FSBM

FACULTE DES SCIENCES BEN M'SICK
UNIVERSITÉ HASSAN II DE CASABLANCA

RAPPORT PROJET DE FIN DE MODULE

Système de Prédiction des Prix de Voitures d'Occasion au Maroc

Réalisé par :

- ISMAIL MOUSDIK
- MOHAMED EDDAOUDI

Encadre par :

- PR. SANAA EL FILALI
- PR. LAILA EL JIANI

Module : Apprentissage Automatique

Formation : licence d'excellence en intelligence Artificielle

Etablissement : Faculté des Sciences Ben M'sick.

Année universitaire : 2024-2025

Table de Contenu

Liste des Figures

Résume

Le marché des voitures d'occasion au Maroc est un secteur économique dynamique, caractérisé par une forte demande mais souffrant d'une évaluation subjective des prix. Ce projet vise à développer un système basé sur l'intelligence artificielle pour prédire avec précision les prix des voitures d'occasion en fonction de leurs caractéristiques techniques et des tendances du marché local. À travers une méthodologie structurée, nous avons collecté des données à partir de la plateforme Avito.ma via des techniques de web scraping, nettoyé et préparé ces données, puis effectué une analyse exploratoire pour identifier les facteurs influençant les prix. Quatre modèles d'apprentissage automatique (Decision Tree, Random Forest, XGBoost, Ridge) ont été entraînés et évalués, avec le modèle Random Forest (Version 3) démontrant les meilleures performances (R^2 = 0.902, RMSE = 26520.342). Le modèle a été déployé sous forme d'une API RESTful utilisant FastAPI et intégré dans une extension Chrome, Car Price Estimator Pro, permettant aux utilisateurs d'obtenir des estimations de prix directement sur Avito.ma ou via une saisie manuelle. Ce produit minimum viable (MVP) offre une solution pratique pour réduire l'asymétrie d'information sur le marché, avec des perspectives d'amélioration incluant un déploiement cloud et une optimisation des modèles. Ce rapport démontre la faisabilité d'un outil d'estimation des prix fiable et accessible, contribuant à une meilleure transparence dans les transactions automobiles au Maroc.

Introduction générale

Le marché des voitures d'occasion au Maroc représente un secteur économique clé, où des millions de transactions sont effectuées chaque année, principalement via des plateformes en ligne comme Avito.ma et Moteur.ma. Cependant, l'évaluation précise des prix reste un défi majeur en raison de la multiplicité des facteurs influençant la valeur d'un véhicule, tels que l'année de fabrication, le kilométrage, la marque, le modèle, le type de carburant, et l'état général. Cette complexité entraîne une asymétrie d'information, où les vendeurs risquent de sous-évaluer ou de surévaluer leurs véhicules, et les acheteurs peinent à identifier les offres justes. Face à cette problématique, l'utilisation de l'intelligence artificielle (IA) offre une opportunité unique pour développer des outils d'estimation objectifs et fiables.

Ce rapport présente un projet visant à concevoir et déployer un système basé sur l'IA pour prédire les prix des voitures d'occasion au Maroc. L'approche adoptée repose sur plusieurs étapes clés : la collecte de données via web scraping, le nettoyage et l'analyse exploratoire des données, l'entraînement de modèles prédictifs, et le déploiement d'un produit minimum viable (MVP) sous forme d'une API et d'une extension de navigateur. Ce système vise à améliorer la transparence du marché en fournissant aux utilisateurs des estimations précises et accessibles, tout en réduisant l'incertitude associée aux transactions. Le rapport est structuré en six chapitres, couvrant le contexte du projet, la collecte et la préparation des données, l'analyse exploratoire, la sélection et l'entraînement des modèles, le déploiement, et les conclusions. À travers ce projet, nous démontrons comment l'IA peut transformer un marché traditionnel en un écosystème plus équitable et efficace.

CHAPITRE 1 : CONTEXTE GENERAL DU PROJET

1 Contexte General du Projet

1.1 Introduction

Au Maroc, le marché des voitures d'occasion est en pleine expansion, stimulé par l'accessibilité limitée aux véhicules neufs pour une large partie de la population. Les plateformes en ligne comme Avito.ma et Moteur.ma sont devenues des références pour l'achat et la vente de véhicules d'occasion. Cependant, l'estimation des prix reste souvent subjective, basée sur des comparaisons approximatives ou des évaluations empiriques.

Problématique :

L'absence d'outils fiables pour estimer objectivement la valeur des véhicules d'occasion au Maroc crée une asymétrie d'information sur le marché. Les vendeurs risquent de sous-évaluer ou surévaluer leurs véhicules, tandis que les acheteurs manquent de repères pour identifier les offres avantageuses. Cette situation génère de l'incertitude et peut ralentir les transactions.

1.2 Objectif général

Développer un système basé sur l'intelligence artificielle capable de prédire avec précision le prix des voitures d'occasion au Maroc en fonction de leurs caractéristiques techniques, leur état et les tendances du marché local.

1.3 Objectifs spécifiques

- Constituer une base de données représentative du marché automobile d'occasion marocain
- Identifier les facteurs déterminants influençant les prix des véhicules
- Construire un modèle prédictif performant exploitant ces facteurs
- Développer une interface utilisateur intuitive permettant d'obtenir des estimations rapides et fiables

1.4 Méthodologie globale

Notre approche se décompose en plusieurs étapes clés :

1 Collecte de données via le web Scraping de sites spécialisés

2 Préparation et nettoyage du jeu de données
3 Analyse exploratoire pour comprendre les distributions et relations entre variables
4 Développement et entraînement de modèles prédictifs
5 Optimisation et validation des performances du modèle 6. Déploiement sous forme d'application accessible.

1.5 Conclusion

Le Chapitre 2 a détaillé le processus de collecte de données à partir d'Avito.ma, en mettant en évidence l'utilisation de techniques de web scraping optimisées avec Python, BeautifulSoup, et une approche multi-threadée. La justification des choix technologiques et les mesures prises pour respecter l'étiquette du web ont permis de constituer un jeu de données riche et représentatif, couvrant plus de 2500 pages d'annonces. Ces données brutes, bien que nécessitant un nettoyage approfondi, forment la base nécessaire pour les analyses ultérieures. Le Chapitre 3 abordera la préparation et le nettoyage de ces données pour les rendre exploitables par les modèles prédictifs.

CHAPITRE 2 : COLLECTE DE DONNEES

2 Collecte de données

2.1 Introduction

La collecte de données constitue une étape fondamentale dans le développement d'un système de prédiction basé sur l'intelligence artificielle, car la qualité et la représentativité des données influencent directement les performances des modèles prédictifs. Ce chapitre décrit le processus de collecte des données à partir de la plateforme Avito.ma, choisie pour sa position dominante sur le marché des petites annonces au Maroc. Nous présentons les sources de données, la méthodologie de web scraping mise en œuvre, ainsi que les choix technologiques adoptés pour garantir une collecte efficace et respectueuse des contraintes éthiques et techniques. Ce chapitre pose les fondations pour les étapes ultérieures de préparation et d'analyse des données.

2.2 Description des sources de données

Pour notre étude, nous avons sélectionné Avito.ma comme source principale de données en raison de sa position dominante sur le marché des petites annonces au Maroc. Ce site regroupe un large éventail d'annonces de voitures d'occasion couvrant diverses marques, modèles, gammes de prix et régions géographiques, offrant ainsi une vue représentative du marché national.

2.3 Méthodologie de web Scraping

Le processus de collecte de données a été implémenté en Python en utilisant les bibliothèques suivantes :

- Requests : pour l'envoi de requêtes HTTP et la récupération du contenu des pages
- BeautifulSoup : pour l'analyse et l'extraction des données à partir du code HTML
- Pandas : pour la structuration et le stockage des données collectées
- ThreadPoolExecutor : pour paralléliser les requêtes et optimiser la vitesse de collecte

Notre script de Scraping implémente plusieurs fonctionnalités importantes :

- Rotation des User-Agents pour éviter la détection
- Gestion des erreurs et tentatives de reconnexion
- Délais aléatoires entre les requêtes pour respecter l'étiquette du web
- Sauvegarde progressive des données pour éviter les pertes en cas d'interruption

Les sélecteurs HTML ont été identifiés par une analyse préalable de la structure des pages d'annonces, ciblant les éléments contenant les informations pertinentes comme *le prix, la marque, le modèle, l'année, le kilométrage, et autres caractéristiques techniques.*

Pour consulter le code [cliquer ici]

url	year	type_boit	type_carburant	kilometrage	marke	model	puissance	premiere_main	Nombre_doors	city	price
tps://www.avito.ma/fr/ain_sebaa/voitures_d'o...	2021	Automatique	Essence	60 000 - 64 999	Fiat	500	7 CV	Oui	5.0	Ain Sebaa, Casablanca	159 000 DH
tps://www.avito.ma/fr/maarif/voitures_d'occa...	2025	Automatique	Diesel	0 - 4 999	BMW	Série 4	8 CV	Oui	3.0	Maarif, Casablanca	NaN
ps://www.avito.ma/fr/2_mars/voitures_de_loc...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2 Mars, Casablanca	200 DH
tps://www.avito.ma/fr/oulfa/voitures_d'occas...	2022	Automatique	Diesel	40 000 - 44 999	Volkswagen	Touareg	12 CV	Oui	5.0	Oulfa, Casablanca	NaN
tps://www.avito.ma/fr/maarif/voitures_d'occa...	2025	Automatique	Diesel	0 - 4 999	Audi	Q8	12 CV	NaN	5.0	Maarif, Casablanca	NaN
tps://www.avito.ma/fr/al_maghrib_al_arabi/vo...	2018	Manuelle	Diesel	150 000 - 159 999	Hyundai	Creta	6 CV	Oui	5.0	Al Maghrib Al Arabi, Kénitra	175 000 DH
tps://www.avito.ma/fr/ain_sebaa/voitures_d'o...	2021	Automatique	Diesel	95 000 - 99 999	Dacia	Duster	6 CV	Oui	5.0	Ain Sebaa, Casablanca	187 000 DH
ttps://www.avito.ma/fr/anfa/voitures_d'occasi...	2018	Automatique	Diesel	20 000 - 24 999	Mercedes-Benz	Classe V	9 CV	Oui	5.0	Anfa, Casablanca	845 000 DH
tps://www.avito.ma/fr/ain_sebaa/voitures_d'o...	2022	Manuelle	Diesel	60 000 - 64 999	Peugeot	208	6 CV	Oui	5.0	Ain Sebaa, Casablanca	168 000 DH
ps://www.avito.ma/fr/ain_amiyer/voitures_de...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Ain Amiyer, Fès	230 DH

FIGURE 1 : L'ENSEMBLE DE DONNEES AVANT LE PRETRAITEMENT

2.4 Justification des choix technologiques

Le choix de **Python** s'est imposé naturellement grâce à son écosystème riche en bibliothèques dédiées à la manipulation de données et au web Scraping.

BeautifulSoup a été préféré à d'autres alternatives comme **Scrapy** pour sa simplicité d'utilisation et sa flexibilité. L'implémentation multi-threadé nous a permis d'optimiser considérablement le temps de collecte tout en maintenant un comportement respectueux envers le serveur cible.

La collecte a été structurée en plusieurs sessions pour limiter la charge sur le serveur et éviter d'être bloqué. Nous avons ainsi parcouru plus de ***2500 pages***, représentant un volume significatif d'annonces de voitures d'occasion disponibles sur le marché marocain.

2.5 Conclusion

Le Chapitre 2 a détaillé le processus de collecte de données à partir d'Avito.ma, en mettant en évidence l'utilisation de techniques de web scraping optimisées avec Python, BeautifulSoup, et une approche

multi-threadée. La justification des choix technologiques et les mesures prises pour respecter l'étiquette du web ont permis de constituer un jeu de données riche et représentatif, couvrant plus de 2500 pages d'annonces. Ces données brutes, bien que nécessitant un nettoyage approfondi, forment la base nécessaire pour les analyses ultérieures. Le Chapitre 3 abordera la préparation et le nettoyage de ces données pour les rendre exploitables par les modèles prédictifs.

CHAPITRE 3 : PREPARATION DE DONNEES

3 Préparation de données

3.1 Introduction

La qualité d'un modèle prédictif repose en grande partie sur la préparation minutieuse des données, qui vise à éliminer les incohérences, les erreurs, et les biais présents dans les données brutes. Ce chapitre détaille les étapes de nettoyage et de prétraitement appliquées au jeu de données collecté à partir d'Avito.ma, afin de le rendre exploitable pour l'analyse exploratoire et l'entraînement des modèles. Nous abordons la suppression des valeurs manquantes, l'élimination des doublons, la standardisation des formats, la normalisation des valeurs textuelles, l'encodage des variables catégoriques, et le traitement des valeurs aberrantes. Ces opérations garantissent que les données sont cohérentes, fiables, et adaptées aux exigences des algorithmes d'apprentissage automatique.

3.2 Nettoyage des données

Le jeu de données brut a nécessité plusieurs opérations de nettoyage pour le rendre exploitable :

3.2.1 Suppression des valeurs manquantes :

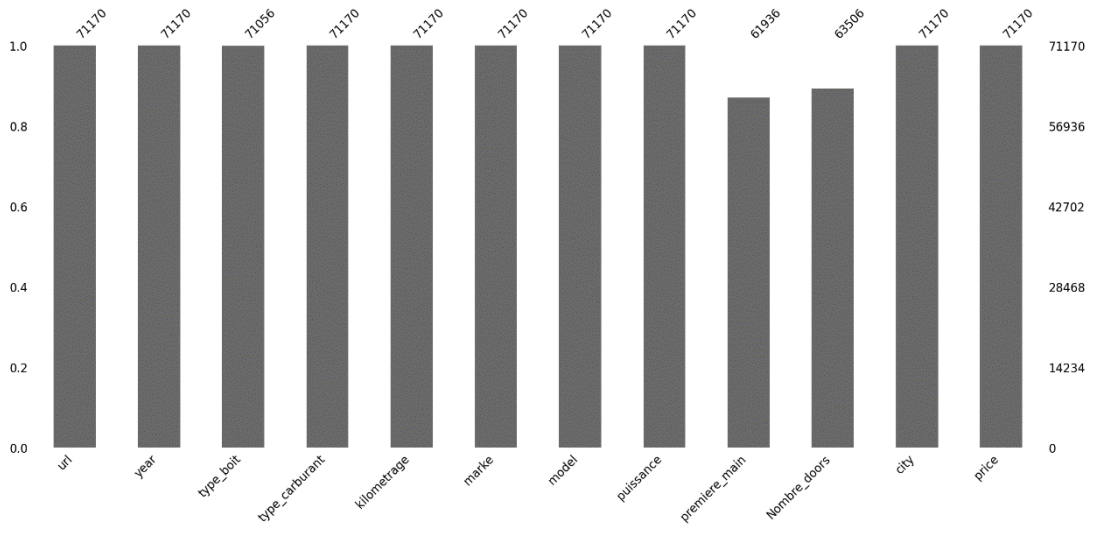

FIGURE 2 : NOMBRE DES VALEURS NON NULLES POUR CHAQUE VARIABLE

Nous avons éliminé les enregistrements incomplets concernant les variables critiques `(prix, type de boîte, type de carburant, kilométrage, marque, modèle, puissance)`.

3.2.2 Élimination des doublons :

Les annonces dupliquées ont été identifiées sur la base de l'URL unique et seule la première occurrence a été conservée.

3.2.3 Standardisation des formats :

- `Prix` : Extraction de la valeur numérique en supprimant le symbole "DH" et les espaces
- `Kilométrage` : Conversion des plages (ex: "100000-150000") en valeurs moyennes et suppression des préfixes comme "Plus de"
- `Puissance` : Extraction des valeurs numériques en éliminant "CV"
- `Année` : Conversion en format numérique

3.2.4 Normalisation des valeurs textuelles :

- Conversion en minuscules des variables catégorielles (type de carburant, type de boîte, marque, modèle)
- Standardisation des noms de villes, notamment conversion des noms en arabe vers leur équivalent français

3.2.5 Encodage des variables catégoriques :

Pour la variable "`première main`", nous avons appliqué un encodage ordinal :

- "Oui" → 2
- Valeur manquante → 1
- "Non" → 0

Cette approche préserve l'information ordinale implicite dans cette caractéristique.

3.3 Traitement des valeurs aberrantes

L'identification des valeurs aberrantes a été réalisée via l'analyse visuelle des *boxplots* et l'application du score Z. Nous avons appliqué un seuil de *Z-score* de 3 pour identifier les outliers, ce qui correspond approximativement aux valeurs situées à plus de 3 écarts-types de la moyenne.

Des filtres spécifiques ont également été appliqués pour le prix :

- Élimination des véhicules dont le prix est inférieur à 25 000 DH (potentiellement des erreurs ou des véhicules non fonctionnels)
- Suppression des annonces avec des prix supérieurs à 1 000 000 DH (segment de luxe non représentatif du marché général)

Ces opérations de nettoyage ont réduit le jeu de données à ***59 150 entrées*** valides sur les 13 variables collectées.

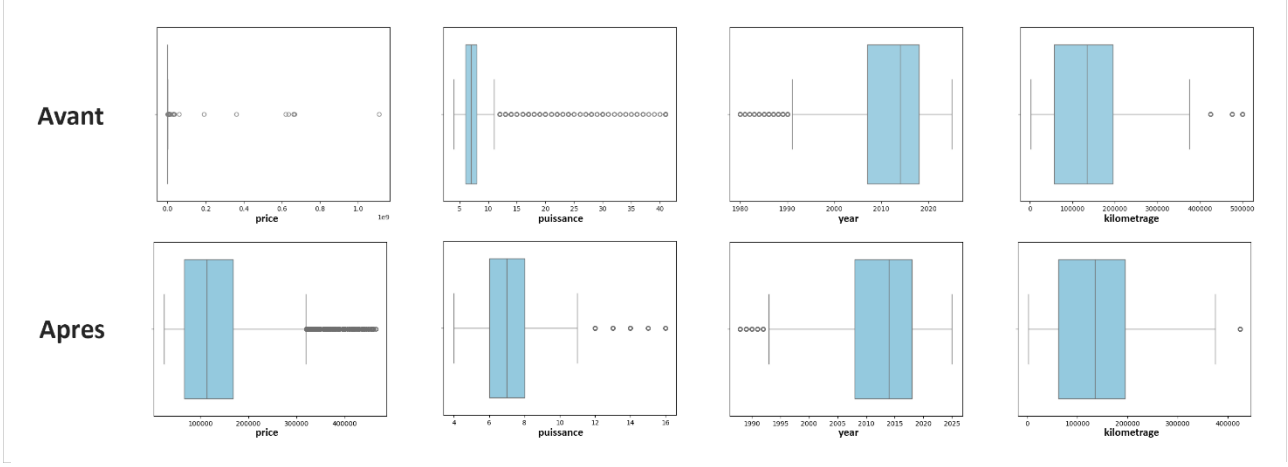

FIGURE 3 : LISTE DES BOXPLOTS AVANT ET APRES TRAITEMENT DES VALEURS ABERRANTES

Pour consulter le code [cliquer ici]

3.4 Conclusion

Le Chapitre 3 a décrit les étapes rigoureuses de préparation des données, transformant un jeu de données brut en une base propre et structurée de 59 150 entrées valides. Les opérations de nettoyage, incluant la suppression des valeurs manquantes et des doublons, la standardisation des formats, et le traitement des valeurs aberrantes, ont permis d'améliorer la qualité des données tout en préservant leur représentativité. L'encodage des variables catégoriques et la normalisation des valeurs textuelles ont préparé le terrain pour une analyse approfondie. Le Chapitre 4 s'appuiera sur ces données nettoyées pour réaliser une analyse exploratoire, visant à identifier les tendances et les relations clés influençant les prix des voitures d'occasion.

CHAPITRE 4 : ANALYSE EXPLORATOIRE DES DONNEES (EDA)

4 Analyse Exploratoire des Données (EDA)

4.1 Introduction

L'analyse exploratoire des données (EDA) est une étape essentielle pour comprendre les caractéristiques du jeu de données et identifier les facteurs influençant les prix des voitures d'occasion. Ce chapitre présente une analyse détaillée des données nettoyées, en examinant la distribution des variables clés, les relations entre elles, et les tendances spécifiques au marché marocain. À travers des visualisations telles que des histogrammes, des boxplots, des matrices de corrélation, et des diagrammes bivariés, nous mettons en évidence les dynamiques du marché, comme la prédominance des boîtes manuelles ou la forte représentation de certaines marques. Cette analyse guide la sélection des caractéristiques et la construction des modèles prédictifs dans les chapitres suivants.

4.2 Distribution des variables clés

Prix :

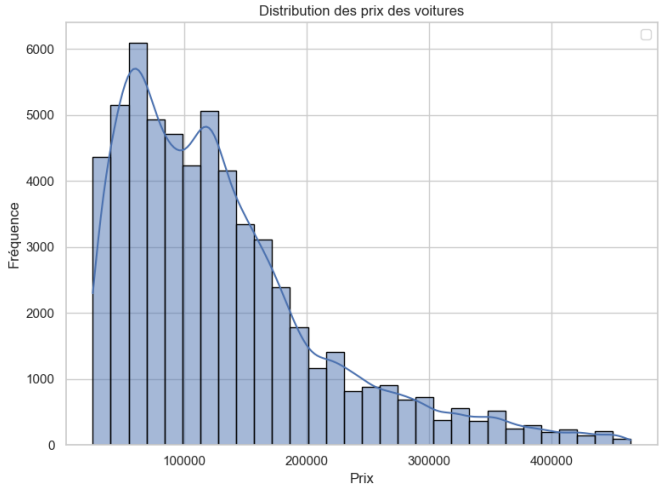

FIGURE 4 : DISTRIBUTION DES PRIX DES VEHICULES D'OCCASION

La distribution des prix présente une asymétrie positive (skewness), avec une concentration importante dans les segments économiques et moyens, et une queue qui s'étend vers les prix élevés.

Puissance fiscale :

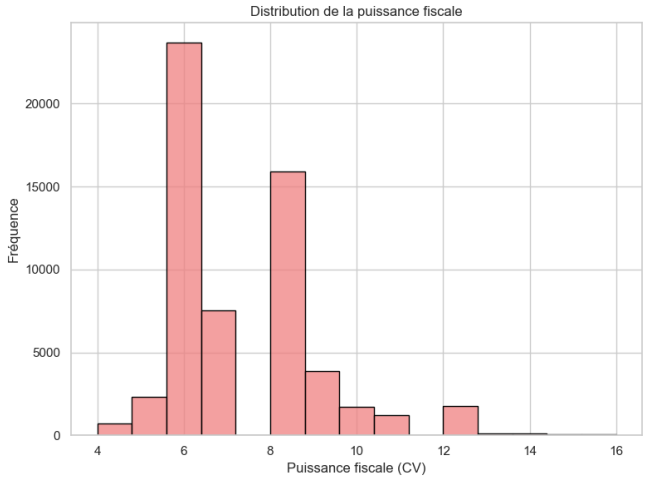

FIGURE 5 : DISTRIBUTION DE LA PUISSANCE FISCALE (EN CV)

Cette variable suit une distribution multimodale, avec des pics autour des valeurs courantes (6-8 CV), correspondant aux segments les plus populaires sur le marché marocain.

Type de boîte de vitesses :

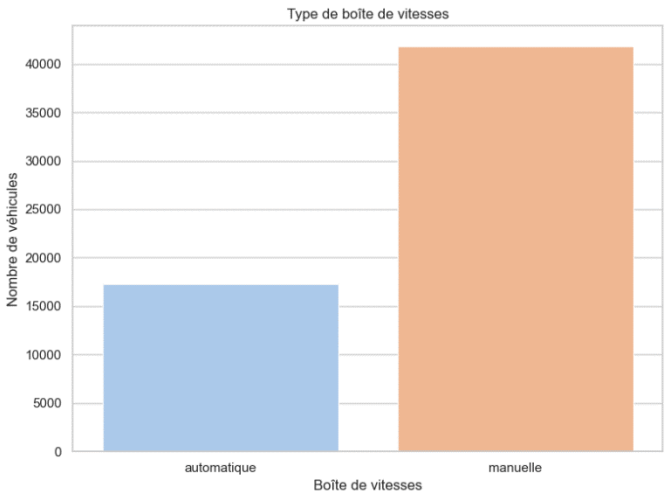

FIGURE 6 : REPARTITION DES TYPES DE BOITES DE VITESSES

L'analyse révèle une prédominance des boîtes manuelles (environ 80%) par rapport aux automatiques (20%), reflétant les préférences traditionnelles du marché marocain.

Type de carburant :

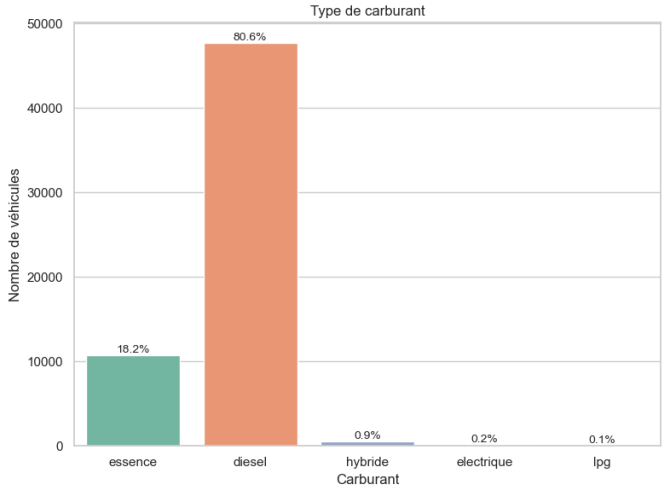

FIGURE 7 : REPARTITION DES TYPES DE CARBURANT UTILISES

La répartition montre une majorité de véhicules diesel, suivis par l'essence, avec une présence marginale des motorisations hybrides et électriques.

Marques et modèles :

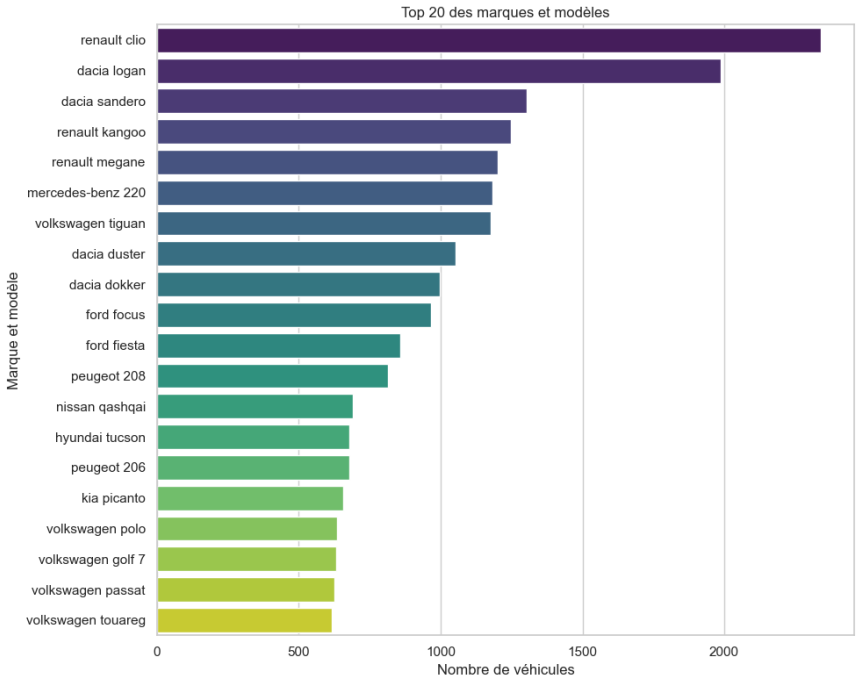

FIGURE 8 : TOP 20 DES MODELES DE VEHICULES LES PLUS REPRESENTES SUR LE MARCHE DE L'OCCASION

L'examen des 20 modèles les plus fréquents dans le jeu de données met en évidence une prédominance marquée de la **Renault Clio**, suivie de la **Dacia Logan**, ces deux modèles totalisant à eux seuls une part significative du parc analysé. Les marques **Renault** et **Dacia** occupent une position dominante, avec plusieurs modèles figurant dans le classement (Sandero, Kangoo, Mégane, Duster, Dokker), traduisant leur forte implantation sur le marché de l'occasion au Maroc.

Par ailleurs, d'autres constructeurs tels que **Volkswagen**, **Ford**, **Peugeot** et **Hyundai** sont également bien représentés, avec des modèles populaires comme la Volkswagen Tiguan, la Ford Fiesta, la Peugeot 208 ou encore le Hyundai Tucson. Cette concentration autour de marques européennes et coréennes reflète les tendances actuelles du marché local, marqué par une préférence pour des véhicules accessibles, robustes et bien adaptés à un usage urbain ou périurbain.

4.3 Analyse des relations entre variables :

4.3.1 Matrice de corrélation

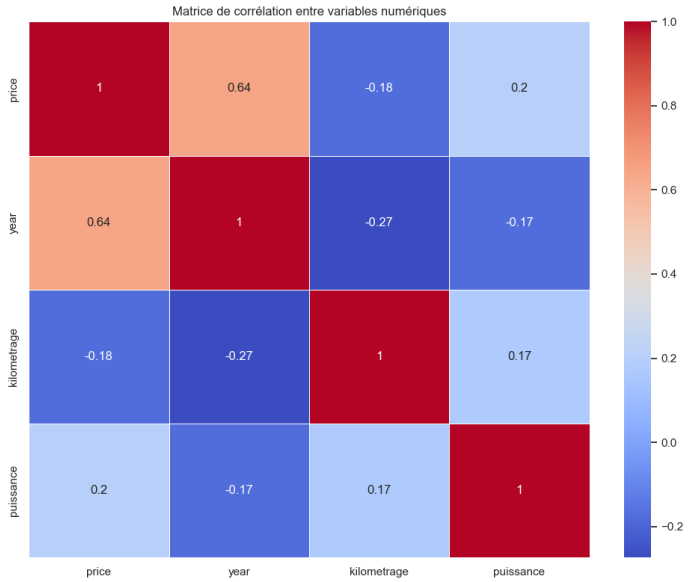

FIGURE 9 : MATRICE DE CORRELATION ENTRE VARIABLES NUMERIQUES

L'analyse des corrélations a mis en évidence plusieurs relations significatives :

- Corrélation positive forte entre le prix et l'année de fabrication
- Corrélation positive entre le prix et la puissance fiscale
- Corrélation négative entre le prix et le kilométrage

4.3.2 Visualisations bivariées

Prix vs Année de fabrication

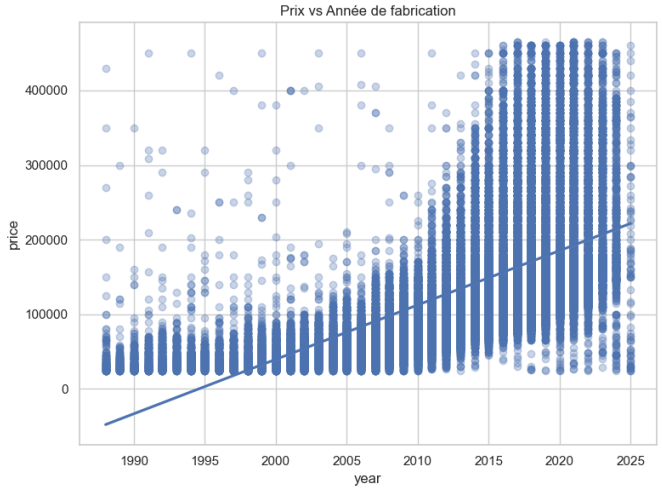

FIGURE 10 : RELATION ENTRE LE PRIX ET L'ANNEE DE FABRICATION

Les nuages de points avec régression linéaire ont confirmé la relation positive entre le prix et l'année de fabrication, **plus l'année est récente, plus le prix tend à augmenter**, ce qui reflète un comportement attendu sur le marché de l'occasion, où l'usure et l'ancienneté impactent directement la valeur perçue du véhicule.

Prix vs Kilométrage

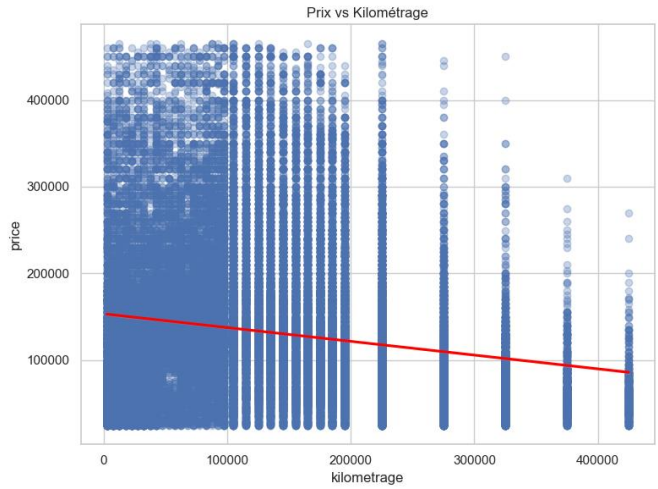

FIGURE 11 : RELATION ENTRE LE PRIX ET LE KILOMETRAGE

Le nuage de points avec régression met en évidence une **corrélation négative entre le kilométrage et le prix des véhicules**. De manière générale, **plus le kilométrage est élevé, plus le prix tend à diminuer**, ce qui reflète un comportement attendu sur le marché de l'occasion, où l'usure et l'ancienneté impactent directement la valeur perçue du véhicule.

La **régression linéaire** (en rouge) confirme cette tendance, bien que la dispersion des points indique une **relation non strictement linéaire** : certains véhicules à fort kilométrage conservent des prix élevés, probablement en raison d'autres facteurs comme la marque, le type de motorisation ou l'année de fabrication. Inversement, de nombreux véhicules faiblement kilométrés se positionnent dans des fourchettes de prix très variées, soulignant l'influence de variables additionnelles sur la valorisation finale.

Prix vs Puissance fiscale

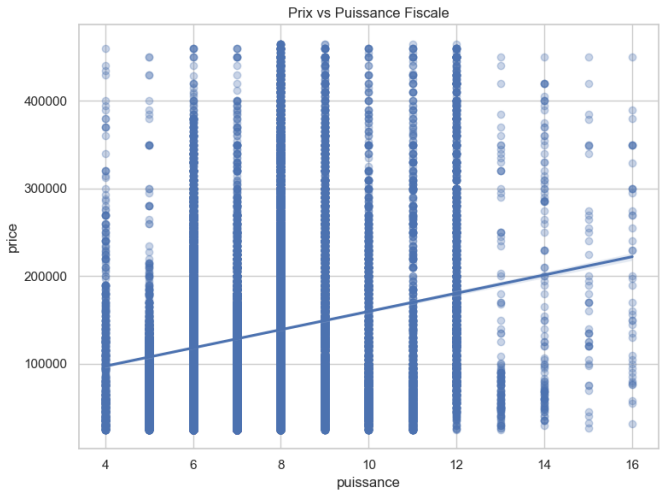

FIGURE 12 : RELATION ENTRE LE PRIX ET LA PUISSANCE FISCALE

La puissance fiscale a montré une corrélation positive avec le prix, particulièrement prononcée dans les segments supérieurs

4.3.3 Analyses segmentées

Violon - Prix selon le type de carburant

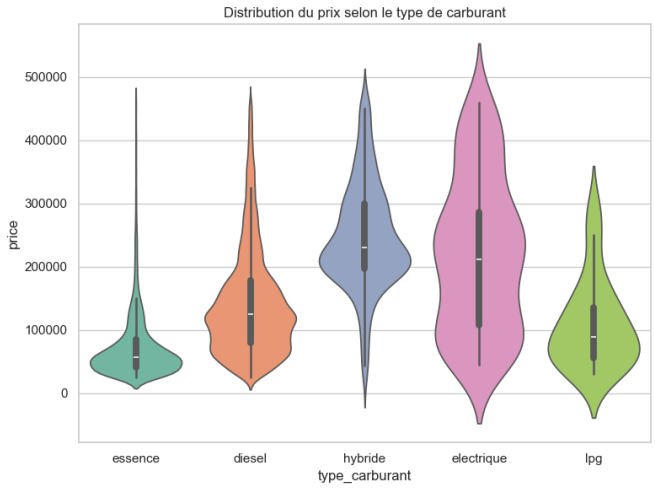

FIGURE 13 : DISTRIBUTION DES PRIX SELON LE TYPE DE CARBURANT

Les véhicules diesel dominent le marché, mais les motorisations alternatives présentent des variations de prix notables.

Violon - Prix selon le type de boîte de vitesses

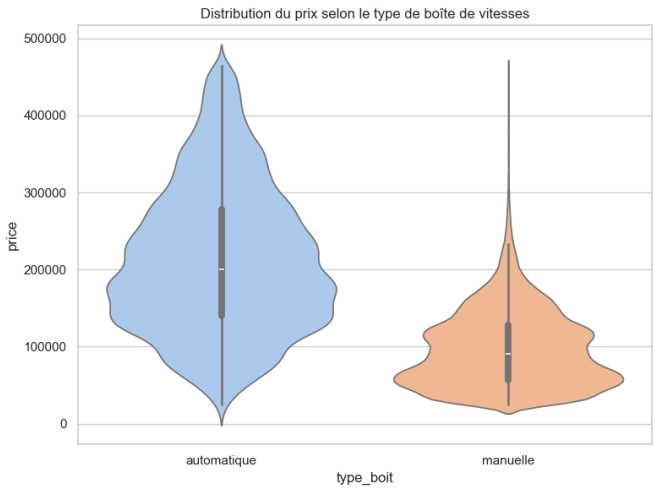

FIGURE 14 : DISTRIBUTION DES PRIX SELON LE TYPE DE BOITE DE VITESSES

Les véhicules à boîte automatique affichent des prix médianement plus élevés.

Les diagrammes en violon ont révélé des différences significatives de distribution des prix selon le type de carburant et de boîte de vitesses, avec des prix généralement plus élevés pour les véhicules à boîte automatique.

Pairplot segmenté par carburant

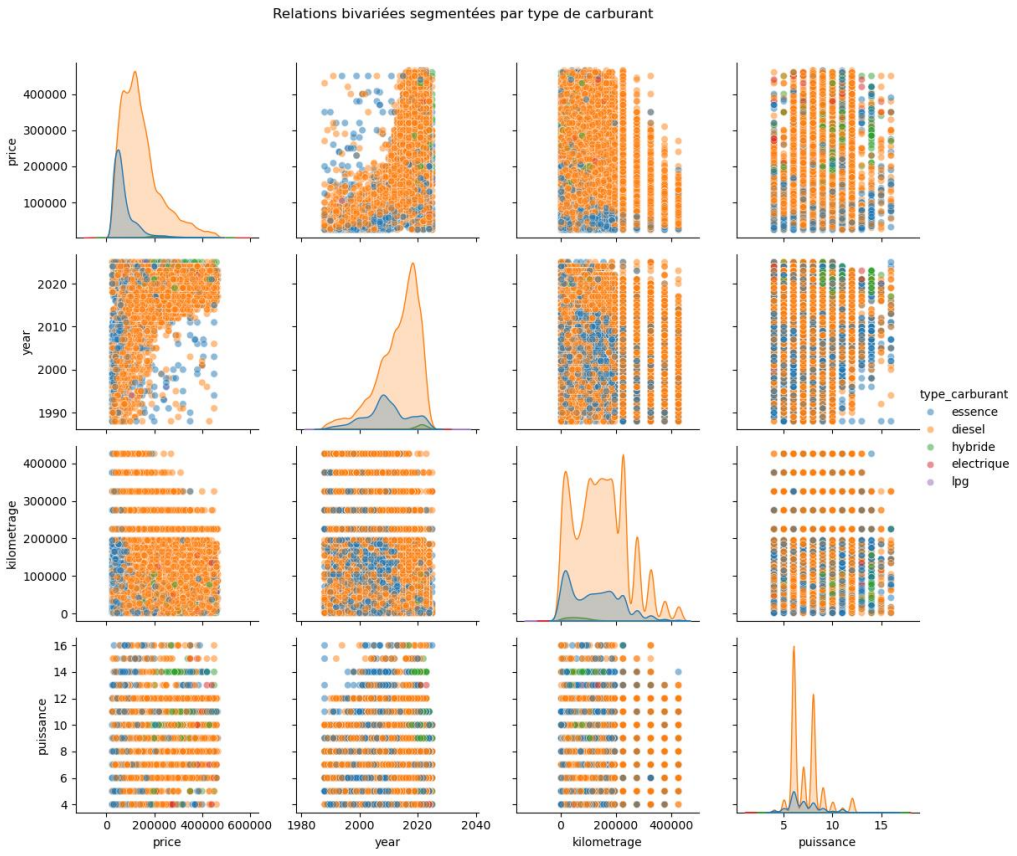

FIGURE 15 : RELATIONS BIVARIEES ENTRE VARIABLES SEGMENTEES PAR TYPE DE CARBURANT

Les pairplots segmentés par type de carburant ont montré des comportements distincts pour les véhicules diesel et essence, notamment en termes de relation entre *kilométrage et prix.*

Ces analyses exploratoires ont permis d'identifier les facteurs les plus influents sur le prix des véhicules d'occasion et de confirmer certaines hypothèses concernant le comportement du marché marocain.

4.4 Conclusion

L'analyse exploratoire des données (**EDA**) est une étape essentielle pour comprendre les caractéristiques du jeu de données et identifier les facteurs influençant les prix des voitures d'occasion. Ce chapitre présente une analyse détaillée des données nettoyées, en examinant la distribution des variables clés, les relations entre elles, et les tendances spécifiques au marché marocain. À travers des visualisations telles que des histogrammes, des boxplots, des matrices de corrélation, et des diagrammes bivariés, nous mettons en évidence les dynamiques du marché, comme la prédominance des boîtes manuelles ou la forte représentation de certaines marques. Cette analyse guide la sélection des caractéristiques et la construction des modèles prédictifs dans les chapitres suivants.

CHAPITRE 5 : SELECTION ET ENTRAINEMENT DE MODELES

5 Évaluation et Expérimentation

5.1 Introduction

Dans ce chapitre, nous abordons l'étape cruciale de la sélection et de l'entraînement des modèles prédictifs pour estimer les prix des voitures d'occasion au Maroc. Cette phase s'appuie sur les données nettoyées et analysées dans les chapitres précédents, en mettant l'accent sur la construction de pipelines de prétraitement robustes et l'évaluation de plusieurs algorithmes d'apprentissage automatique. Notre objectif est d'identifier le modèle le plus performant en termes de précision, de généralisation et de robustesse, tout en optimisant les hyperparamètres pour maximiser les performances prédictives.

5.2 Approche méthodologique

Notre démarche de modélisation s'est articulée autour d'une stratégie progressive et itérative, visant à identifier la configuration optimale pour prédire avec précision les prix des véhicules d'occasion sur le marché marocain. Cette approche s'est déclinée en trois versions distinctes, chacune présentant un raffinement spécifique dans la sélection des caractéristiques et le paramétrage des modèles.

5.2.1 Sélection des variables prédictives

L'efficacité d'un modèle prédictif repose fondamentalement sur la pertinence des variables explicatives utilisées. Notre processus de sélection des variables s'est déroulé en trois versions progressives :

Version 1 : Caractéristiques fondamentales

- Variables utilisées : `année`, `puissance`, `kilométrage`, `type_boîte`, `type_carburant`, `marque`, `modèle`
- Cette version minimaliste se concentre sur les attributs techniques essentiels du véhicule, offrant une base comparative pour les itérations suivantes.

Version 2 : Élargissement contextuel

- Variables utilisées : `année`, `type_boîte`, `type_carburant`, `kilométrage`, `marque`, `modèle`, `puissance`, `première_main`, `nombre_portes`, `ville`
- Cette version enrichit le modèle avec des caractéristiques additionnelles susceptibles d'influencer la valorisation du véhicule, notamment la localisation géographique et les aspects qualitatifs comme l'historique de propriété.

Version 3 : Raffinement sélectif

- Variables utilisées : année, type_boîte, type_carburant, kilométrage, marque, modèle, puissance, première_main, nombre_portes
- Cette version représente un compromis entre complexité et pertinence, excluant spécifiquement la variable ville qui introduisait potentiellement du bruit dans les prédictions.

5.2.2 Prétraitement des données

Pour garantir la robustesse et l'efficacité des modèles, nous avons mis en œuvre un pipeline de prétraitement standardisé comprenant :

- **Standardisation des variables numériques** via `StandardScaler`, permettant de ramener les variables à une échelle comparable et d'améliorer la convergence des algorithmes d'optimisation
- **Encodage des variables catégorielles** par `OneHotEncoder` avec l'option `handle_unknown='ignore'`, facilitant la gestion des catégories nouvelles ou rares lors des prédictions
- **Transformation des données structurées** par `ColumnTransformer`, assurant un traitement différencié et approprié pour chaque type de variable

Ce pipeline garantit la cohérence du prétraitement entre les phases d'entraînement et de prédiction, élément crucial pour l'exploitation du modèle en production.

Pour consulter le code [cliquer ici]

5.3 Modèles d'apprentissage automatique

Notre stratégie d'exploration algorithmique s'est orientée vers quatre modèles de régression aux caractéristiques complémentaires :

5.3.1 Présentation des algorithmes sélectionnés

1. **Arbre de décision (DecisionTreeRegressor)**
   - Modèle non paramétrique capable de capturer des relations non linéaires complexes
   - Interprétabilité directe des règles de décision

- Sensibilité au surapprentissage nécessitant une attention particulière au paramétrage

2. **Forêt aléatoire (RandomForestRegressor)**
   - Ensemble d'arbres de décision entraînés sur des sous-échantillons aléatoires
   - Robustesse accrue face au surapprentissage grâce à l'agrégation des prédictions
   - Capacité à mesurer l'importance relative des caractéristiques
3. **XGBoost (XGBRegressor)**
   - Implémentation optimisée de l'algorithme de boosting gradient
   - Performance généralement supérieure sur les problèmes de régression complexes
   - Flexibilité dans la gestion des fonctions de perte et des métriques d'évaluation
4. **Ridge (RidgeRegressor)**
   - Variante régularisée de la régression linéaire
   - Réduction du risque de surapprentissage par pénalisation L2
   - Modèle de référence pour évaluer la présence de non-linéarités significatives

5.3.2 Méthodologie d'évaluation

Pour garantir une évaluation rigoureuse et objective des performances, nous avons implémenté un protocole d'évaluation multi-critères :

- **Partitionnement des données** : Division en ensembles d'entraînement (80%) et de test (20%)
- **Validation croisée** : K-fold avec k=10 pour estimer la stabilité des performances
- **Métriques d'évaluation** :
  - R^2 (coefficient de détermination)
  - R^2 ajusté (prenant en compte le nombre de variables)
  - R^2 cross-validé (moyenne sur les 10 folds)
  - RMSE (Root Mean Square Error), pour quantifier l'erreur de prédiction en unités monétaires (dirhams)

Cette méthodologie permet d'évaluer non seulement la précision des prédictions, mais également leur fiabilité et la capacité de généralisation des modèles.

5.4 Résultats comparatifs

Les performances des différents modèles à travers les trois versions de sélection de caractéristiques sont synthétisées dans le tableau suivant :

Version	Modèle	R^2	R^2 ajusté	R^2 cross-validé	RMSE
Version 1	Arbre de décision	0.879	0.871	0.866	27175.017
	Forêt aléatoire	0.909	0.902	0.903	23640.588
	XGBoost	0.907	0.900	0.905	23845.559
	Ridge	0.818	0.806	0.819	33333.345
Version 2	Arbre de décision	0.853	0.838	0.851	32553.599
	Forêt aléatoire	0.907	0.897	0.905	25894.937
	XGBoost	0.904	0.894	0.902	26317.599
	Ridge	0.809	0.790	0.806	37052.212
Version 3	Arbre de décision	0.880	0.839	0.854	32856.404
	Forêt aléatoire	0.902	0.895	0.901	26520.342
	XGBoost	0.903	0.896	0.902	26402.974
	Ridge	0.820	0.795	0.805	37102.789

5.4.1 Analyse des performances

Plusieurs observations significatives se dégagent de cette analyse comparative :

1. **Supériorité des modèles ensemblistes** : Les algorithmes de forêt aléatoire et XGBoost surpassent systématiquement l'arbre de décision simple et la régression Ridge, avec un R^2 dépassant 0.90 dans toutes les versions.
2. **Impact de la sélection de caractéristiques** :
   - **La Version 1**, malgré sa simplicité, obtient d'excellents résultats, suggérant que les variables fondamentales capturent déjà l'essentiel de la variance des prix.

- L'introduction de la variable `ville` dans la **Version 2** n'améliore pas significativement les performances et semble même dégrader légèrement certaines métriques, probablement en raison du bruit introduit.
- **La Version 3** représente un bon compromis, avec des performances proches de la Version 1 mais intégrant des variables additionnelles potentiellement utiles pour des cas spécifiques.

3. **Stabilité des modèles** : La faible différence entre le R^2 et le R^2 cross-validé (généralement inférieure à 0.01) témoigne d'une bonne stabilité des modèles face à différents sous-ensembles de données.
4. **Précision monétaire** : Les meilleurs modèles affichent une RMSE d'environ 23 000 - 26 000 dirhams, ce qui représente une marge d'erreur acceptable sur le marché automobile d'occasion marocain.

5.4.2 Visualisation des Performances

Les performances des modèles ont été visualisées à l'aide de graphiques en barres horizontales, illustrant les métriques R^2, R^2 ajusté, R^2 validé et RMSE pour chaque modèle et version du pipeline. Ces visualisations, générées via la fonction `evaluate_and_visualize_models`, ont permis de comparer visuellement les différences entre les versions et d'identifier les modèles les plus performants.

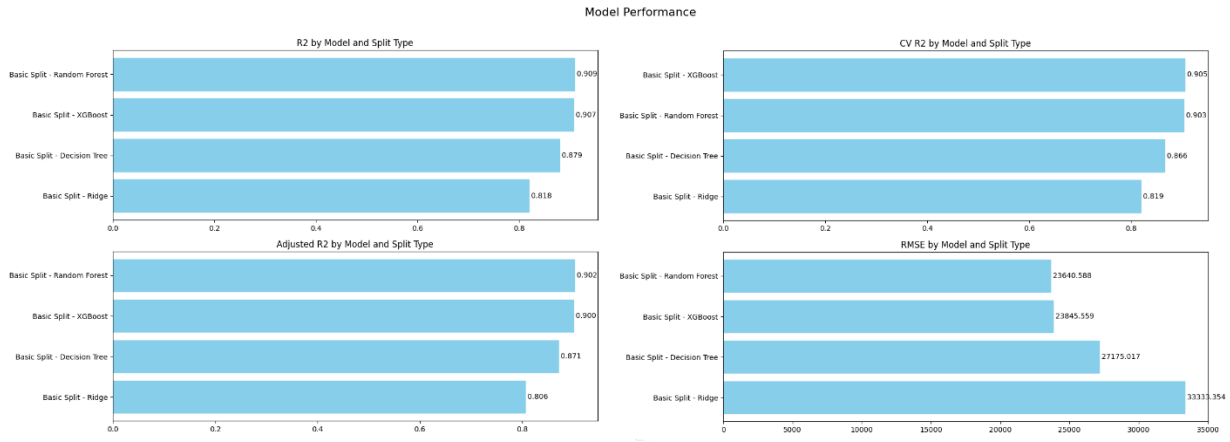

FIGURE 16 : PERFORMANCES DES MODELES POUR LA VERSION 1

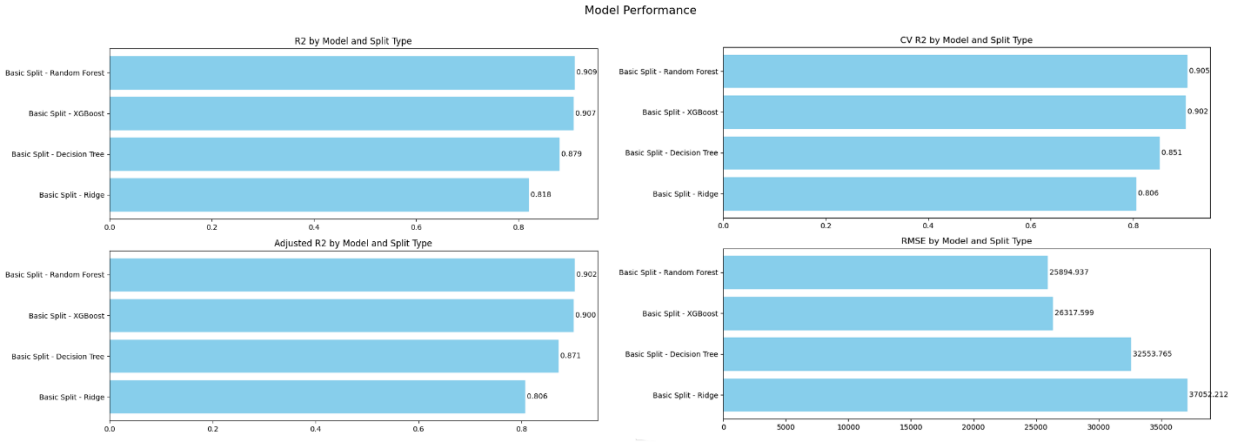

FIGURE 17 : PERFORMANCES DES MODELES POUR LA VERSION 2.

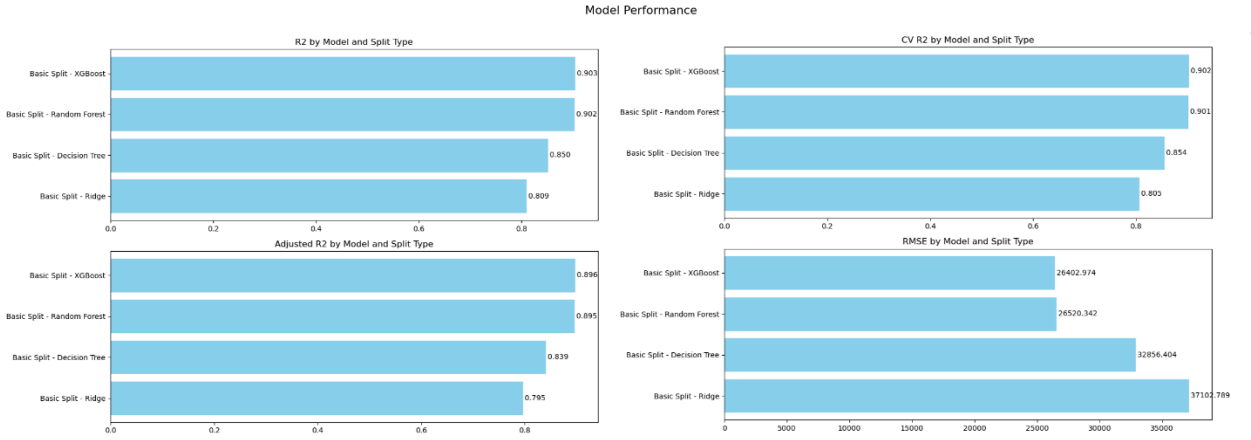

FIGURE 18 : PERFORMANCES DES MODELES POUR LA VERSION 3.

Les graphiques confirment que la Version 3 offre les meilleures performances globales, avec des valeurs élevées de R^2 et des RMSE modérés pour les modèles Random Forest et XGBoost.

5.4.3 Sélection du modèle final

Sur la base de ces résultats, nous avons retenu la **Forêt aléatoire de la Version 1** comme modèle principal pour notre système de prédiction, avec les justifications suivantes :

- **Performance optimale** : Meilleur R^2 (0.909) et RMSE la plus faible (23640.588 DH)
- **Robustesse** : Excellente stabilité entre validation simple et croisée (écart de 0.006)
- **Parsimonie** : Utilisation d'un nombre réduit de variables, facilitant la maintenance et le déploiement

- **Interprétabilité** : Possibilité d'extraire l'importance relative des caractéristiques pour l'explicabilité des prédictions

Néanmoins, pour garantir une évaluation continue des performances, nous conservons également l'implémentation du modèle XGBoost de la Version 1 comme alternative, celui-ci offrant des performances très proches avec un potentiel d'optimisation supplémentaire.

5.5 Sauvegarde et exportation des modèles

Pour faciliter le déploiement et garantir la reproductibilité des prédictions, nous avons exporté les éléments suivants à l'aide de la bibliothèque `joblib` :

- **Pipeline de prétraitement** (`preprocessor.joblib`) : Encapsulant toutes les transformations nécessaires pour les nouvelles données

Modèles entraînés : Sauvegardés individuellement au format pickle (`.pkl`) avec des noms explicites

- **Métadonnées associées** : Informations sur les variables utilisées et leurs caractéristiques

Ce processus assure la cohérence entre les phases d'entraînement et d'inférence, élément critique pour la fiabilité du système en production.

5.6 Conclusion

Ce chapitre a démontré la faisabilité de développer un système prédictif performant pour l'estimation des prix des voitures d'occasion au Maroc. La Version 3 du pipeline, combinée aux modèles Random Forest ou XGBoost, constitue une base solide pour la phase de déploiement (Chapitre 6). Les prochaines étapes incluront l'optimisation des hyperparamètres, l'intégration des modèles dans une interface utilisateur, et la validation continue des performances sur de nouvelles données.

CHAPITRE 6 : DEPLOIEMENT DE MODEL ET CREATION D'UN MVP

6 Déploiement de model et création d'un MVP

6.1 Introduction

Ce chapitre décrit la phase de déploiement du modèle prédictif développé dans le Chapitre 5, ainsi que la création d'un produit minimum viable (MVP) pour valider l'approche proposée. L'objectif est de rendre le système de prédiction des prix des voitures d'occasion au Maroc accessible aux utilisateurs finaux, notamment les acheteurs et vendeurs sur des plateformes comme Avito.ma. Pour ce faire, nous avons implémenté une API RESTful basée sur FastAPI pour exposer le modèle prédictif et une extension de navigateur (Car Price Estimator Pro) pour offrir une interface utilisateur intuitive. Ces composants forment un MVP qui permet de tester la viabilité du système dans un contexte réel.

6.2 Architecture du Système

Le système déployé repose sur une architecture client-serveur, composée des éléments suivants :

1. **Backend (API)** : Une **API RESTful** développée avec **FastAPI**, qui charge le modèle Random Forest (sélectionné pour ses performances optimales dans le Chapitre 5) et le pipeline de prétraitement pour fournir des estimations de prix en réponse aux requêtes des clients.
2. **Frontend (Extension de Navigateur)** : Une extension Chrome appelée *Car Price Estimator Pro,* développée avec **React**, **TailwindCSS** et **Vite**, qui permet aux utilisateurs d'interagir avec l'API via une interface graphique intuitive.
3. **Modèle et Prétraitement** : Les fichiers `random_forest_model.joblib` et `preprocessor.joblib` sauvegardés dans le Chapitre 5, utilisés pour effectuer les prédictions.

Cette architecture garantit une séparation claire des responsabilités, une modularité et une évolutivité pour des développements futurs.

6.3 Implémentation du Backend

6.3.1 Configuration de l'API

L'API est construite avec **FastAPI**, un framework Python moderne et performant pour le développement **d'API RESTful**. Le fichier principal (main.py) configure l'application FastAPI avec les fonctionnalités suivantes :

- **CORS (Cross-Origin Resource Sharing)** : Configuré pour permettre des requêtes depuis n'importe quelle origine, facilitant l'intégration avec le frontend.
- **Sécurité par Clé API** : Une authentification par clé API est implémentée via `APIKeyHeader`. Les requêtes doivent inclure un en-tête **X-API-KEY** correspondant à la clé définie dans les variables d'environnement (fichier `.env`).
- **Documentation Automatique** : **FastAPI** génère une interface **Swagger** accessible via `/docs`, permettant aux développeurs de tester les endpoints.

6.3.2 Endpoints de l'API

L'API expose deux endpoints principaux :

1. **`POST /estimate` :**

- **Fonction :** Prédit le prix d'une voiture d'occasion en fonction des caractéristiques fournies.
- **Entrée :** Un objet JSON conforme au schéma **`CarData`** (défini dans schemas.py), contenant les caractéristiques du véhicule telles que l'année de fabrication, le type de boîte de vitesses, le type de carburant, le kilométrage, la marque, le modèle, la puissance, etc.
- **Sortie :** Un objet JSON conforme au schéma **`EstimationResult`**, contenant le prix estimé, le prix listé (si fourni), un score de confiance, la devise (MAD), et des détails sur le modèle utilisé.

Exemple de Requête :

```
{
  "year": 2015,
  "type_boit": "manuelle",
  "type_carburant": "diesel",
  "kilometrage": 80000,
  "marke": "toyota",
  "model": "corolla",
  "puissance": 110,
  "premiere_main": 1,
  "Nombre_doors": 5,
}
```

Exemple de Réponse :

```
{
  "listedPrice": 150000.0,
  "estimatedPrice": 145000.0,
  "confidence": 85.0,
  "currency": "MAD",
  "details": {
    "modelUsed": "RandomForestRegressor",
    "featuresUsed": ["year", "type_boit", "type_carburant", "kilometrage",
"marke", "model", "puissance", "premiere_main", "Nombre_doors"]
  }
}
```


2. **GET /health :**
   - **Fonction** : Vérifie l'état de l'API et la disponibilité du modèle.
   - **Sortie** : Un objet JSON indiquant l'état de santé (healthy) et si le modèle est chargé.

Exemple de Réponse :


```
{
  "status": "healthy",
  "model_loaded": true
}
```


6.3.3 Service de Prédiction

Le module prediction_service.py encapsule la logique de prédiction. La classe **PredictionService** est conçue comme un singleton pour gérer le chargement et l'utilisation du modèle et du pipeline de prétraitement. Les principales fonctionnalités incluent :

- **Chargement des Modèles** : Les fichiers `random_forest_model.joblib` et preprocessor.joblib sont chargés à l'aide de **joblib**. Le modèle et le préprocesseur sont stockés en mémoire pour éviter des rechargements coûteux.
- **Prétraitement des Données** : Les données d'entrée sont converties en un **DataFrame Pandas**, puis transformées par le pipeline de prétraitement pour correspondre au format attendu par le modèle.
- **Prédiction** : Le modèle Random Forest effectue la prédiction du prix en fonction des caractéristiques transformées.
- **Score de Confiance** : Un score de confiance est calculé en fonction de l'âge du véhicule et du kilométrage, avec une base de 90 % réduite par des facteurs liés à l'usure (0,5 % par an d'âge, 5 % par 100 000 km, avec un minimum de 50 %).

En cas d'absence du préprocesseur, une méthode de prétraitement manuelle `(_manual_preprocessing)` est utilisée comme solution de secours, bien que moins robuste.

6.3.4 Gestion des Erreurs

L'API gère les erreurs de manière robuste :

- **Erreurs d'Authentification** : Une clé API invalide déclenche une erreur HTTP 403.
- **Erreurs de Prédiction** : Les exceptions lors du traitement des données ou des prédictions sont capturées et renvoyées comme des erreurs HTTP 400 avec un message détaillé.
- **Modèle Non Chargé** : Si le modèle ou le préprocesseur ne peut être chargé, une erreur RuntimeError est levée.

6.4 Implémentation du Frontend

6.4.1 Car Price Estimator Pro

Le frontend est une extension Chrome nommée *Car Price Estimator Pro,* conçue pour offrir une expérience utilisateur fluide et intuitive. Elle permet aux utilisateurs d'obtenir des estimations de prix directement sur les annonces d'Avito.ma ou via une saisie manuelle des caractéristiques du véhicule.

6.4.2 Fonctionnalités Principales

- **Estimation Manuelle** : Les utilisateurs peuvent saisir manuellement les caractéristiques du véhicule (année, kilométrage, type de carburant, etc.) pour obtenir une estimation.
- **Extraction Automatique** : L'extension utilise les API d'extension Chrome pour extraire automatiquement les détails des annonces sur Avito.ma, réduisant l'effort requis de la part de l'utilisateur.
- **Score de Confiance** : Affiche un pourcentage de confiance pour indiquer la fiabilité de l'estimation.
- **Analyse des Offres** : Compare le prix listé à l'estimation pour déterminer si l'offre est avantageuse ou surévaluée.
- **Mode Autonome** : Une page autonome permet des estimations sans dépendre des annonces d'Avito.ma.

6.4.3 Technologies Utilisées

- **React** : Framework JavaScript pour la construction de l'interface utilisateur dynamique.
- **TailwindCSS** : Framework CSS pour un design épuré et responsive.
- **Vite** : Outil de build rapide pour le développement et la production.

- **Chrome Extension APIs** : Pour l'intégration avec le navigateur et l'extraction de données des pages web.
- **Headless UI** : Pour des composants d'interface accessibles et personnalisables.

6.4.4 Intégration avec l'API

L'extension communique avec l'API via des requêtes HTTP POST vers ***l'endpoint /estimate***. Le fichier *src/services/apiService.js* gère les appels API, en s'assurant que les données sont normalisées avant l'envoi et que les réponses sont correctement interprétées. Les erreurs API sont affichées à l'utilisateur de manière claire.

6.5 Guide d'utilisation du MVP

Le MVP (Minimum Viable Product) déployé offre une expérience utilisateur complète pour l'estimation des prix des véhicules d'occasion sur le marché marocain.

6.5.1 Utilisation de l'extension navigateur

Installation :

1. Télécharger l'extension depuis le dépôt GitHub
2. Ouvrir Chrome et naviguer vers chrome://extensions/
3. Activer le mode développeur
4. Cliquer sur "Charger l'extension non empaquetée" et sélectionner le dossier dist

Estimation sur Avito.ma :

1. Naviguer vers une annonce de voiture sur Avito.ma
2. Cliquer sur l'icône de l'extension dans la barre d'outils
3. Sélectionner "Estimation site web" pour extraire automatiquement les données de l'annonce
4. Consulter l'estimation de prix, le score de confiance et l'analyse de l'offre

Estimation manuelle :

- Cliquer sur l'icône de l'extension
- Sélectionner "Estimation manuelle"
- Remplir les caractéristiques du véhicule
- Cliquer sur "Estimer" pour obtenir l'évaluation

6.5.2 Interprétation des résultats

L'extension fournit plusieurs indicateurs pour faciliter la prise de décision :

- **Prix estimé** : Valeur prédite par le modèle en dirhams marocains
- **Score de confiance** : Pourcentage indiquant la fiabilité de l'estimation
- **Analyse de l'offre** : Classification en "Bonne affaire", "Prix correct" ou "Surévalué".

6.6 Validation du MVP

Le MVP a été testé dans un environnement simulé pour évaluer sa fonctionnalité et son utilité. Les tests ont inclus :

- **Tests d'API** : Validation des réponses de l'endpoint `/estimate` avec des données variées, y compris des cas extrêmes (véhicules très anciens ou à fort kilométrage).
- **Tests d'Extension** : Vérification de l'extraction automatique sur Avito.ma et de la saisie manuelle, avec un accent sur la convivialité et la précision des estimations.
- **Performance** : Évaluation des temps de réponse de l'API (généralement < 1 seconde) et de l'extension (interface réactive).

Les résultats ont montré que le MVP répond aux objectifs principaux : fournir des estimations rapides, fiables et accessibles, avec une interface intuitive pour les utilisateurs non techniques.

6.7 Limites actuelles

Bien que fonctionnel et utile dans sa version actuelle, notre MVP présente certaines limitations qui feront l'objet d'améliorations dans les versions futures :

- **Couverture du marché** : Focalisation actuelle sur Avito.ma, excluant d'autres plateformes d'annonces
- **Variabilité géographique** : Prise en compte limitée des spécificités régionales influençant les prix
- **Facteurs qualitatifs** : Absence d'intégration des aspects visuels (photos) dans l'évaluation
- **Évolution temporelle** : Nécessité d'un mécanisme de mise à jour régulière du modèle pour refléter les tendances du marché

6.8 Conclusion

Ce chapitre a décrit le déploiement du modèle prédictif sous forme d'une *API FastAPI* et d'une extension *Chrome*, formant un MVP fonctionnel pour l'estimation des prix des voitures d'occasion au Maroc. Le système répond aux besoins des acheteurs et vendeurs en offrant des estimations rapides, fiables et accessibles directement sur Avito.ma. Bien que des limites subsistent, le MVP constitue une preuve de concept solide, ouvrant la voie à des améliorations futures pour un déploiement à grande échelle.

Figures

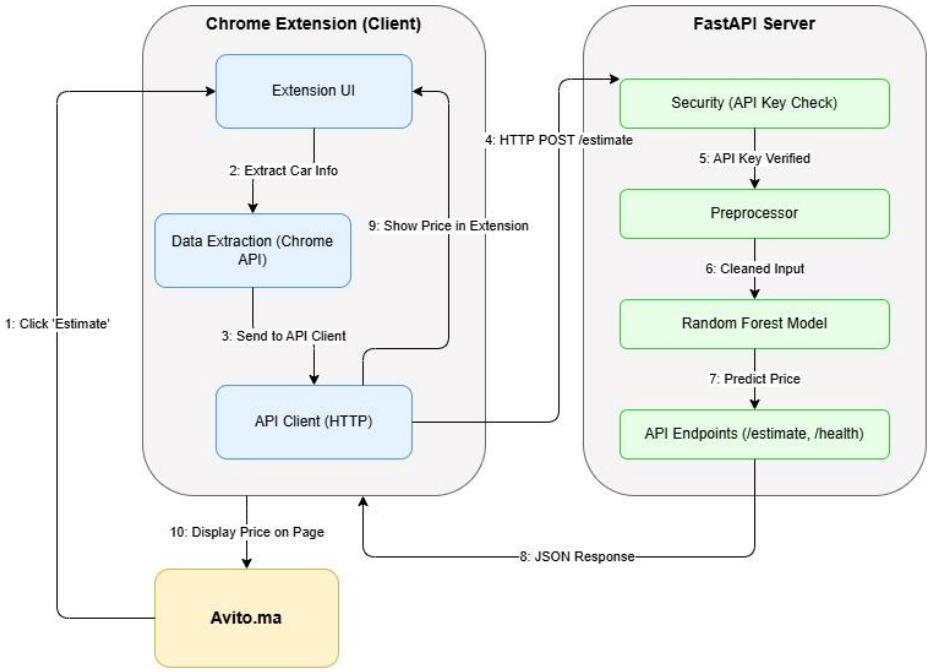

FIGURE 19 : ARCHITECTURE DU SYSTEME CLIENT-SERVEUR (API ET EXTENSION).

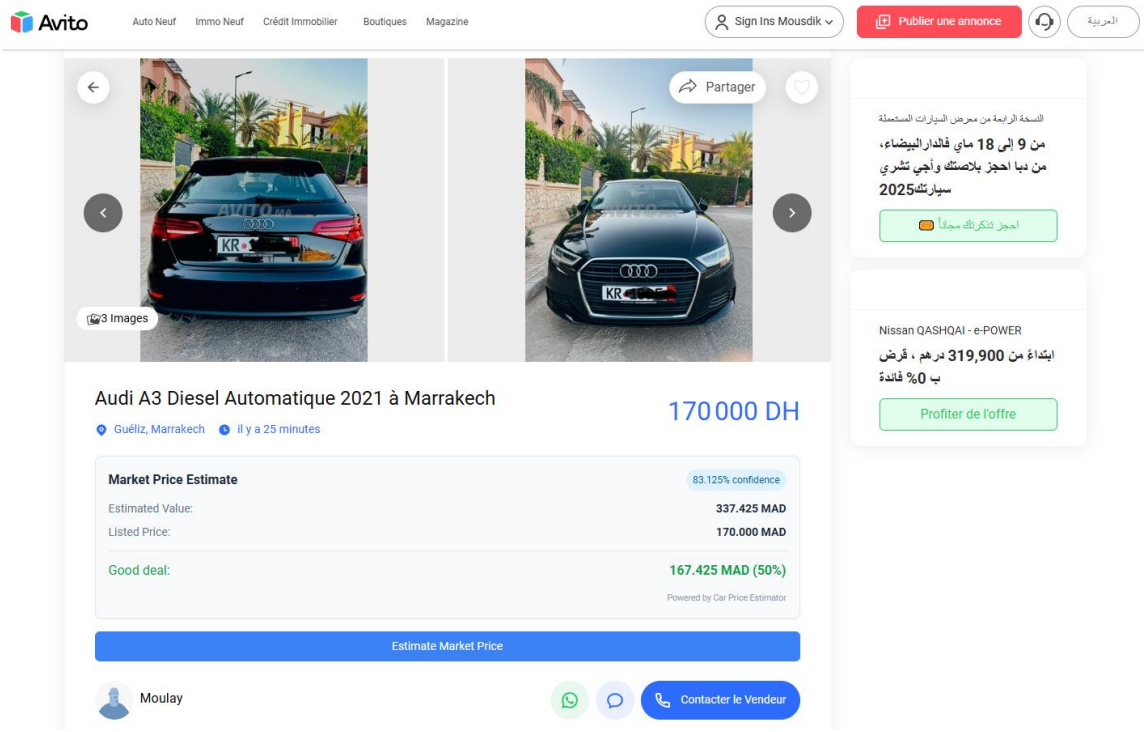

FIGURE 20 : INTERFACE DE L'EXTENSION *CAR PRICE ESTIMATOR PRO* SUR UNE ANNONCE AVITO.MA.

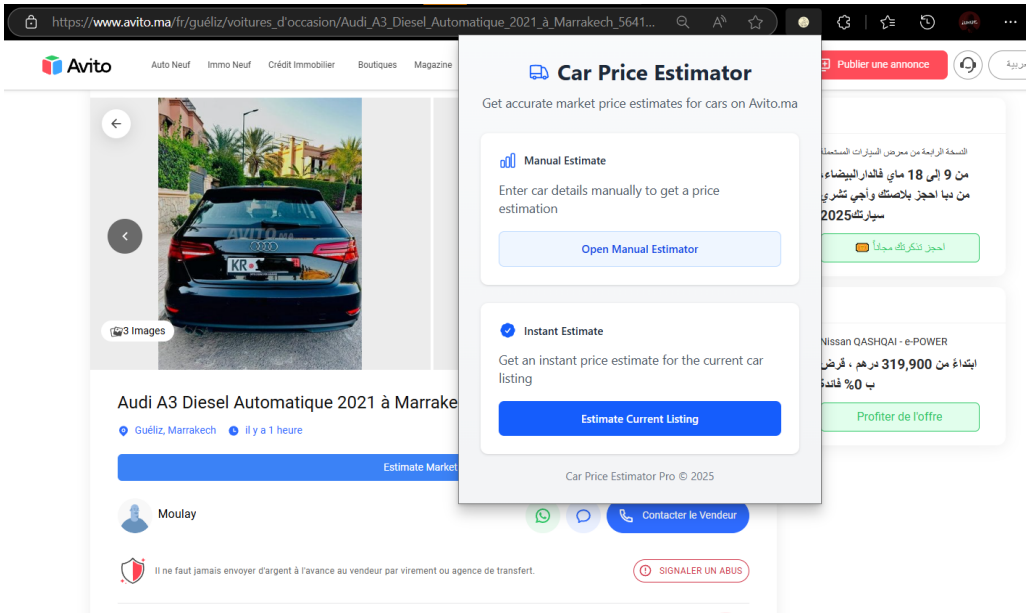

FIGURE 21 : INTERFACE DE L'EXTENSION *CAR PRICE ESTIMATOR PRO*.

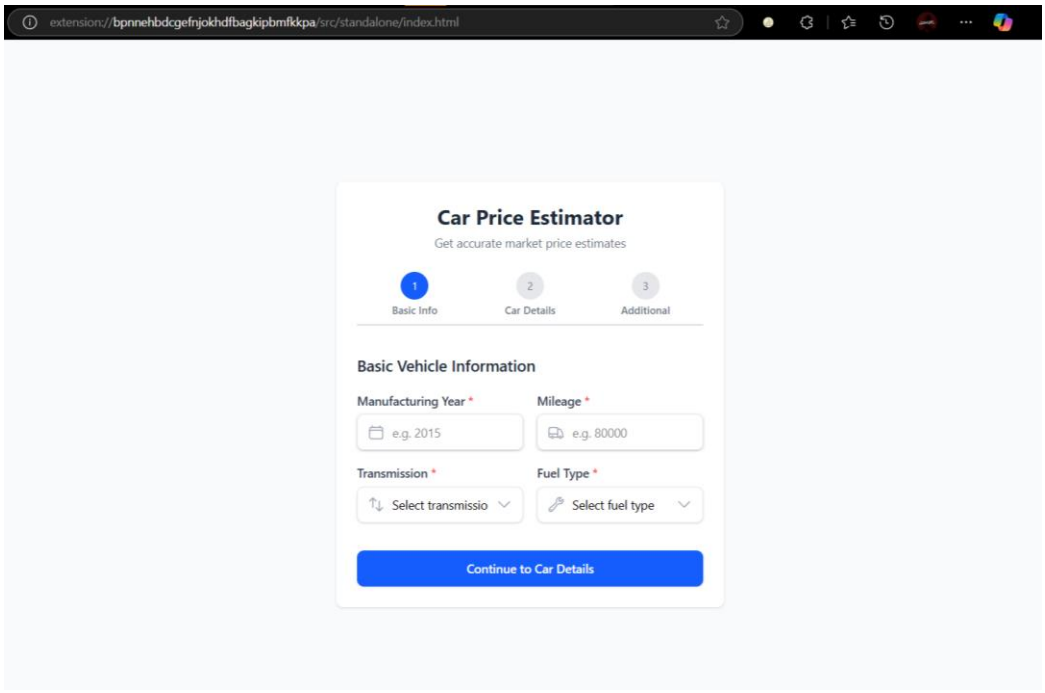

FIGURE 22 : INTERFACE ESTIMATION MANUELLE DE L'EXTENSION

7 Conclusion Générale

Ce projet d'estimation des prix des véhicules d'occasion au Maroc démontre la capacité des techniques d'intelligence artificielle à résoudre des problématiques économiques concrètes. Grâce à une méthodologie rigoureuse allant de la collecte de données jusqu'au déploiement d'une solution fonctionnelle, nous avons développé un système capable d'estimer avec précision la valeur d'un véhicule selon ses caractéristiques.

Les résultats sont particulièrement probants, avec un modèle de forêt aléatoire atteignant un coefficient de détermination (R^2) de 0.909 et une erreur moyenne d'environ 23 000 dirhams. Cette performance démontre la pertinence de notre approche et la fiabilité des prédictions pour un marché aussi complexe et volatil.

Le déploiement sous forme d'une API et d'une extension navigateur transforme ces capacités prédictives en un outil pratique et accessible, contribuant à réduire l'asymétrie d'information sur le marché de l'occasion marocain.

Néanmoins, le système présente certaines limites, notamment sa focalisation actuelle sur Avito.ma, une prise en compte limitée des spécificités régionales, et l'absence d'intégration des aspects visuels dans l'évaluation. Les perspectives d'évolution incluent l'extension à d'autres plateformes, l'intégration de l'analyse d'images, et le développement d'un mécanisme de mise à jour régulière pour refléter les évolutions du marché.

Ce projet illustre l'impact concret des technologies d'intelligence artificielle lorsqu'elles sont appliquées à des problématiques locales bien définies, avec une approche combinant rigueur méthodologique et sensibilité aux spécificités contextuelles.

Références

1. Github Repo. *Code source partie machine learning.* *https://github.com/eddaoudi-mohamed/Predict_Price_Car*
2. XGBoost. (2025 Feb 6).*XGBoost Documentation.* XGBoost Documentation — xgboost 3.0.1 documentation
3. FastAPI. (2024). *FastAPI Documentation*. https://fastapi.tiangolo.com/
4. Richardson, L. (2007). *Beautiful Soup Documentation*. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
5. Chrome Developers. (2024). *Chrome Extensions Documentation*. https://developer.chrome.com/docs/extensions/
6. Pandas *Documentation*. https://pandas.pydata.org/
7. Draw.io – *outil de modélisation*. https://app.diagrams.net/
8. Random Forest Regression - *scikit-learn User Guide* https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
9. Joblib*: Python Object Serialization.* https://joblib.readthedocs.io/en/latest/
10. scikit-learn *Documentation.* https://scikit-learn.org/stable/
11. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.

Annexes

Annexe A :

TABLEAU 2 : DICTIONNAIRE DES VARIABLES

Variable	Description	Type	Exemple
year	Année de fabrication	Numérique	2018
type_boit	Type de boîte de vitesses	Catégorielle	"Manuelle", "Automatique"
type_carburant	Type de carburant	Catégorielle	"Diesel", "Essence", "Hybride", "Electrique"
kilometrage	Kilométrage du véhicule	Numérique	120000
marke	Marque du véhicule	Catégorielle	"Renault", "Dacia", "Volkswagen"
model	Modèle du véhicule	Catégorielle	"Clio", "Logan", "Golf"
puissance	Puissance fiscale en CV	Numérique	6
premiere_main	Premier propriétaire	Ordinal	0 (Non), 1 (Inconnu), 2 (Oui)
Nombre_doors	Nombre de portes	Numérique	5
city	Ville de mise en vente	Catégorielle	"Casablanca", "Rabat", "Marrakech"
price	Prix annoncé en dirhams	Numérique	125000

Annexe B :

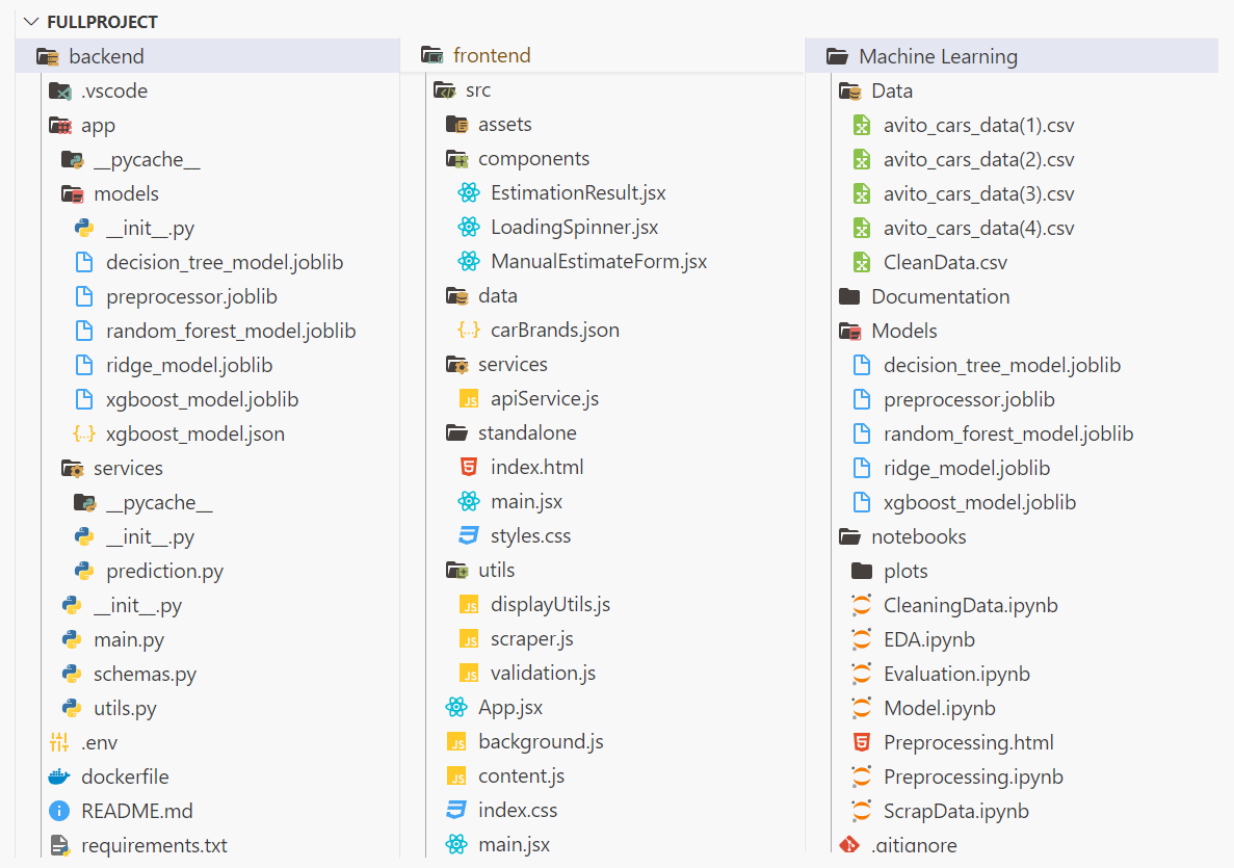

FIGURE 23 : STRUCTURE DU CODE SOURCE

Annexe C : Code Source

Extraits de code pour :

- `Frontend/src/services/apiService.js`

```
/**
 * Estimate car price using API
 * @param {Object} carData - Car details for estimation
 * @returns {Promise<Object>} - Estimation result
 */
export async function estimateCarPrice(carData) {
  try {
    const payload =  normalizeCarData(carData);
    console.log("API-service -> Sending request to:", `${API_BASE_URL}/estimate`);
    console.log("API-service -> With data:", JSON.stringify(payload, null, 2));
    const response = await fetch(`http://127.0.0.1:8000/estimate`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'X-API-KEY': `mysecretkey123`
      },
      body: JSON.stringify(payload)
    });

    if (!response.ok) {
      const errorData = await response.json();
      throw new Error(errorData.message || `API request failed with status ${response.status}`);
    }

    return await response.json();
  } catch (error) {
    console.error('API Error:', error);
    throw new Error(`Failed to connect to API: ${error.message}`);
  }
}
```

```
/**
 * Normalize car data for API request
 * @param {Object} data - Raw car data
 * @returns {Object} - Normalized car data
 */
export function normalizeCarData(data) {
  let kilometrage = 0;
  if (typeof data.kilometrage === "string" && data.kilometrage.includes("-")) {
    const [min, max] = data.kilometrage
      .split("-")
      .map(val => parseInt(val.replace(/\D/g, '')));
    kilometrage = Math.round((min + max) / 2);
  } else {
    kilometrage = parseInt(data.kilometrage) || 0;
  }
  let premiere_main_value = 1;
  const main = data.premiere_main?.toLowerCase();
  if (main === "non") premiere_main_value = 0;
  else if (main === "oui") premiere_main_value = 2;

  return {
    year: parseInt(data.year) || 2000,
    type_boit: data.type_boit?.toLowerCase() || 'manuelle',
    type_carburant: data.type_carburant?.toLowerCase() || 'essence',
    kilometrage,
    marke: data.marke?.toLowerCase() || 'unknown',
    model: data.model?.toLowerCase() || 'unknown',
    puissance: data.puissance ? parseInt(data.puissance.toString().match(/\d+/)?.[0]) : null,
    premiere_main: premiere_main_value,
    Nombre_doors: parseInt(data.Nombre_doors) || 5,
    city: data.city?.toLowerCase() || 'unknown',
    price: data.price ? parseInt(data.price.toString().replace(/\D/g, '')) : null,
    url: data.url || null
  };
}
```

FIGURE 24 : EXTENSION APISERVICE.JS FICHIER

- `Backend/app/main.py`

```
load_dotenv()

app = FastAPI(
    title="Car Price Estimator API",
    description="API for estimating car prices based on machine learning model",
    version="1.0.0",
    docs_url="/docs",
    redoc_url=None
)

# CORS Configuration
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# API Key Security
api_key_header = APIKeyHeader(name="X-API-KEY")

def get_api_key(api_key: str = Security(api_key_header)) -> str:
    if api_key != os.getenv("API_KEY"):
        print(f"Invalid API Key {api_key} should be {os.getenv('API_KEY')}")
        raise HTTPException(status_code=403, detail="Invalid API Key")
    return api_key
```

```
@app.post("/estimate", response_model=EstimationResult)
async def estimate_price(
    car_data: CarData,
    api_key: str = Depends(get_api_key)
):
    """
    Estimate car price based on provided features

    Parameters:
    - **year**: Manufacturing year
    - **type_boit**: Transmission type (Manuelle/Automatique)
    - **type_carburant**: Fuel type (Diesel/Essence/Hybride/Electrique)
    - **kilometrage**: Mileage in kilometers
    - **marke**: Car make/brand
    - **model**: Car model
    - **puissance**: Engine power (optional)
    - **premiere_main**: First hand (Oui/Non, optional)
    - **Nombre_doors**: Number of doors (optional, default=5)
    - **city**: Location city (optional)
    - **price**: Current listed price (optional)
    - **url**: Car listing URL (optional)
    """
    try:
        return prediction_service.predict(car_data)
    except Exception as e:
        raise HTTPException(
            status_code=400,
            detail=f"Error processing request: {str(e)}"
        )

@app.get("/health")
async def health_check():
    return {"status": "healthy", "model_loaded": prediction_service.model is not None}
```

FIGURE 25 : SCRIPT PYTHON FASTAPI (FICHIER `MAIN.PY`)

- Modèle de prétraitement (pipeline `scikit-learn`)

```python
def create_preprocessing_pipeline(df, target_col='price', test_size=0.2, random_state=42):
    """
    Fixed preprocessing pipeline with proper column handling
    """
    try:
        # Deep copy the DataFrame to prevent modifications
        df = df.copy()

        # Verify target column exists
        if target_col not in df.columns:
            raise ValueError(f"Target column '{target_col}' not found in DataFrame")

        # Separate features and target
        X = df.drop(columns=[target_col])
        y = df[target_col]

        print(f"Input shapes - X: {X.shape}, y: {y.shape}\n")

        # Create basic train-test split first
        X_train_basic, X_test_basic, y_train_basic, y_test_basic = train_test_split(
            X, y, test_size=test_size, random_state=random_state
        )

        # Get selected features
        selected_features = features[:-1]  # Exclude target column

        if not selected_features:
            raise ValueError("No features were selected by the feature selector")

        print(f"\nSelected features: {selected_features}")

        # Filter DataFrames to only include selected features
        X_train_basic_selected = X_train_basic[selected_features]
        X_test_basic_selected = X_test_basic[selected_features]

        # Now create preprocessing pipeline for the selected features only
        transformers = []

        # Identify numerical and categorical columns among selected features
        num_attribs = [col for col in selected_features
                       if X_train_basic[col].dtype in ['int64', 'float64']]
        cat_attribs = [col for col in selected_features
                       if X_train_basic[col].dtype == 'object']

        if num_attribs:
            num_pipeline = make_pipeline(StandardScaler())
            transformers.append(('num', num_pipeline, num_attribs))

        if cat_attribs:
            cat_pipeline = make_pipeline(OneHotEncoder(handle_unknown='ignore'))
            transformers.append(('cat', cat_pipeline, cat_attribs))

        preprocessor = ColumnTransformer(transformers)

        # Try to create stratified split using most important feature

        result = {
            'basic_split': {
                'X_train': X_train_basic_selected,
                'X_test': X_test_basic_selected,
                'y_train': y_train_basic,
                'y_test': y_test_basic,
                'preprocessor': preprocessor
            },
            'selected_features': selected_features
        }

        return result
```

FIGURE 26 : SCRIPTE POUR LA CREATION DE PIPELINE PRETRAITEMENT