IPSSI

IA CAR PRICE PREDICTION

Thassadhith AIT LARBI
Juba ASMA
Ouissal BALLOUTI
Joanne MISSIHOUN

Groupe 3

Contenu

- PROBLEMATIQUE
- METHODOLOGIE
- MISE EN PLACE
- DEMONSTARTION

PROBLEMATIQUE

Développer une IA pour prédire le prix de vente des voitures d'occasion à partir des cractéristiques de ces dernières

Methodologie

1

Collecte de données

2

EDA & Feature Enginering

3

Intelligence Artificielle

4

Interface Web

Collecte de données

Scrapping

- Site de vente des voiture d'occasion (lacentrale.fr)
- Outils (Bsoup, user-agent .....)
- Récupération des liens
- Données récupérées : Marque , nb de km, puissance, co2 ...
- Acceptation de cookies

Difficultés

- Contournement du contrôle anti-scrapping mis en place par lacentrale

Liens

```python
import requests
from bs4 import BeautifulSoup
from fake_useragent import UserAgent

def recup_liens_voitures():
    ua = UserAgent()

    # Liste pour stocker tous les href
    all_links = []

    # Page de départ
    i = 1

    # Continuer à scraper tant que nous avons moins de 1500 liens
    while len(all_links) < 1500:
        url = f'https://www.lacentrale.fr/listing?makesModelsCommercialNames=&options=&p

        headers = {'User-Agent': ua.random}
        response = requests.get(url, headers=headers)
        if response.status_code != 200:
            print(f"Erreur de requête pour {url}, code d'état: {response.status_code}")
            i += 1
            continue
        soup = BeautifulSoup(response.content, 'html.parser')
```


```python
            i += 1
            continue
        soup = BeautifulSoup(response.content, 'html.parser')
```


```python

        cars = soup.find_all('div', class_='searchCardContainer')
        print(f"Pas de voitures trouvées sur la page {url}")

        for car in cars:
            if car is not None:  # Vérifiez que le car n'est pas None
                try:
                    href = car.find("a").get('href')
                    if href is not None:  # Vérifiez que href n'est pas None
                        # Ajoutez href à la liste all_links
                        all_links.append(href)
                    else:
                        print("Le lien n'a pas d'attribut href.")
                except AttributeError:  # attrape l'exception si car.find("a") est None
                    print("Pas de balise <a> trouvée dans cet élément.")
            else:
                print("Aucun élément trouvé avec la classe spécifiée.")

        # Ne pas oublier d'augmenter le compteur de la page
        i += 1

    # Retourne la liste all_links à la fin de la fonction
    return all_links
```

Caractéristiques

```python
price =soup.find({"span"}, {"class":"PriceInformation_classifiedPrice__b-Jae"}).text
price = price.replace("\u202f", "").replace("\xa0", "").replace("€", "").strip()
price = int(price)
soup.find({"li"}, {"id":"year"}).find({"span"},{"class":"Text_Text_text Text_Text_body1"}).t
soup.find({"li"}, {"id":"origin"}).find({"span"},{"class":"Text_Text_text Text_Text_body1"})
soup.find({"li"}, {"id":"firstCirculationDate"}).find({"span"},{"class":"Text_Text_text Text
soup.find({"li"}, {"id":"firstHand"}).find({"span"},{"class":"Text_Text_text Text_Text_body1
soup.find({"li"}, {"id":"mileage"}).find({"span"},{"class":"Text_Text_text Text_Text_body1"}
soup.find({"li"}, {"id":"energy"}).find({"span"},{"class":"Text_Text_text Text_Text_body1"})
soup.find({"li"}, {"id":"gearbox"}).find({"span"},{"class":"Text_Text_text Text_Text_body1"}
soup.find({"li"}, {"id":"externalColor"}).find({"span"},{"class":"Text_Text_text Text_Text_b
```

```python
def scrape_car_info(links):
    car_info_list = []

    # Spécifiez les headers de votre navigateur
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/5

    for link in links:
        page = requests.get(link, headers=headers)
        soup = BeautifulSoup(page.content, 'html.parser')
```

Cookies

```python
# Passer à l'iframe par son nom
time.sleep(3)
driver.switch_to.frame("sp_message_iframe_839013")
time.sleep(3)
# Ensuite, essayez de localiser le bouton et cliquez dessus
time.sleep(3)
accepter_button = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.XPATH, '//button[@title="Accepter"]'))
)
time.sleep(3)
```

Données récupérées

```
Index(['Prix', 'year', 'origin', 'firstCirculationDate', 'firstHand', 'energy',
       'gearbox', 'externalColor', 'doors', 'critAir', 'owners',
       'mileage_value', 'co2_value', 'powerDIN_value',
       'ratedHorsePower_value'],
      dtype='object')
```

- Prix
- Année de fabrication
- Origine
- Premère date de mise en circulation
- Source d'énergie

- Type de boîte
- Couleur
- Nombre de portes
- Puissance des chevaux
- Kilométrage

EDA & Feature Engineering

EDA

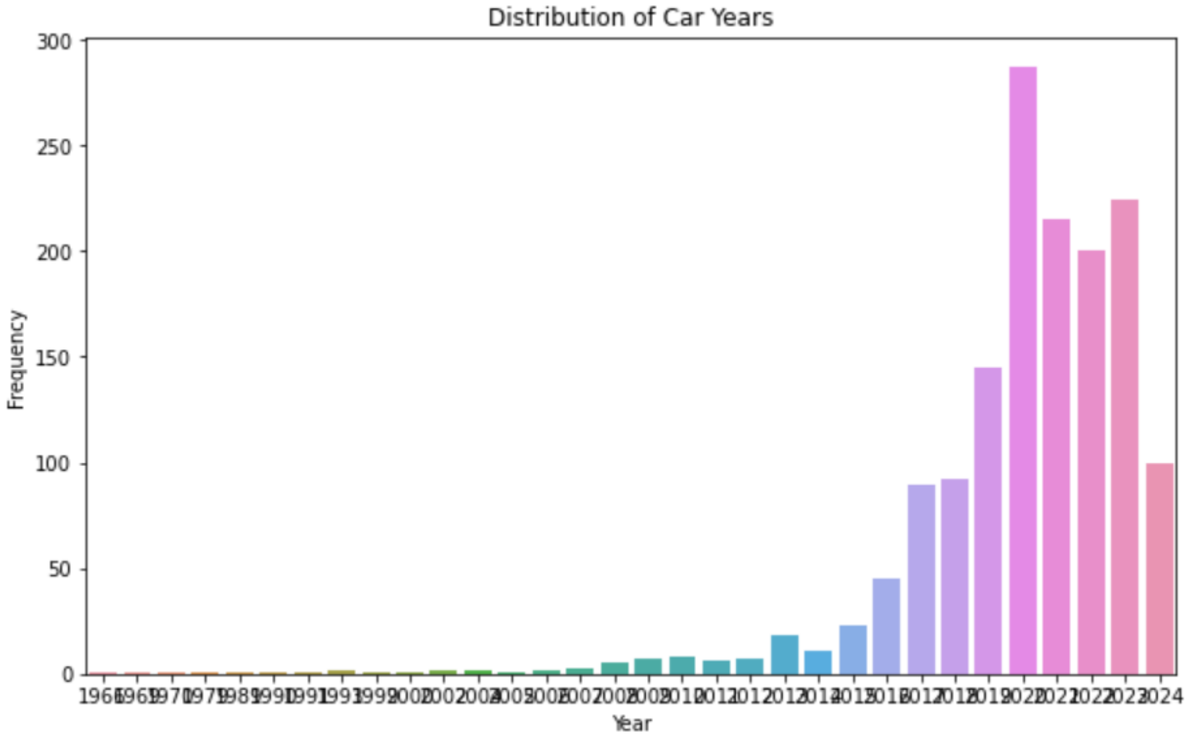
- Distribution de quelque attributs.
- Suppression des valeurs aberrantes (outliers) pour 'Prix'
- Matrice de Corrélation

Feature Engineering

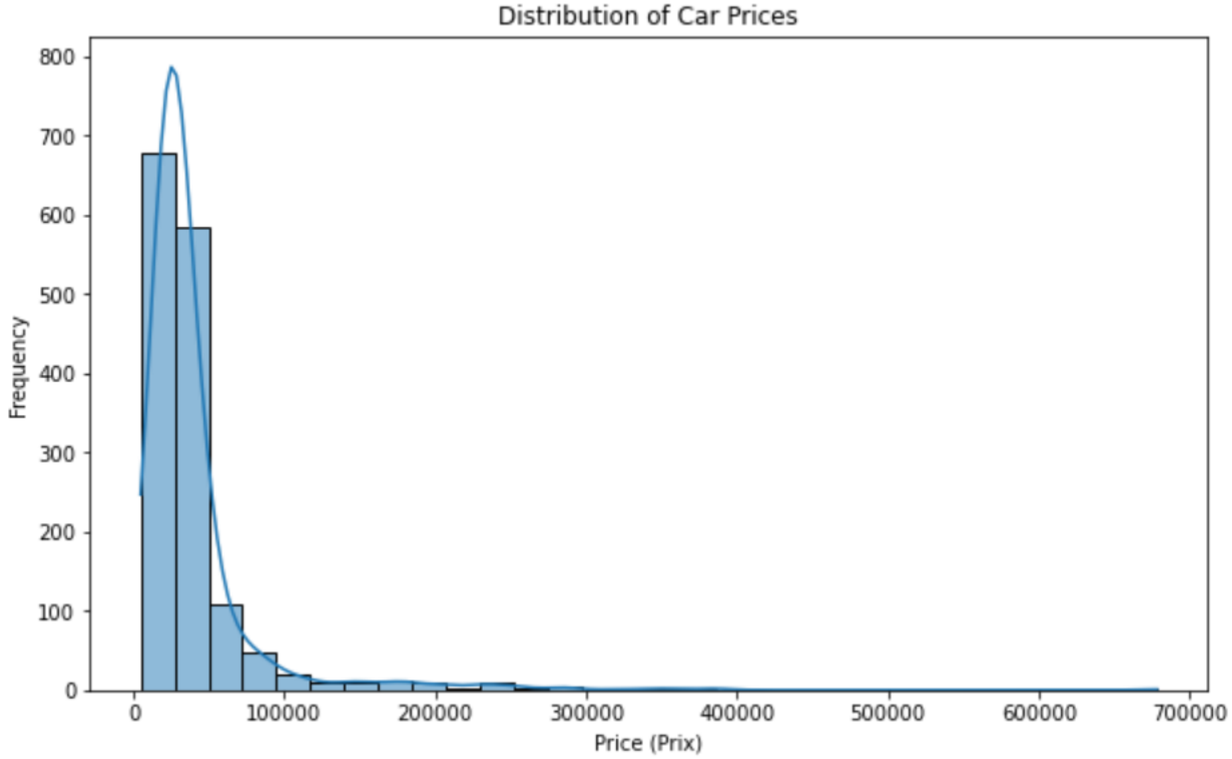
- Nettoyage des données.
- Encodage des variables catégorielles :
  - One-hot encoding
  - LabelEncoder
- Prétraitement pour la Modélisation.

Analyse Exploratoire de Données (EDA)

- **Distribution de quelques attributs**



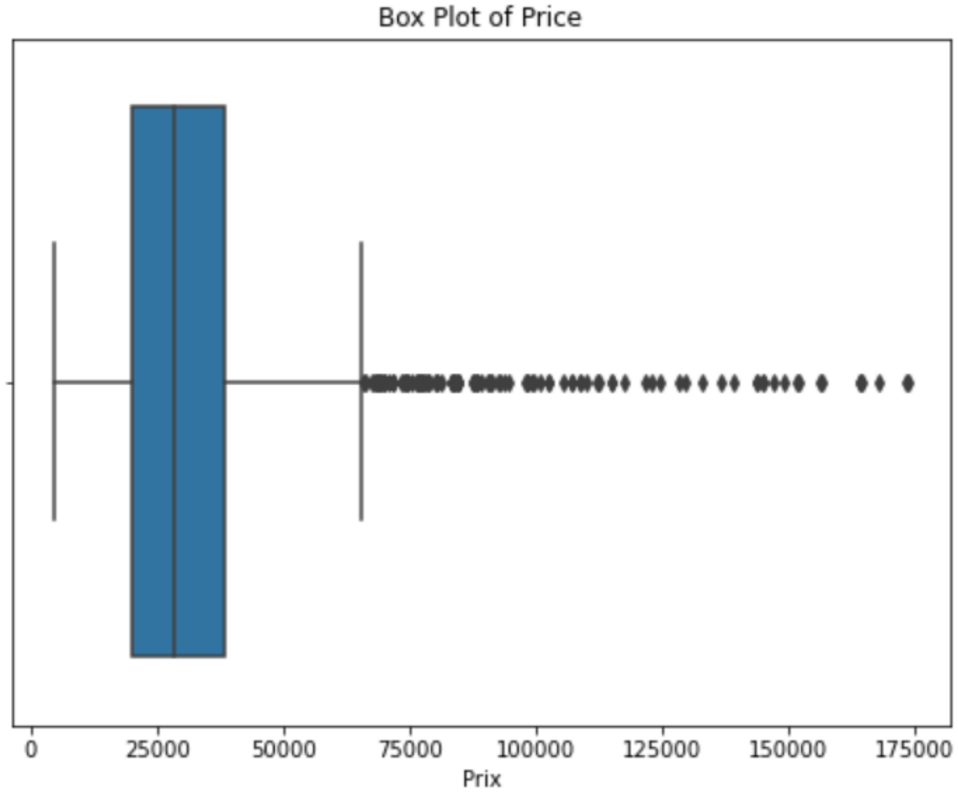
Année de mise en circulation



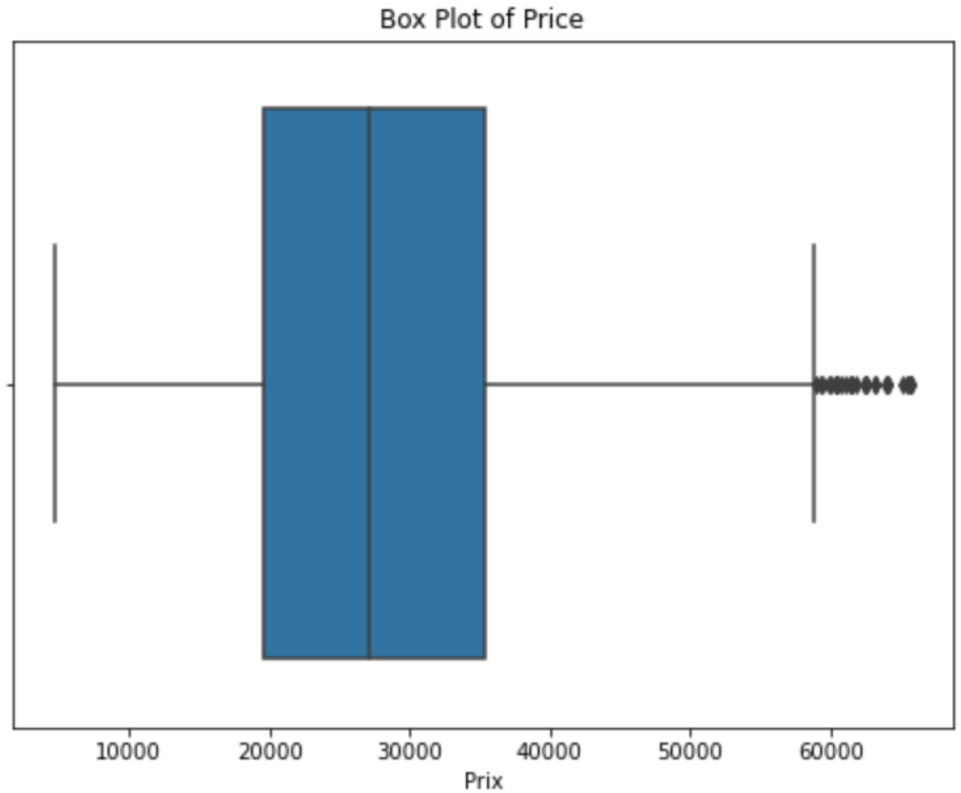
Prix de vente

Analyse Exploratoire de Données (EDA)

- Suppression des valeurs aberrantes (outliers) pour 'Prix'



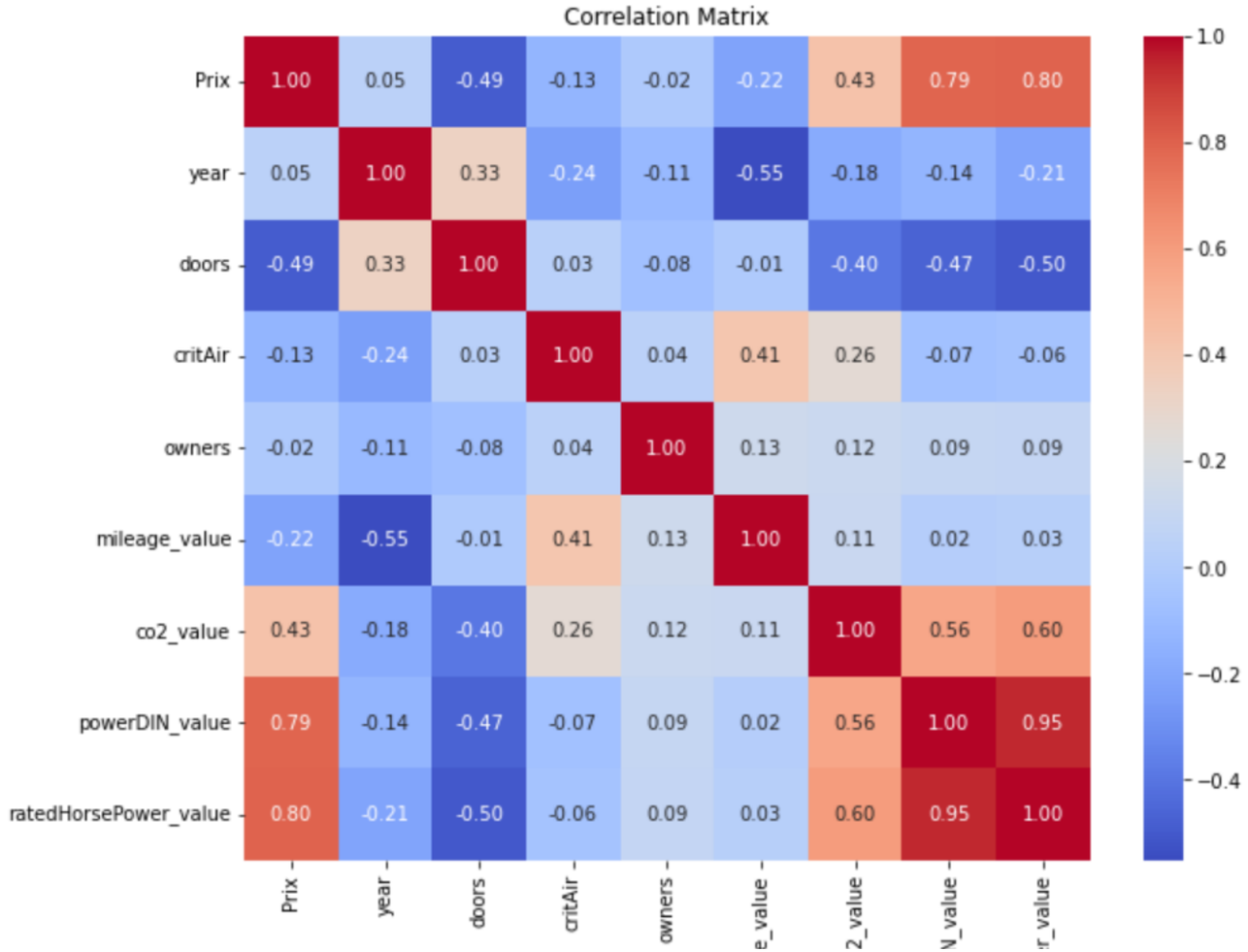
Avant



Après

Analyse Exploratoire de Données (EDA)

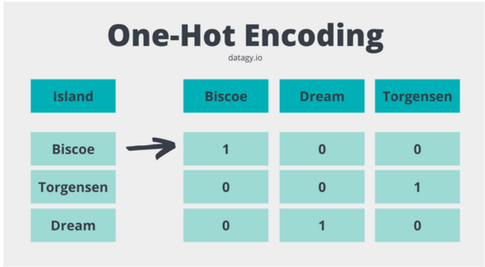
- **Matrice de Corrélation**



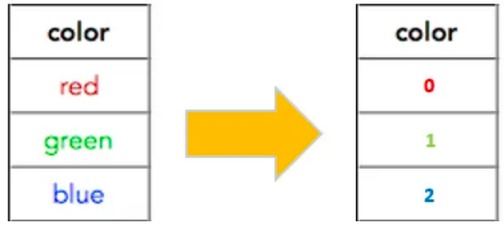
- **Variables avec corrélation positive avec le Prix :**
  - Year
  - Co2_value
  - PowerDIIN_value
  - RatedHorsePower_value
- **Variables avec corrélation négatives avec le Prix :**
  - Doors
  - Owners
  - CritAir
  - Mileage_value

Features engineering

- ## Encodage des variables catégorielles :
  - One-hot encoding : Transformation des colonnes 'energy', origin & externalColors.



  - LabelEncoder : Transformation des colonnes gearbox & firsthand.



- ## Prétraitement pour la Modélisation :
  - Extraction des Caractéristiques : Extraction des valeurs numériques de quelques attributs.
  - Mise à l'échelle Min-Max.
  - Normalisation avec StandardScaler.

Intelligence Artificielle

Train - test

Train : 80%

Test : 20%

Modèles

- Random Forest
- XGboost
- Regression Linéaire

Modèle de prédiction Random Forest

- Application du modèle sur l'ensemble des variables

```python
# Model Training
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

Résultats :

Mean Absolute Error:

3918.90110701107

Mean Squared Error:

29054904.51321033

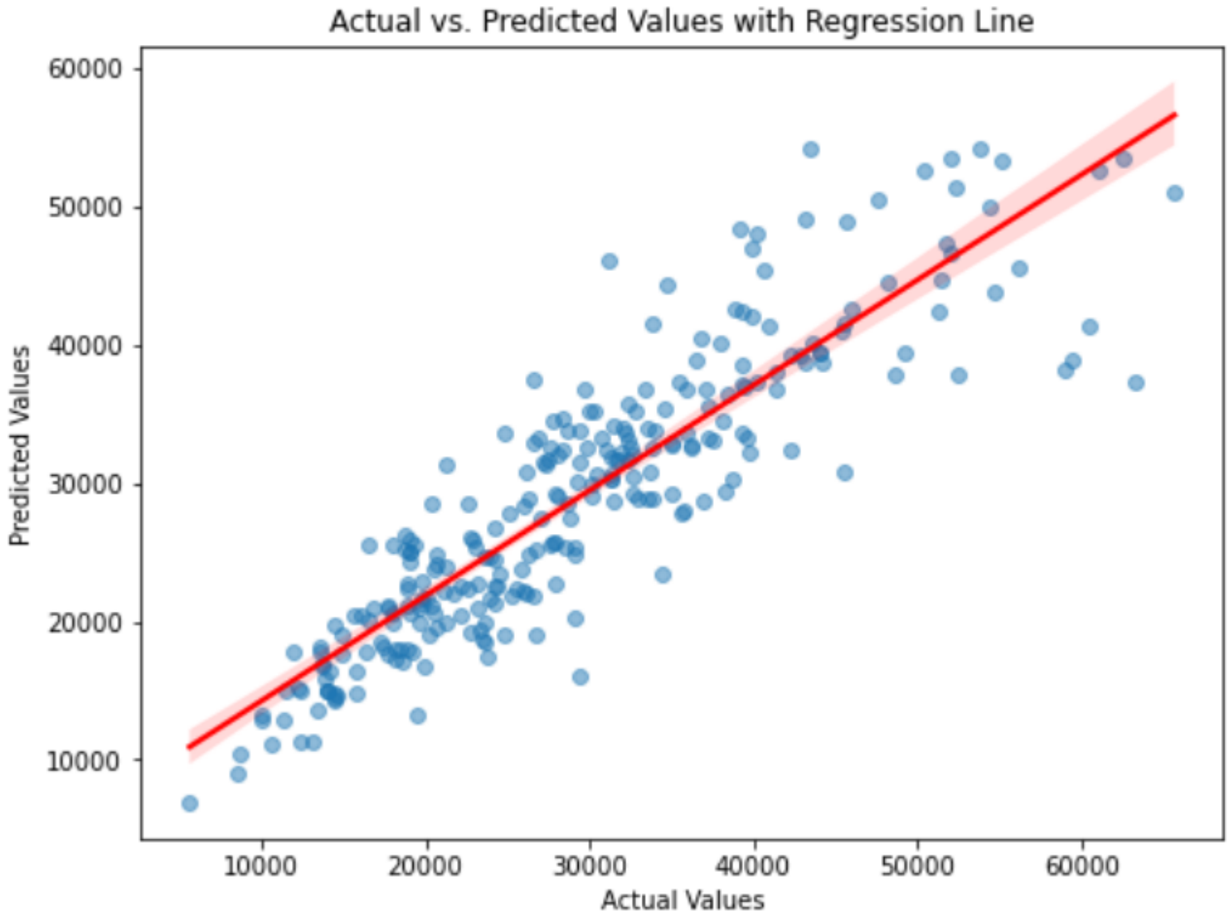
R-squared:

0.79597590663845

Modèle de prédiction Random Forest

- Scalling (Mise à l'échelle Min-Max)

- Random Forest Model MAE: 3864.2778597785973
- Random Forest Model R-squared: 0.7924668326057912



Optimisation du modèle Random Forest

- Recherche des paramètres optimaux

```
#  Define the parameter grid for Grid Search
param_grid = {
    'n_estimators': [100, 200, 300],          # Number of trees in the forest
    'max_depth': [None, 10, 20, 30],          # Maximum depth of the tree
    'min_samples_split': [2, 5, 10],          # Minimum number of samples required to split an internal node
    'min_samples_leaf': [1, 2, 4]             # Minimum number of samples required to be at a leaf node
}

# 5. Create the Random Forest regressor
rf_model = RandomForestRegressor(random_state=42)

# 6. Initialize Grid Search with the Random Forest model and parameter grid
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5, n_jobs=-1, scoring='neg_mean_absc

# 7. Perform Grid Search on the training data
grid_search.fit(X_train_normalized, y_train)

# 8. Get the best hyperparameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# 9. Evaluate the best model on the test set
y_pred_best = best_model.predict(X_test_normalized)
mae_best = mean_absolute_error(y_test, y_pred_best)
r2_best = r2_score(y_test, y_pred_best)
```

Résultats :

Best Hyperparameters {

'max_depth': None,

'min_samples_leaf': 1,

'min_samples_split': 2,

'n_estimators': 100}

Best MAE on Test Set:

3915.2778597785973

Best R-squared on Test Set:

0.796084520193262 8

Application du modèle Random Forest sur les variables les plus corrélées

Résultats :

Mean Absolute Error:
4086.2179797611116

Mean Squared Error:
32759861.791027702

R-squared:
0.7699596260065077

Optimisation des Hyperparamètres avec GridSearchCV :

Best Hyperparameters:
{'max_depth': None,
'min_samples_leaf': 2,
'min_samples_split': 10,
'n_estimators': 100}

Best MAE on Test Set:
4109.632257739608

R-squared on Test Set:
0.7701906387525453

Optimisation des Hyperparamètres avec RandomizedSearchCV :

Best Hyperparameters (Randomized):
{'max_depth': 20,
'min_samples_leaf': 7,
'min_samples_split': 12,
'n_estimators': 187}

Best MAE (Randomized):
4349.206760635928

R-squared (Randomized):
0.7509629090743738

Modèle de prédiction XGboost

- Application du modèle sur l'ensemble des variables

```python
# Create the XGBoost regressor
xgb_model = XGBRegressor(random_state=42)

# Train the XGBoost model
xgb_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_xgb = xgb_model.predict(X_test)

# Calculate the metrics (MAE and R2 score)
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
r2_xgb = r2_score(y_test, y_pred_xgb)

print("XGBoost Model MAE:", mae_xgb)
print("XGBoost Model R-squared:", r2_xgb)
```

Résultats :

XGBoost Model MAE:

4366.544548907403

XGBoost Model R-squared:

0.7384743001422346

Optimisation du modèle XGBoost

Résultats avec des Hyperparamètres définis :

XGBoost Model MAE:
4194.740250590982

XGBoost Model R-squared:
0.7616487893047997

Résultats des Hyperparamètres avec RandomizedSearchCV

Best Hyperparameters (Randomized):
{'colsample_bytree': 1.0, 'gamma': 0.1, 'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight': 3, 'n_estimators': 393, 'random_state': 42, 'subsample': 0.9}

XGBoost Model MAE (Randomized):
4203.153749135147

XGBoost Model R-squared (Randomized):
0.769440468907724

Application du modèle XGBoost sur les variables les plus corrélées

Résultats avec des Hyperparamètres définis :

Mean Absolute Error:
4086.2179797611116

Mean Squared Error:
32759861.791027702

R-squared:
0.769959626006507 7

Optimisation des Hyperparamètres avec GridSearchCV :

Best Hyperparameters:
{'max_depth': None,
'min_samples_leaf': 2,
'min_samples_split': 10,
'n_estimators': 100}

Best MAE on Test Set:
4109.632257739608

R-squared on Test Set:
0.7701906387525453

Résultats des Hyperparamètres avec RandomizedSearchCV

Best Hyperparameters :
{'max_depth': 20,
'min_samples_leaf': 7,
'min_samples_split': 12,
'n_estimators': 187}

Best MAE on Test Set :
4349.206760635928

R-squared on Test Set :
0.7509629090743738

Modèle de prédiction SVM


```python
from sklearn.svm import SVR

# Create the Support Vector Regression (SVR) model
svm_model = SVR(kernel='linear')  # You can choose the kernel type ('linear', 'rbf', etc.)

# Train the SVR model
svm_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_svm = svm_model.predict(X_test)

# Calculate the metrics (MAE and R2 score)
mae_svm = mean_absolute_error(y_test, y_pred_svm)
r2_svm = r2_score(y_test, y_pred_svm)

print("SVM Model MAE:", mae_svm)
print("SVM Model R-squared:", r2_svm)
```

Résultats :

SVM Model MAE:

9502.350209089602

SVM Model R-squared:

0.04031887681528401

Modèle de Régression linéaire multiple

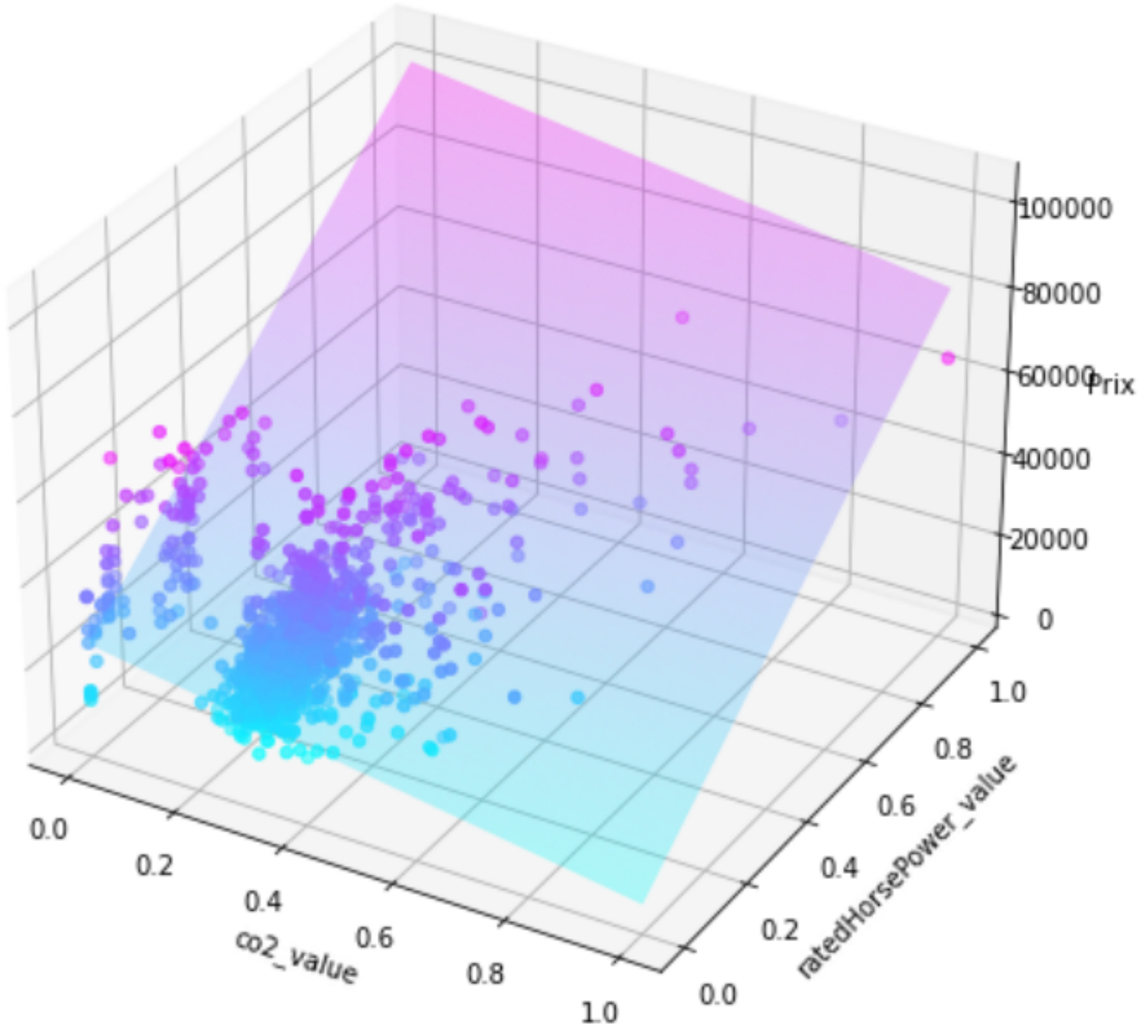
- Application du modèle sur l'ensemble des variables

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Création de l'objet de régression linéaire
reg = LinearRegression()

# Entraînement du modèle sur l'ensemble d'entraînement avec la fonction fit()
reg.fit(X_train[features], y_train)

# Prédiction des prix sur l'ensemble de test avec la fonction predict()
y_pred = reg.predict(X_test[features])

# Calcul du coefficient de détermination R² avec la fonction r2_score
r2 = r2_score(y_test, y_pred)
```

Résultats

RegLinear Model R-squared:
0.60568779028342

Application du modèle de Régression linéaire multiple sur les variables "co2-value" et "ratedHorsePower_value"



Résultats

RegLinear Model R-squared :

0.3464890374752

Interface Web

plotly

React

scikit learn

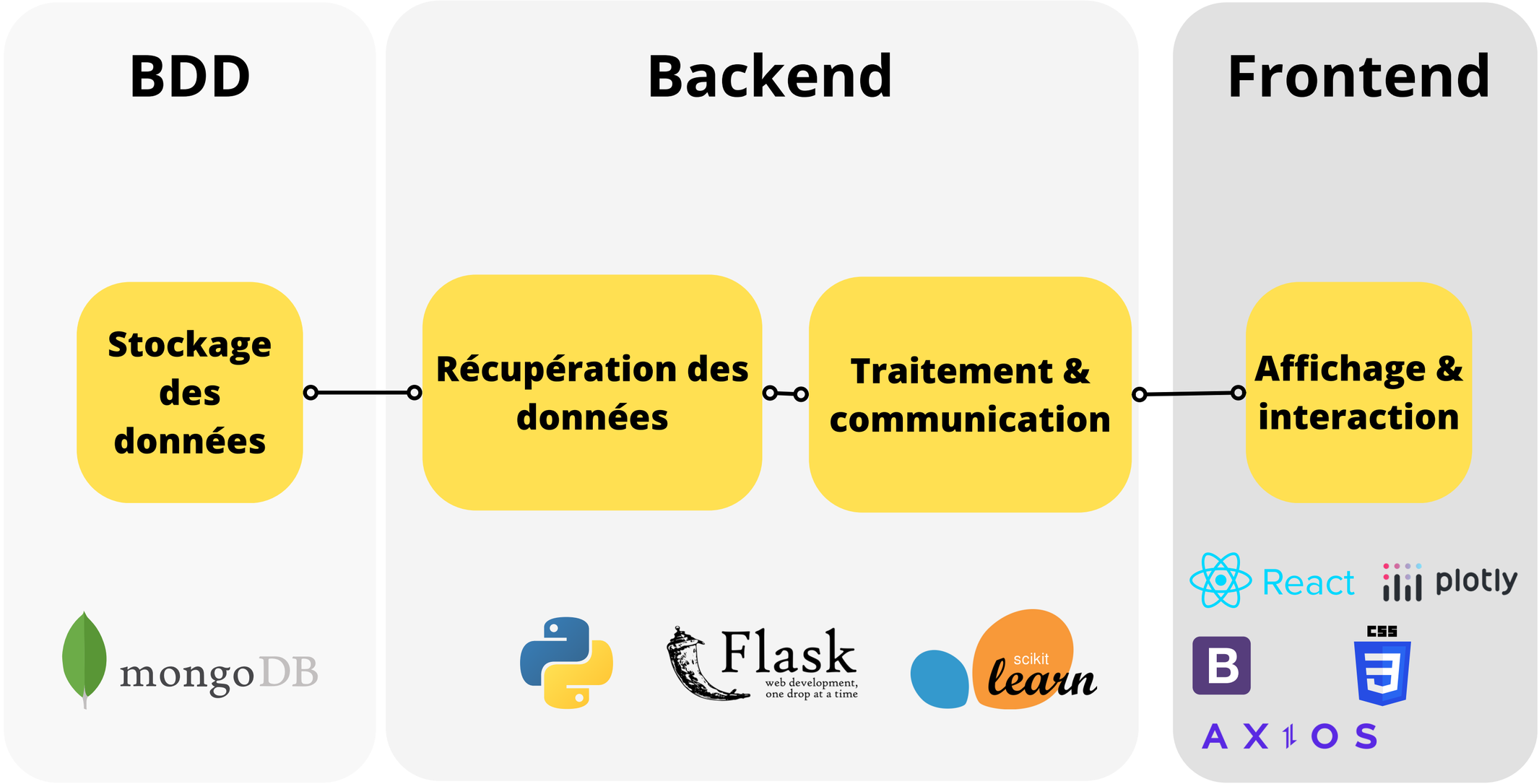
Flask
web development,
one drop at a time

mongoDB

B

CSS

AXIOS


BDD
Backend
Frontend
Stockage des données
Récupération des données
Traitement & communication
Affichage & interaction
mongoDB
Flask
web development, one drop at a time
scikit learn
React
plotly
AXIOS


825963
996801
00:01:16:21
DEMO
94.05%
999724

Questions ?