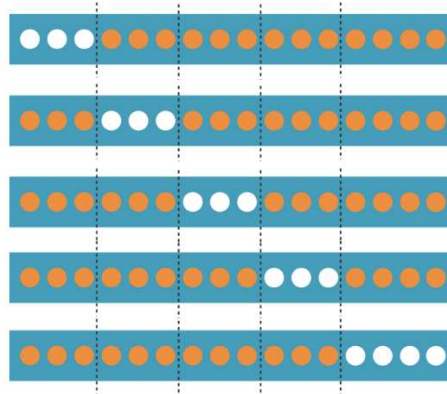
Sources :

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Recherche exhaustive sur les valeurs de paramètre spécifiées pour un estimateur.

GridSearchCV implémente une méthode "fit" et une méthode "score". Il implémente également "score_samples", "predict", "predict_proba", "decision_function", "transform" et "inverse_transform" s'ils sont implémentés dans l'estimateur utilisé.

Les paramètres de l'estimateur utilisé pour appliquer ces méthodes sont optimisés par une recherche de grille à validation croisée sur une grille de paramètres.



5.8 Annexe 8 : Hyperparamètres

Sources :

- <https://fr.wikipedia.org/wiki/Hyperparam%C3%A8tre>

Dans l'apprentissage automatique, un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage. En revanche, les valeurs des autres paramètres (généralement la pondération de nœuds) sont obtenues par apprentissage.

Un exemple d'hyperparamètre de modèle est la topologie et la taille d'un réseau de neurones. Des exemples d'hyperparamètres d'algorithme sont la vitesse d'apprentissage et la taille des lots.

Les différents hyperparamètres varient en fonction de la nature des algorithmes d'apprentissage, par exemple certains algorithmes d'apprentissage automatique simples (comme la régression des moindres carrés) n'en nécessitent aucun. Compte tenu de ces hyperparamètres, l'algorithme d'apprentissage apprend les paramètres à partir des données. Par exemple, la régression LASSO est un algorithme qui ajoute un hyperparamètre de régularisation à la régression des moindres carrés, qui doit être défini avant d'estimer les paramètres via l'algorithme d'apprentissage.

5.9 Annexe 9 : Score (MAE, RMSE, R2)

MAE : Différence absolue entre les vraies valeurs et les valeurs prédites.

MSE : Moyenne des écarts au carré entre les vraies valeurs et les valeurs prédites.

RMSE : Correspond à la racine carrée du MSE.

Median ABS error : Médiane des différences absolues des erreurs.

CV mean : Moyenne des différents score R2 produits après avoir effectué un GridSearch.

STD : Dispersion des points autour de la moyenne des différentes distributions.