Start coding or generate with AI.

Mission 1: Exploration et préparation des données

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler
```

```
housing = fetch_california_housing(as_frame=True)
```

```
housing.data.head(5)
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

```
print(housing.DESCR)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:
    - MedInc        median income in block group
    - HouseAge      median house age in block group
    - AveRooms      average number of rooms per household
    - AveBedrms     average number of bedrooms per household
    - Population    block group population
    - AveOccup      average number of household members
    - Latitude      block group latitude
    - Longitude     block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surprisingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. rubric:: References
```

```
    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
      Statistics and Probability Letters, 33 (1997) 291-297
```

```
housing.target
```

	MedHouseVal
0	4.526
1	3.585
2	3.521
3	3.413
4	3.422
...	...
20635	0.781
20636	0.771
20637	0.923
20638	0.847
20639	0.894

20640 rows × 1 columns

dtype: float64

```
# Ajouter la target 'MedHouseVal' au DataFrame
housing.data["MedHouseVal"] = housing.target
```

```
housing.data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   MedInc       20640 non-null  float64
 1   HouseAge     20640 non-null  float64
 2   AveRooms     20640 non-null  float64
 3   AveBedrms    20640 non-null  float64
 4   Population   20640 non-null  float64
 5   AveOccup     20640 non-null  float64
 6   Latitude     20640 non-null  float64
 7   Longitude    20640 non-null  float64
 8   MedHouseVal  20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

```
#Verification de donnees manquantes

housing.data.isnull().sum()
```

	0
MedInc	0
HouseAge	0
AveRooms	0
AveBedrms	0
Population	0
AveOccup	0
Latitude	0
Longitude	0
MedHouseVal	0

dtype: int64

Pas de donnee manquante dans le jeu de donnee

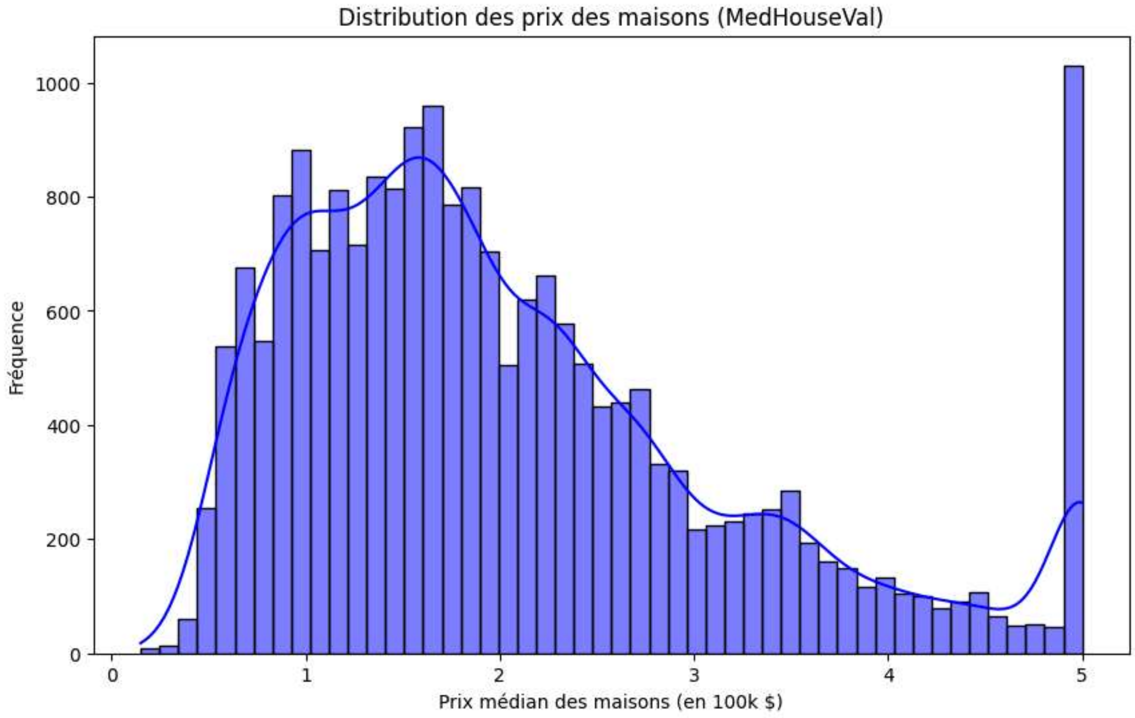
```
housing.data.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

Dans le dataset California Housing, certaines colonnes comme `AveRooms` et `AveBedrms` présentent des valeurs élevées qui, à première vue, pourraient sembler être des valeurs aberrantes. Toutefois, ces valeurs sont souvent dues à la manière dont les moyennes sont calculées par ménage (household) dans chaque zone géographique.

Ces observations sont conservées dans le dataset car elles apportent une information importante sur les zones à faible densité de population ou à usage saisonnier.

```
# Analyse exploratoire : Distribution de la variable cible
plt.figure(figsize=(10, 6))
sns.histplot(housing.data["MedHouseVal"], bins=50, kde=True, color='blue')
plt.title("Distribution des prix des maisons (MedHouseVal)")
plt.xlabel("Prix médian des maisons (en 100k $)")
plt.ylabel("Fréquence")
plt.show()
```



```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Séparer les features (X) et la cible (y)
X = housing.data.drop("MedHouseVal", axis=1)  # Toutes les colonnes sauf la cible
y = housing.data["MedHouseVal"]  # Variable cible

# Diviser les données en ensembles d'entraînement (80%) et de test (20%)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardisation des données
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Vérification des tailles des ensembles
print("Taille des ensembles après division :")
print(f"X_train : {X_train.shape}, X_test : {X_test.shape}")
print(f"y_train : {y_train.shape}, y_test : {y_test.shape}")
```

```
Taille des ensembles après division :
X_train : (16512, 8), X_test : (4128, 8)
y_train : (16512,), y_test : (4128,)
```