Prévision du prix de l'immobilier à l'aide du machine learning
XGBOOST REGRESSION

ndohmoise@gmail.com
ESI | M2R

July 2, 2022

1 Codage

1.1 Importation des bibliothèques nécessaires

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

1.2 Récupération de la base de données

```
#lecture de la base de donn es en ligne
house_dataset = sklearn.datasets.load_boston()
print(house_dataset)
```

1.3 Chargement des données dans le DataFrame

```
# Loading the dataset to a Pandas DataFrame
house_dataframe = pd.DataFrame
        (house_dataset.data, columns = house_dataset.feature_names)
```

1.4 Affichage des données en tête

```
house_dataframe.head()
house_dataframe['price'] = house_dataset.target
house_dataframe.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

1.5 Déterminons le nombre de lignes et de colonnes

```
house_dataframe.shape()
```

Nombre de lignes = 506
Nombre de colonnes = 14

1.6 Cherchons s'il y'a des valeurs nuls

```
house_dataframe.isnull().sum()
```

1.7 Détermination des caractéristiques

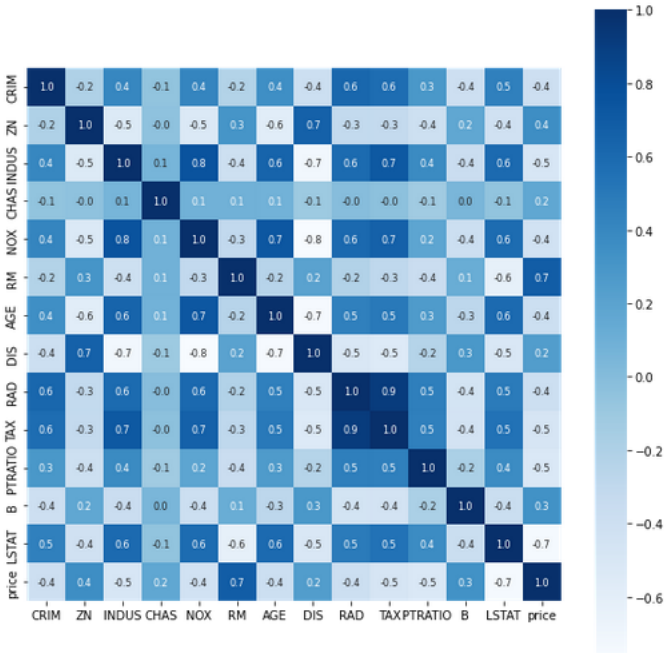
```
house_dataframe.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRAT
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.45
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.60(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.40(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.05(
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.20(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.00(

1.8 Déterminons le taux de corrélation de chaque variable

```
#Détermination de la corrélation
correlation = house_dataframe.corr()

#Affichage dans une matrice de données
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',
annot=True, annot_kws={'size':8}, cmap='Blues')
```



1.9 Découpage des données (entraînements et tests)

```
X = house_dataframe.drop(['price'], axis=1)
Y = house_dataframe['price']
print(X)
print(Y)
X_train, X_test, Y_train, Y_test =
train_test_split(X, Y, test_size = 0.2, random_state = 2)

print(X.shape, X_train.shape, X_test.shape)
```

Donnée de base : $(506, 13)$
Données d'entraînements : $(404, 13)$
Données de tests : $(102, 13)$

1.10 Mise en place du modèle d'entraînement

```
model = XGBRegressor()
model.fit(X_train,Y_train)
```

1.11 Mise en place du modèle de prédiction

```
training_data_prediction = model.predict(X_train)
print(training_data_prediction)
```

1.12 Évaluation du modèle

```
# R squared error
score_1 = metrics.r2_score(Y_train, training_data_prediction)

# Mean Absolute Error
score_2 = metrics.mean_absolute_error(Y_train, training_data_prediction)
```

```
print("R squared error : ", score_1)
print('Mean Absolute Error : ', score_2)
```

RSE = 0.9999948236320982 MAE = 0.0145848437110976

1.13 Tracer de la droite de régression obtenue

```
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Prix réels")
plt.ylabel("Prix prédits")
plt.title("Prix réel vs prix prévu")
plt.show()
```