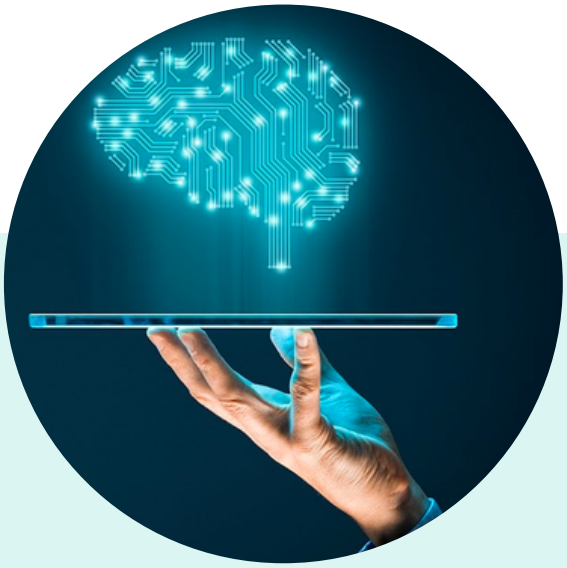
RAPPORT DE PROJET

DATA MINING MACHINE LEARNING

SUD Cloud & IoT - INE2



Année universitaire : 2022/2023

PREPARE PAR:

- ADNI YOUSRA
- KIMDIL ABDELLAH
- YAAKOUBI YASMINE

ENCADRÉ PAR:

- PR. IKRAM EL ASRI

PLAN

INTRODUCTION

Ce rapport présente l'analyse, le prétraitement, la visualisation et la construction d'un modèle de prédiction pour les données de voitures provenant de Wandaloo. Les données contiennent des informations sur les caractéristiques de différentes voitures, telles que la marque, le modèle, l'année de fabrication, la puissance, le kilométrage, le carburant, la transmission et le prix.

Dans la première partie de ce rapport, nous avons diagnostiqué les données et identifié les problèmes de nettoyage liés aux valeurs manquantes, aux doublons et à la structure. Nous avons proposé des solutions pour ces problèmes en utilisant des techniques telles que le remplacement des valeurs manquantes, la suppression des doublons et l'ajustement de la structure des données.

Dans la deuxième partie, nous avons utilisé la visualisation et les statistiques pour mieux comprendre les données de voitures. Nous avons créé des graphiques pour visualiser les relations entre les différentes caractéristiques et le prix des voitures. Nous avons également utilisé des statistiques descriptives pour résumer les caractéristiques clés des données.

Enfin, dans la troisième partie de ce rapport, nous avons construit un modèle de prédiction pour prédire le prix d'une voiture en utilisant les caractéristiques restantes du prétraitement.

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

La première partie du projet de prétraitement des données consiste à diagnostiquer les données et à identifier les problèmes de nettoyage liés à différents aspects tels que les valeurs manquantes, les doublons, la structure, etc.
Le processus de diagnostic implique l'examen des données pour repérer les problèmes qui peuvent être présents. Les données peuvent contenir des valeurs manquantes, des doublons ou des incohérences dans leur structure. Ces problèmes peuvent affecter la qualité des données et fausser les résultats de l'analyse.

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns #Seaborn is a Python data visualization library based on matplotlib
!pip install missingno
import missingno as msno #Missing data visualization module for Python
```

En premier lieu on importe les bibliothèques de Python qui vont nous aider dans notre projet . En voici une brève explication:

- **pandas**: une bibliothèque puissante pour la manipulation et l'analyse de données. Elle fournit des structures de données pour stocker et manipuler efficacement de grands ensembles de données, ainsi que des fonctions pour nettoyer, transformer et analyser des données.
- **matplotlib**: une bibliothèque pour la visualisation de données en Python. Elle permet de créer des graphiques en 2D et 3D, des diagrammes de dispersion, des histogrammes, etc.
- **numpy**: une bibliothèque pour le calcul scientifique en Python. Elle fournit des fonctions pour effectuer des calculs mathématiques complexes, des opérations sur les tableaux et les matrices, et des statistiques.
- **seaborn**: une bibliothèque pour la visualisation de données en Python, basée sur matplotlib. Elle permet de créer des graphiques plus complexes et est souvent utilisée pour la visualisation de données statistiques.
- **missingno**: une bibliothèque pour la visualisation de données manquantes en Python. Elle permet de visualiser rapidement les valeurs manquantes dans un ensemble de données et de comprendre leur distribution.

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

On commence par nettoyer la colonne "Price" et créer deux nouvelles colonnes, "price" et "sale_type". Voici un résumé de ce qu'on a fait :

```
# Replace "DH" with ","
cars['Price'] = cars['Price'].str.replace("DH",",")
# Print the header of the column
cars['Price'].head()
```

On a remplacé chaque occurrence de "DH" par une virgule dans la colonne "Prix" après on imprime l'en-tête de la colonne "Prix" pour vérifier si le remplacement a réussi.

```
# Split column into two
div = cars['Price'].str.split(",", expand = True)
div.head()
```

```
# Assign correct columns to price and sale_type columns in cars
cars['price'] = div[0]
cars['sale_type'] = div[1]
# Replace the empty values in the "Sale_Type" column with "no info".
cars['sale_type'] = cars['sale_type'].replace('', 'no info')
```

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

```
# Delete "*" and "* *" from column sale_type
cars['sale_type'] = cars['sale_type'].str.strip("* *")
cars['sale_type'] = cars['sale_type'].str.strip("*")
# Print the header and confirm new column creation
cars.head()
```

Puis on divise la colonne "Prix" en deux colonnes, "prix" et "sale_type", en utilisant la virgule comme délimiteur. et on attribut les colonnes correctes à "price" et "sale_type" dans le DataFrame "cars". on remplace toute valeur vide dans la colonne "sale_type" par "no info".Puis on supprime les astérisques et les espaces de la colonne "sale_type".On imprime l'en-tête du DataFrame "cars" pour confirmer la création des nouvelles colonnes

```
# Remove whitespace from the 'price' column
cars['price'] = cars['price'].str.strip()
# Convert the values of the 'price' column to float
cars['price'] = pd.to_numeric(cars['price'], errors='coerce')
print(cars.head())
```

```
# Drop Price column
cars.drop('Price', axis = 1, inplace = True)
cars.head()
```

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

Enfin on supprime la colonne originale "Price" du DataFrame "cars".et on supprime aussi tout espace blanc de début ou de fin de la colonne "prix". et on convertit les valeurs de la colonne "prix" en valeurs floats. Toute valeur non numérique sera convertie en NaN (Not a Number) et traitée avec le paramètre "errors='coerce'". Dans l'ensemble, ce code est un processus de nettoyage des données pour la colonne "Price" dans le DataFrame "cars", en la divisant en deux colonnes et en convertissant la colonne "price" dans un format numérique.

```
# Remove "cv" from 'Puissance fiscale' before conversion to float
cars['Puissance fiscale'] = cars['Puissance fiscale'].str.strip("cv")


# Remove whitespace from the 'Puissance fiscale' column
cars['Puissance fiscale'] = cars['Puissance fiscale'].str.strip()

#replacing missing values and dashes in the 'Puissance fiscale' column with 0 any empty strings with 0.
cars['Puissance fiscale'] = cars['Puissance fiscale'].fillna('0').replace('-', '0')
cars['Puissance fiscale'] = cars['Puissance fiscale'].replace('', '0')

# Convert Puissance fiscale to float
cars['Puissance fiscale'] = cars['Puissance fiscale'].astype('float')

# Print header to make sure change was done
cars['Puissance fiscale'].head()
```

Cet extrait de code traite de la colonne "Puissance fiscale" d'un cadre de données appelé "cars". Voici ce que fait le code étape par étape :

- Supprime la chaîne "cv" de la colonne "Puissance fiscale" à l'aide de la méthode strip() de string.

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

- Supprime tous les caractères d'espacement de début et de fin de la colonne "Puissance fiscale" à l'aide de la méthode strip() de string.
- Remplace toute valeur manquante ou tout tiret dans la colonne "Puissance fiscale" par 0 à l'aide des méthodes fillna() et replace() de pandas.
- Remplace toute chaîne vide par 0 en utilisant la méthode replace() de pandas.
- Convertit la colonne 'Puissance fiscale' en un type de données float en utilisant la méthode astype() de pandas.
- Imprime l'en-tête de la colonne "Puissance fiscale" pour vérifier que les modifications ont été effectuées.
- Dans l'ensemble, cet extrait de code nettoie et prépare la colonne 'Puissance fiscale' pour une analyse ultérieure en la convertissant en un type de données numériques et en remplissant toutes les valeurs manquantes ou invalides avec 0.

```
: cars.dtypes
```

```
Brand                  object
Model                  object
Version                object
Modèle                float64
Main                   object
Kilométrage           float64
Carburant              object
Transmision            object
Puissance fiscale     float64
Couleur extérieure     object
Etat du véhicule       object
Climatisation          object
Vitres électriques     object
Sièges électriques     object
Ordinateur de bord     object
Start & Stop           object
```

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

```
Régulateur de vitesse                object
Allumage auto. des feux              object
Détecteur de pluie                   object
Commandes au volant                  object
Ecran tactile                        object
Rétroviseurs électriques             object
Ouverture auto. du coffre            object
Démarrage mains libres               object
Banquette arrière rabattable 1/3-2/3 object
Caméra de recul                      object
Bluetooth                            object
Jantes aluminium                     object
Volant cuir                          object
Feux de jour                         object
Barres de toit                       object
Toit                                 object
Airbags                              object
ABS                                  object
ESP                                  object
Antipatinage                         object
Architecture                        float64
Cylindrée                           float64
Conso. ville                        float64
Conso. route                        float64
Vitesse maxi.                       float64
Volume du réservoir                 float64
price                               float64
sale_type                            object
dtype: object
```

Notre objectif dans la parie de nettoyage c'était de convertir le maximum en float et on voit d'après la commande "cars.dtypes" qu'on a réussi à convertir "kilométrage","Puissance fiscale", "Architecture", "Cylindrée", "Conso. ville", "Conso. route", "Vitesse maxi.", "Volume du réservoir", "price".

```
cars = cars.fillna({'Architecture':0,
                    'Cylindrée':0,
                    'Vitesse maxi.':0})
```

PARTIE 1 : PRÉTRAITEMENT DES DONNÉES

L'extrait de code remplit les valeurs manquantes (NaN) dans les colonnes "Architecture", "Cylindrée" et "Vitesse maxi" de l'ensemble de données "cars" avec la valeur 0.
Cette approche est appelée "imputation" et il s'agit d'une technique courante pour traiter les valeurs manquantes dans les ensembles de données. En remplissant les valeurs manquantes, nous pouvons éviter de supprimer des lignes ou des colonnes entières de données, ce qui pourrait entraîner une perte d'informations.

```
# Impute missing data
cars = cars.fillna({'Ouverture auto. du coffre': 'no info',
                    'Etat du véhicule': 'no info',
                    'Couleur extérieure' : 'no info',
                    'Carburant' : 'no info',
                    'Kilométrage' : 'no info',
                    'Main' : 'no info',
                    'Modèle' : 'no info',
                    'Version' : 'no info',
                    'Transmision' : 'no info'})
```

Nous avons utilisé la méthode fillna() pour remplacer toutes les valeurs NaN dans des colonnes spécifiques par la chaîne de valeur "no info".

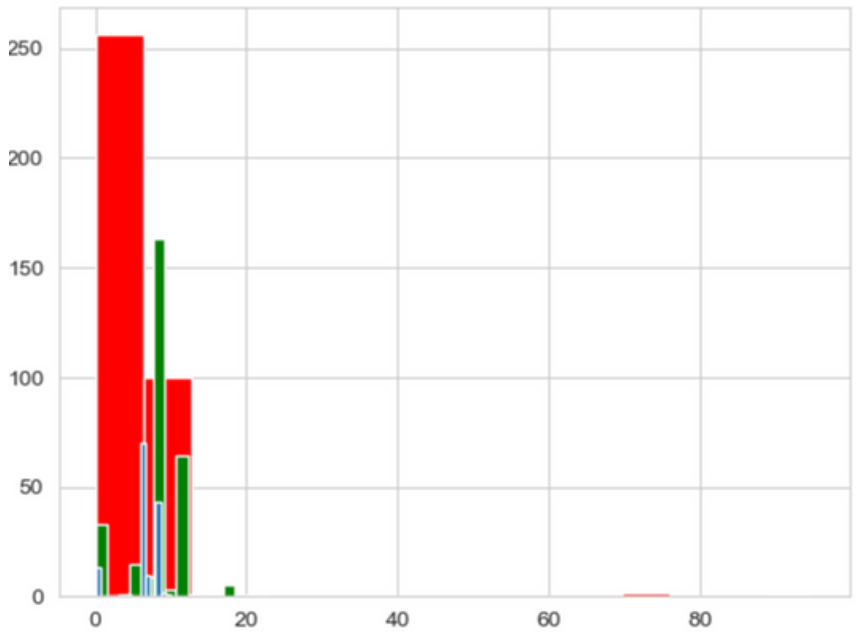
PARTIE 2 : VISUALISATION ET EXPLORATION DES DONNÉES

La visualisation et l'exploration des données sont des étapes importantes dans l'analyse des données. Ces étapes permettent de mieux comprendre les données, d'identifier les tendances, les modèles, les relations entre les variables, les valeurs aberrantes et les anomalies. Ces informations peuvent être utilisées pour prendre des décisions éclairées, résoudre des problèmes et découvrir des opportunités.

La visualisation des données est un moyen de représenter graphiquement les données pour en faciliter l'analyse et la compréhension. Les graphiques peuvent inclure des histogrammes, des graphiques à barres, des graphiques à secteurs, des nuages de points, des cartes thermiques, des diagrammes de dispersion, des graphiques en boîte, des diagrammes de violon, etc. Les choix de graphiques dépendent de la nature des données et des questions que l'on cherche à résoudre.

PARTIE 2 : VISUALISATION ET EXPLORATION DES DONNÉES

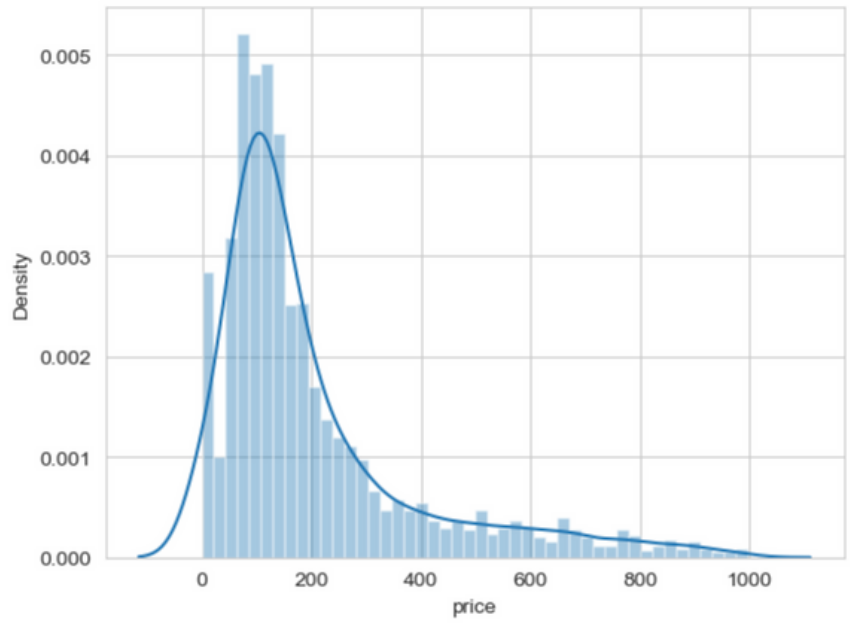
```
# stacked Histograms
Filter1= cars.Brand == 'MERCEDES'
Filter3= cars.Brand == 'SEAT'
Filter2= cars.Brand == 'PEUGEOT'


plt.hist(cars.Puissance_fiscale[Filter2],bins=15, color="red")
plt.hist(cars.Puissance_fiscale[Filter1],bins=15, color="green")
plt.hist(cars.Puissance_fiscale[Filter3],bins=15)
plt.show()
```



Ici on a tracé un histogramme de la variable "Puissance_fiscale" pour trois marques de voitures différentes : MERCEDES, SEAT et PEUGEOT.
L'histogramme permet de comparer la distribution de la puissance fiscale pour chaque marque. La couleur des histogrammes permet de distinguer les différentes marques.

PARTIE 2 : VISUALISATION ET EXPLORATION DES DONNÉES

```
]: sns.distplot(cars['price'])
<AxesSubplot:xlabel='price', ylabel='Density'>
```



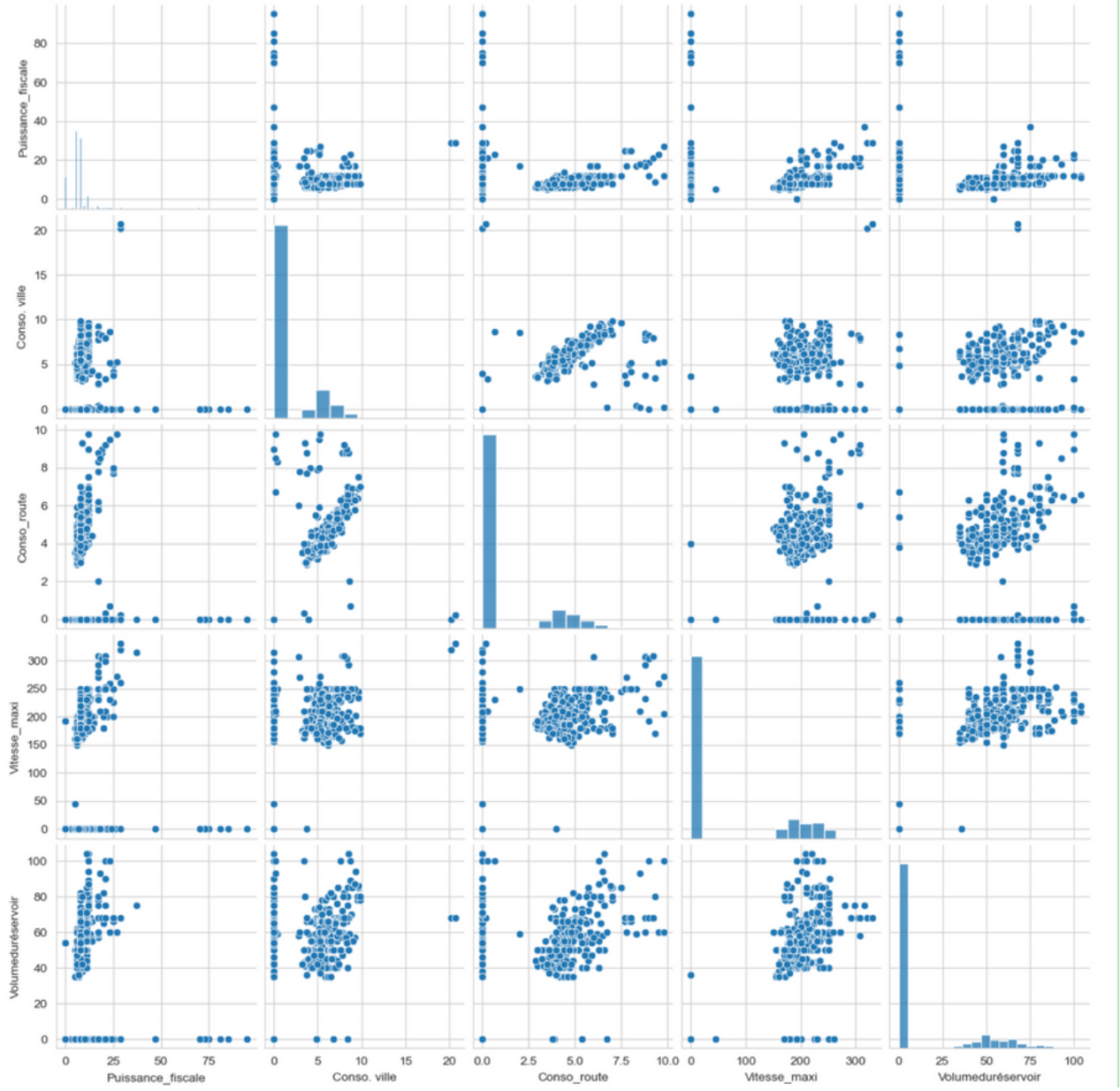
On a utilisé la bibliothèque Seaborn pour tracer un histogramme de la variable "price" du DataFrame "cars".

- La fonction distplot() de Seaborn permet de tracer l'histogramme et la densité de probabilité de la variable "price". Elle peut également afficher une estimation de la densité de probabilité basée sur les données en utilisant une fonction de noyau (par défaut, une fonction de noyau gaussienne).

PARTIE 2 : VISUALISATION ET EXPLORATION DES DONNÉES

```
]: num_cols = ['Kilométrage', 'Puissance_fiscale', 'Conso. ville', 'Conso_route', 'Vitesse_maxi', 'Volumeduréservoir']
   sns.pairplot(cars[num_cols])
```

```
<seaborn.axisgrid.PairGrid at 0x7fa141247d90>
```

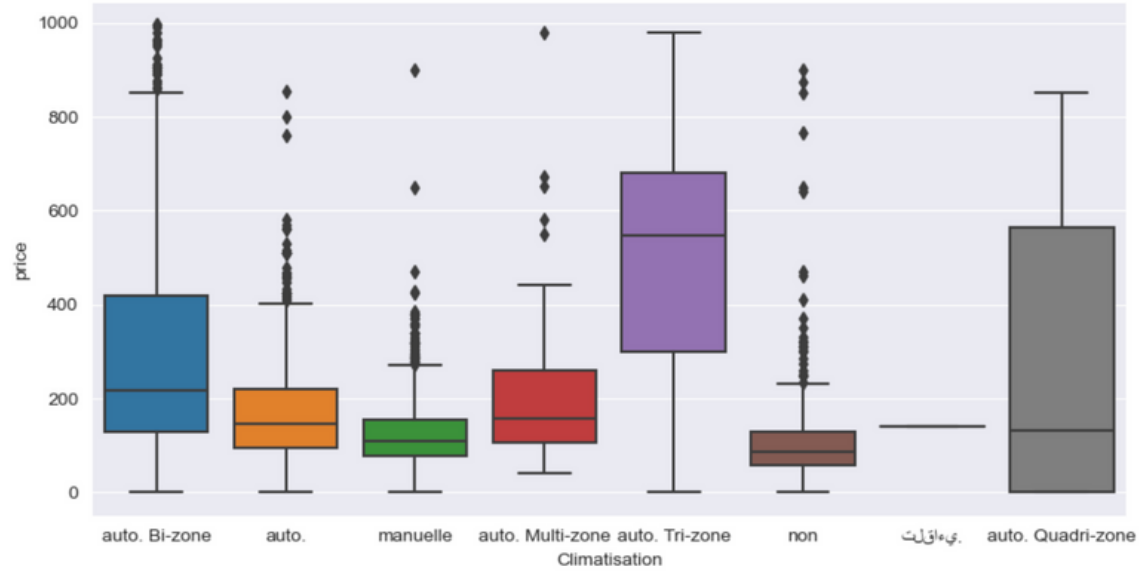


PARTIE 2 : VISUALISATION ET EXPLORATION DES DONNÉES

Visualisation de la relation entre plusieurs variables numériques dans le DataFrame "cars" à l'aide d'une matrice de graphiques en nuage de points. Cela permet de détecter des tendances et des relations potentielles entre les variables, ce qui peut aider à comprendre la structure des données.

```
3]: import matplotlib.pyplot as plt
    import seaborn as sns
    sns.set_style('darkgrid')
    cat_cols = ['Brand', 'Carburant', 'Transmision', 'Couleur extérieure', 'Etat du véhicule', 'Climatisation']
    for col in cat_cols:
        plt.figure(figsize=(10,5))
    sns.boxplot(x=col, y='price', data=cars)
```

```
<AxesSubplot:xlabel='Climatisation', ylabel='price'>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
```



PARTIE 2 : VISUALISATION ET EXPLORATION DES DONNÉES

On a tracé des graphiques en boîte pour chaque colonne catégorielle (Brand, Carburant, Transmision, Couleur extérieure, Etat du véhicule, Climatisation) dans un DataFrame nommé cars.
On importe les modules matplotlib.pyplot et seaborn pour la visualisation des données, on définit le style de graphique par défaut avec sns.set_style('darkgrid') et on crée une liste contenant les noms de chaque colonne catégorielle.
Le code utilise ensuite une boucle for pour parcourir chaque colonne catégorielle et créer une nouvelle figure pour chacune d'elles. Cependant,

PARTIE 3 : PRÉDICTION DE LA CONSTRUCTION DU MODÈLE

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pandas as pd

# Load data
cars = pd.read_csv('wandaloo_cars.csv')

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(cars.drop('price', axis=1), cars['price'], test_size=0.2, random_s

# Convert categorical variables to dummy variables
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)

# Fit linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

Ce code est un exemple d'utilisation du modèle de régression linéaire de la bibliothèque scikit-learn (sklearn) pour prédire le prix des voitures en fonction de différentes caractéristiques. Voici les étapes détaillées du code :

- Importation des bibliothèques : Le code commence par importer les bibliothèques nécessaires pour l'analyse de données et la modélisation, notamment train_test_split et LinearRegression de sklearn, ainsi que pandas pour la manipulation des données.
- Chargement des données : Le code charge un ensemble de données sur les voitures à partir d'un fichier CSV à l'aide de la fonction pd.read_csv().
- Séparation des données : Les données sont ensuite divisées en un ensemble d'apprentissage (X_train, y_train) et un ensemble de test (X_test, y_test) à l'aide de la fonction train_test_split(). L'ensemble de test représente une partie des données qui sera utilisée pour évaluer les performances du modèle.

PARTIE 3 : PRÉDICTION DE LA CONSTRUCTION DU MODÈLE

```
# Predict response variable for test set
y_pred = lin_reg.predict(X_test)

# Evaluate model performance
from sklearn.metrics import mean_squared_error, r2_score

# Mean squared error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean squared error (MSE): {:.2f}".format(mse))

# R-squared score (coefficient of determination)
r2 = r2_score(y_test, y_pred)
print("R-squared score: {:.2f}".format(r2))
```

- Conversion des variables catégorielles en variables fictives : Les variables catégorielles dans les ensembles d'apprentissage et de test sont converties en variables fictives à l'aide de la fonction pd.get_dummies(). Cette étape est nécessaire car la plupart des modèles de machine learning ne peuvent pas traiter directement des variables catégorielles.
- Entraînement du modèle de régression linéaire : Le modèle de régression linéaire est initialisé à l'aide de la fonction LinearRegression(), puis ajusté aux données d'apprentissage à l'aide de la méthode fit().
- Prédiction de la variable de réponse pour l'ensemble de test : La méthode predict() est utilisée pour prédire la variable de réponse (y) pour l'ensemble de test.
- Évaluation des performances du modèle : Les performances du modèle sont évaluées à l'aide de deux mesures courantes de la régression linéaire : le Mean Squared Error (MSE) et le coefficient de détermination R^2. Les résultats sont affichés à l'écran à l'aide de la fonction print().

CONCLUSION

En conclusion, le datamining et le machine learning sont des outils puissants pour extraire des informations précieuses à partir de données complexes. Grâce à ces techniques, il est possible de découvrir des modèles cachés, d'identifier des tendances et de prédire des résultats futurs avec une grande précision.
Cependant, il est important de se rappeler que le datamining et le machine learning ne sont pas des solutions universelles à tous les problèmes. Les résultats obtenus dépendent en grande partie de la qualité et de la quantité des données utilisées, ainsi que de la capacité de l'algorithme à trouver des relations significatives.
En fin de compte, le datamining et le machine learning sont des outils qui peuvent aider à prendre des décisions plus éclairées et à améliorer les résultats dans de nombreux domaines, mais ils doivent être utilisés avec soin et avec une compréhension claire de leurs limites et de leurs avantages.