Dataset - information

```
# Exemple : Stochastic Gradient Descent Regressor avec plusieurs variables

#lien : https://www.kaggle.com/datasets/rajacsp/toronto-apartment-price

"""
Bedroom - How many bedrooms available
Bathroom - How many bathrooms available
Den - Whether den is available or not
Address - Location
Lat - Lattitude
Long - Longitude
Price - Apartment Rental price per month in CAD
"""
```

```
'\nBedroom - How many bedrooms available\nBathroom - How many bathrooms available\nDen
- Whether den is available or not\nAddress - Location\nLat - Lattitude\nLong - Longitud
e\nPrice - Apartment Rental price per month in CAD\n'
```

Importer les bibliothèques

```
# importer les bibliothèques
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

Importer le Dataset

```
# importer le dataset
dataset = pd.read_csv("/content/dataset_LR_Toronto_apartment_rentals_2018.csv")
dataset
```

	Bedroom	Bathroom	Den	Address	Lat	Long	Price
0	2.0	2.0	0.0	3985 Grand Park Drive, 3985 Grand Park Dr, Mis...	43.581639	-79.648193	2450.0
1	1.0	1.0	1.0	361 Front St W, Toronto, ON M5V 3R5, Canada	43.643051	-79.391643	2150.0
2	1.0	1.0	0.0	89 McGill Street, Toronto, ON, M5B 0B1	43.660605	-79.378635	1950.0
3	2.0	2.0	0.0	10 York Street, Toronto, ON, M5J 0E1	43.641087	-79.381405	2900.0
4	1.0	1.0	0.0	80 St Patrick St, Toronto, ON M5T 2X6, Canada	43.652487	-79.389622	1800.0
...	...	...	...	...	...	...	...
1119	3.0	1.0	0.0	, L7S 1R7, Burlington, ON	43.325233	-79.802182	3000.0
1120	1.0	1.0	0.0	, oakville L6M3V5 ON, Canada	43.445426	-79.736833	1200.0
1121	1.0	1.0	0.0	Upper Beaches, Toronto,	43.683386	-79.309409	1800.0

Next steps: View recommended plots

```
dataset.info()
```

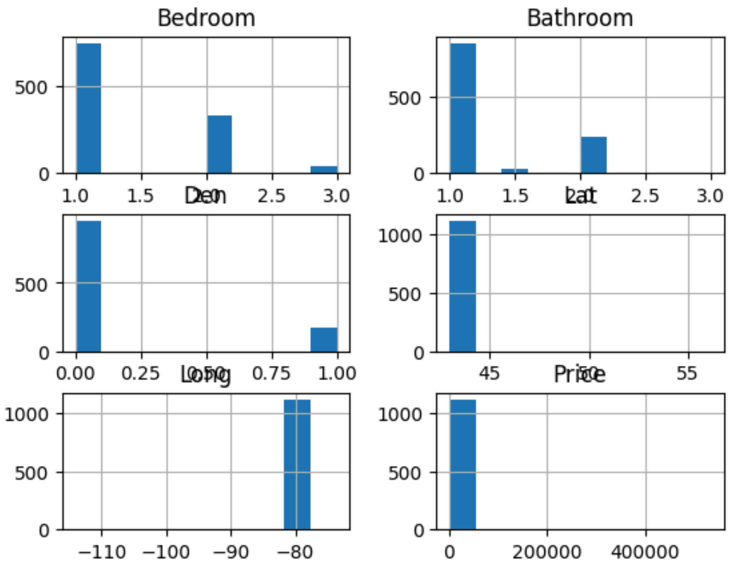
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1124 entries, 0 to 1123
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
 ---  ------    --------------  -----  
 0   Bedroom   1124 non-null   float64
 1   Bathroom  1124 non-null   float64
 2   Den       1124 non-null   float64
 3   Address   1124 non-null   object 
 4   Lat       1124 non-null   float64
 5   Long      1124 non-null   float64
 6   Price     1124 non-null   float64
dtypes: float64(6), object(1)
memory usage: 61.6+ KB
```

```
dataset.describe()
```

	Bedroom	Bathroom	Den	Lat	Long	Price
count	1124.000000	1124.000000	1124.000000	1124.000000	1124.000000	1124.000000
mean	1.370107	1.237544	0.153025	43.703532	-79.500326	3627.912811
std	0.553493	0.431997	0.360172	0.692689	1.760654	27530.542058
min	1.000000	1.000000	0.000000	42.985767	-114.082215	65.000000
25%	1.000000	1.000000	0.000000	43.641355	-79.414319	1759.250000
50%	1.000000	1.000000	0.000000	43.650560	-79.387295	2100.000000
75%	2.000000	1.000000	0.000000	43.666613	-79.377198	2500.000000
max	3.000000	3.000000	1.000000	56.130366	-73.576385	535000.000000

Visualisation des données

```
# Mettre le résultat de toutes les colonnes dans un histogramme - matplotlib
dataset.hist()
plt.show()
```



Prétraitement

```
# 1 - Les données manquantes (OK)
```

```
# 2 - Régularisation des données
# 2-1 Supprimer le signe '$' depuis la collonne 'Price' et le ''.00'
dataset['house_price'] = dataset['Price'].apply(lambda x:  str(x)[1:6])
dataset
```

	Bedroom	Bathroom	Den	Address	Lat	Long	Price	house_price
0	2.0	2.0	0.0	3985 Grand Park Drive, 3985 Grand Park Dr, Mis...	43.581639	-79.648193	2450.0	450.0
1	1.0	1.0	1.0	361 Front St W, Toronto, ON M5V 3R5, Canada	43.643051	-79.391643	2150.0	150.0
2	1.0	1.0	0.0	89 McGill Street, Toronto, ON, M5B 0B1	43.660605	-79.378635	1950.0	950.0
				10 York Street,				

Next steps: View recommended plots

```
# 2-2 Supprimer la virgule
dataset['house_price'] = dataset['house_price'].str.replace(',','')
dataset
```

	Bedroom	Bathroom	Den	Address	Lat	Long	Price	house_price
0	2.0	2.0	0.0	3985 Grand Park Drive, 3985 Grand Park Dr, Mis...	43.581639	-79.648193	2450.0	450.0
1	1.0	1.0	1.0	361 Front St W, Toronto, ON M5V 3R5, Canada	43.643051	-79.391643	2150.0	150.0
2	1.0	1.0	0.0	89 McGill Street, Toronto, ON, M5B 0B1	43.660605	-79.378635	1950.0	950.0
				10 York Street,				

Next steps: View recommended plots

```
# 3 - La sélection des données :
# on garde que les colonnes : Bedroom, Bathroom, Den, Price
dataset = dataset.loc[: , ['Bedroom', 'Bathroom', 'Den', 'house_price']]
#dataset = dataset.loc[: , ['Bedroom', 'house_price']]
dataset
```

	Bedroom	Bathroom	Den	house_price
0	2.0	2.0	0.0	450.0
1	1.0	1.0	1.0	150.0
2	1.0	1.0	0.0	950.0
3	2.0	2.0	0.0	900.0
4	1.0	1.0	0.0	800.0
...	...	...	...	...
1119	3.0	1.0	0.0	000.0
1120	1.0	1.0	0.0	200.0
1121	1.0	1.0	0.0	800.0
1122	2.0	1.0	0.0	200.0
1123	1.0	1.0	0.0	150.0

1124 rows × 4 columns

Next steps: View recommended plots

Apprentissage

```
# Préciser les X et Y
X = dataset.iloc[:,:-1] # X contient toutes les colonnes sauf la dernière
Y = dataset.iloc[:,-1] # Y présente la dernière colonne
```

```
# sélectionner un algorithme (estimateur)
model = SGDRegressor(alpha=0.001,max_iter=1000)
```

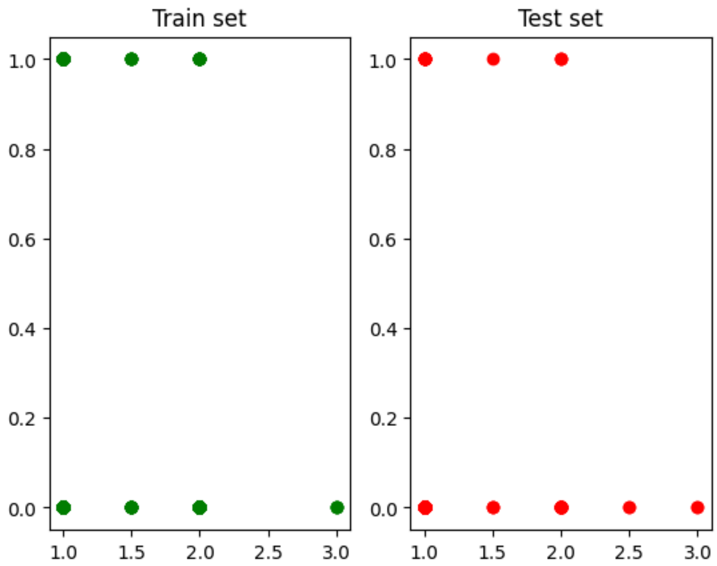
```
# slpit dataset (test et train)
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, random_state=2)

print('Train set:', X_train.shape)
print('Test set:', X_test.shape)
```

```
Train set: (899, 3)
Test set: (225, 3)
```

```
# Visualizer dataset après split
plt.figure()
plt.subplot(121)
plt.scatter(X_train.iloc[:,1], X_train.iloc[:,2], c='green')
plt.title('Train set')
plt.subplot(122)
plt.scatter(X_test.iloc[:,1], X_test.iloc[:,2], c='red')
plt.title('Test set')
```

```
Text(0.5, 1.0, 'Test set')
```



```
# entrainer le modèle sur les données X, Y
model.fit(X_train ,Y_train) #
```

```
▾      SGDRegressor
SGDRegressor(alpha=0.001)
```

```
# évaluer le modèle
model.score(X_test, Y_test)
```

```
-0.0655973039638722
```

```
# utiliser le modèle
prediction = model.predict(X_test)
prediction
```

```
array([615.47537367, 607.79003983, 496.24304919, 607.79003983,
       503.92838303, 503.92838303, 392.38139239, 607.79003983,
       607.79003983, 503.92838303, 607.79003983, 469.9292381 ,
       392.38139239, 503.92838303, 503.92838303, 607.79003983,
       607.79003983, 607.79003983, 607.79003983, 396.22405931,
       462.24390426, 607.79003983, 503.92838303, 496.24304919,
       462.24390426, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 503.92838303, 400.06672623, 503.92838303,
       496.24304919, 503.92838303, 503.92838303, 496.24304919,
       607.79003983, 607.79003983, 607.79003983, 503.92838303,
       607.79003983, 607.79003983, 607.79003983, 462.24390426,
       607.79003983, 503.92838303, 607.79003983, 607.79003983,
       503.92838303, 462.24390426, 503.92838303, 503.92838303,
       384.69605855, 607.79003983, 607.79003983, 496.24304919,
       611.63270675, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 462.24390426, 607.79003983, 607.79003983,
       503.92838303, 503.92838303, 503.92838303, 462.24390426,
       400.06672623, 607.79003983, 607.79003983, 607.79003983,
       462.24390426, 466.08657118, 496.24304919, 607.79003983,
       503.92838303, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 503.92838303, 607.79003983, 607.79003983,
       503.92838303, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 607.79003983, 503.92838303, 607.79003983,
       496.24304919, 607.79003983, 607.79003983, 607.79003983,
       503.92838303, 607.79003983, 469.9292381 , 607.79003983,
       607.79003983, 607.79003983, 503.92838303, 607.79003983,
       607.79003983, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 607.79003983, 496.24304919, 503.92838303,
       607.79003983, 607.79003983, 503.92838303, 607.79003983,
       462.24390426, 607.79003983, 496.24304919, 607.79003983,
       496.24304919, 607.79003983, 503.92838303, 503.92838303,
       607.79003983, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 462.24390426, 607.79003983, 503.92838303,
       503.92838303, 503.92838303, 496.24304919, 496.24304919,
       507.77104995, 503.92838303, 392.38139239, 607.79003983,
       607.79003983, 462.24390426, 607.79003983, 496.24304919,
       607.79003983, 607.79003983, 503.92838303, 496.24304919,
       607.79003983, 607.79003983, 503.92838303, 607.79003983,
       496.24304919, 500.08571611, 607.79003983, 607.79003983,
       503.92838303, 496.24304919, 462.24390426, 496.24304919,
       607.79003983, 462.24390426, 496.24304919, 496.24304919,
       607.79003983, 607.79003983, 607.79003983, 496.24304919,
       607.79003983, 607.79003983, 607.79003983, 607.79003983,
       503.92838303, 384.69605855, 607.79003983, 496.24304919,
       462.24390426, 607.79003983, 607.79003983, 496.24304919,
       607.79003983, 607.79003983, 607.79003983, 607.79003983,
       462.24390426, 607.79003983, 607.79003983, 607.79003983,
       388.53872547, 607.79003983, 496.24304919, 496.24304919,
       607.79003983, 607.79003983, 607.79003983, 607.79003983,
       607.79003983, 496.24304919, 462.24390426, 607.79003983,
       462.24390426, 462.24390426, 607.79003983, 462.24390426,
       607.79003983, 607.79003983, 607.79003983, 607.79003983,
       462.24390426, 496.24304919, 607.79003983, 503.92838303,
       607.79003983, 607.79003983, 469.9292381 , 607.79003983,
       607.79003983, 607.79003983, 607.79003983, 462.24390426,
       607.79003983, 392.38139239, 503.92838303, 503.92838303,
       607.79003983])
```

```
# les paramètres de la fonction :  Theta1
model.coef_ # Theta1, Theta2, Theta3
```

```
array([-111.54699064,    7.68533384, -145.54613556])
```

```
# les paramètres de la fonction :  Theta0
model.intercept_ # Theta0
```

```
array([711.65169663])
```

```
len(Y_test)
```

```
225
```

```
len(prediction)
```