

Apprentissage automatique

TP 1 - Les données

Halim Djerroud

révision 1.0

— Exercices avec solutions —

Objectifs du TP

- Comprendre le processus de préparation des données.
- Appliquer les différentes techniques de nettoyage, transformation et visualisation des données.
- Mettre en pratique la séparation des jeux de données en ensemble d'entraînement, de validation et de test.
- Expérimenter avec des combinaisons de variables pour améliorer les performances des modèles d'apprentissage.

Outils et Environnement Le TP a été réalisé à l'aide des outils et environnements suivants :

- **Langage** : Python 3.x
- **Bibliothèques** : Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
- **IDE** : Jupyter Notebook / VS Code / Google Colab

Une des solutions consiste à installer anaconda <https://www.anaconda.com/download/success>. Une fois anaconda installé :

```
source ~/anaconda3/bin/activate
```

Créer un nouvel environnement nommé ml :

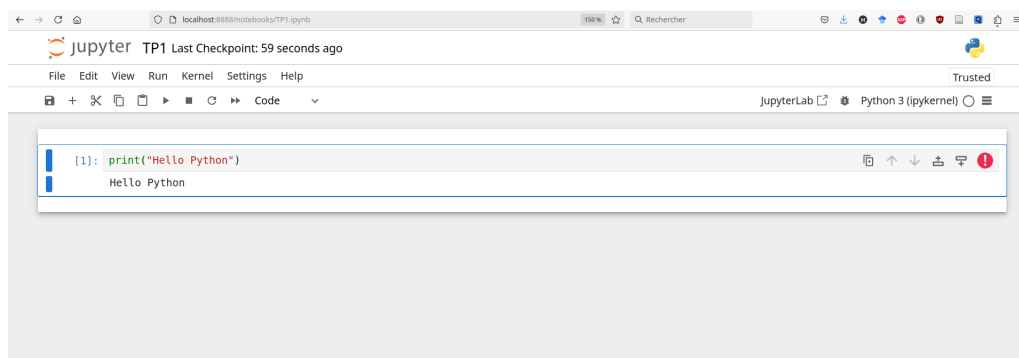
```
conda create --name ml # python=3.9
conda activate ml
(ml) conda install jupyter numpy scikit-learn matplotlib seaborn pandas
```

Lancer jupyter notebook :

```
cd /my/workspace
jupyter notebook
```

Dans un navigateur entrez l'adresse suivante :

<http://localhost:8888/tree>



1 Exercice : le dataset *California Housing*

Exercice inspiré du livre d'Aurélien Géron intitulé "Machine Learning avec Scikit-Learn", Chapitre 2 - 'Un projet de Machine Learning de bout en bout'.

1.1 Importer les Données

Utilisez la bibliothèque **Scikit-learn** pour charger le dataset California Housing et afficher les 5 premières lignes.

Solution :

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
df = data.frame
print(df.head())
```

Résultat :

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

1.2 Examiner la Structure des Données

1. Affichez les informations générales sur les colonnes.
2. Expliquer le contenu de chaque colonne et indiquer dans quelle unité est exprimée.
3. Lister les types de données et les valeurs manquantes.

Solution :

```
print(data.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.
```


1.2 Examiner la Structure des Données

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

En français :

MedInc : revenu médian dans le groupe de blocs
HouseAge : âge médian des maisons dans le groupe de blocs
AveRooms : nombre moyen de pièces par ménage
AveBedrms : nombre moyen de chambres par ménage
Population : population du groupe de blocs
AveOccup : nombre moyen de membres par ménage
Latitude : latitude du groupe de blocs
Longitude : longitude du groupe de blocs

Informations :

```
print(df.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
None
```

Description :

```
print(df.describe())
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744
std	1.899822	12.585558	2.474173	0.473911	1132.462122
min	0.499900	1.000000	0.846154	0.333333	3.000000
25%	2.563400	18.000000	4.440716	1.006079	787.000000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000
max	15.000100	52.000000	141.909091	34.066667	35682.000000

	AveOccup	Latitude	Longitude	MedHouseVal
count	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.070655	35.631861	-119.569704	2.068558
std	10.386050	2.135952	2.003532	1.153956
min	0.692308	32.540000	-124.350000	0.149990
25%	2.429741	33.930000	-121.800000	1.196000
50%	2.818116	34.260000	-118.490000	1.797000
75%	3.282261	37.710000	-118.010000	2.647250
max	1243.333333	41.950000	-114.310000	5.000010

Les valeurs manquantes

```
print(df.isnull().sum())
```


1.3 Utiliser le vrai dataset

Résultat :

```
MedInc      0
HouseAge    0
AveRooms    0
AveBedrms   0
Population  0
AveOccup    0
Latitude    0
Longitude   0
MedHouseVal 0
dtype: int64
```

1.3 Utiliser le vrai dataset

Dans cette section au lieu d'utiliser le dataset présent directement sur Scikit-Learn nous allons utiliser le dataset original. Télécharger et extraire le fichier *housing.tar.gz* sur le lien suivant :

<https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz>

Dans un Shell :

```
wget https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz
tar xfvz housing.tgz
```

Puis en Python :

```
df = pd.read_csv('housing.csv')
```

1.3.1 Étudier la différence

Examiner le nouveau dataset est déterminer les différences entre les deux.

Informations :

```
print(df.info())
```

Résultat :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   longitude              20640 non-null  float64
1   latitude               20640 non-null  float64
2   housing_median_age     20640 non-null  float64
3   total_rooms            20640 non-null  float64
4   total_bedrooms         20433 non-null  float64
5   population             20640 non-null  float64
6   households             20640 non-null  float64
7   median_income          20640 non-null  float64
8   median_house_value     20640 non-null  float64
9   ocean_proximity        20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
```

Les valeurs nulles :

```
print(df.isnull().sum())
```

Résultat :

```
longitude      0
latitude       0
housing_median_age  0
total_rooms    0
```


1.4 Explorer et Visualiser les Données

```
total_bedrooms    207
population         0
households         0
median_income      0
median_house_value 0
ocean_proximity    0
dtype: int64
```

Commentaire :

Toutes les variables sont numérique, excepté `ocean_proximity`. Elle est de type `object`.

```
df["ocean_proximity"].value_counts()
```

Résultat :

```
ocean_proximity
<1H OCEAN    9136
INLAND       6551
NEAR OCEAN   2658
NEAR BAY     2290
ISLAND        5
Name: count, dtype: int64
```

```
print(df.describe())
```

Résultat :

	longitude	latitude	housing_median_age	total_rooms
count	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081
std	2.003532	2.135952	12.585558	2181.615252
min	-124.350000	32.540000	1.000000	2.000000
25%	-121.800000	33.930000	18.000000	1447.750000
50%	-118.490000	34.260000	29.000000	2127.000000
75%	-118.010000	37.710000	37.000000	3148.000000
max	-114.310000	41.950000	52.000000	39320.000000

	total_bedrooms	population	households	median_income
count	20433.000000	20640.000000	20640.000000	20640.000000
mean	537.870553	1425.476744	499.539680	3.870671
std	421.385070	1132.462122	382.329753	1.899822
min	1.000000	3.000000	1.000000	0.499900
25%	296.000000	787.000000	280.000000	2.563400
50%	435.000000	1166.000000	409.000000	3.534800
75%	647.000000	1725.000000	605.000000	4.743250
max	6445.000000	35682.000000	6082.000000	15.000100

	median_house_value
count	20640.000000
mean	206855.816909
std	115395.615874
min	14999.000000
25%	119600.000000
50%	179700.000000
75%	264725.000000
max	500001.000000

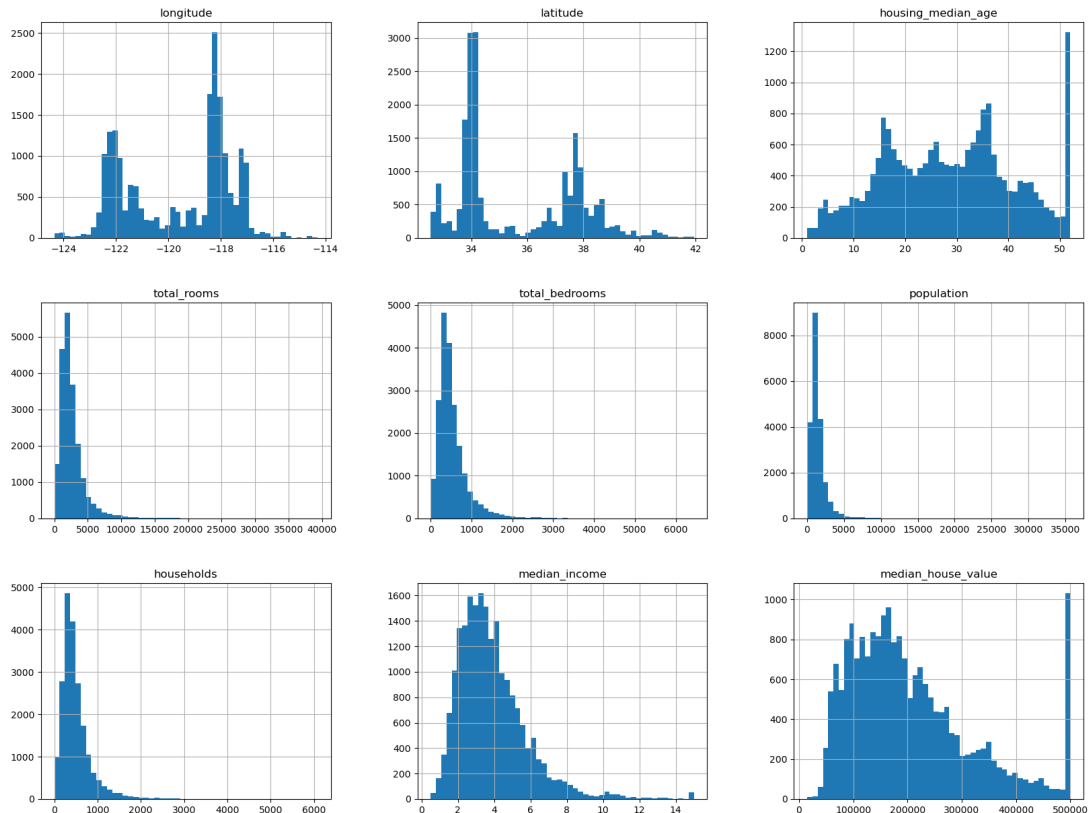
1.4 Explorer et Visualiser les Données

Créez des histogrammes et des nuages de points (scatter plots) pour les variables `MedInc`, `HouseAge` et `MedHouseVal`.

1.4 Explorer et Visualiser les Données

Solution :

```
import matplotlib.pyplot as plt
df.hist(bins=50, figsize=(20,15))
plt.show()
```



Commentaire :

- La variable *median_income* (revenu moyen) ne semble pas exprimée en \$. Après vérification, on vous dit que variable a été mise à l'échelle et plafonnée à 15 (plus exactement = 15.0001) pour les revenus les plus élevés et à 0.5 (plus exactement à 0.49999) pour les revenus les plus bas.
- L'âge moyen des habitations est également plafonné.

Commentaire :

Notez que certains histogrammes sont fortement dissymétriques, ils s'étendent bien plus loin à droite qu'à gauche. Certains algorithmes d'apprentissage peuvent avoir de mal à détecter des structurations.

1.4.1 Visualiser des données géographiques

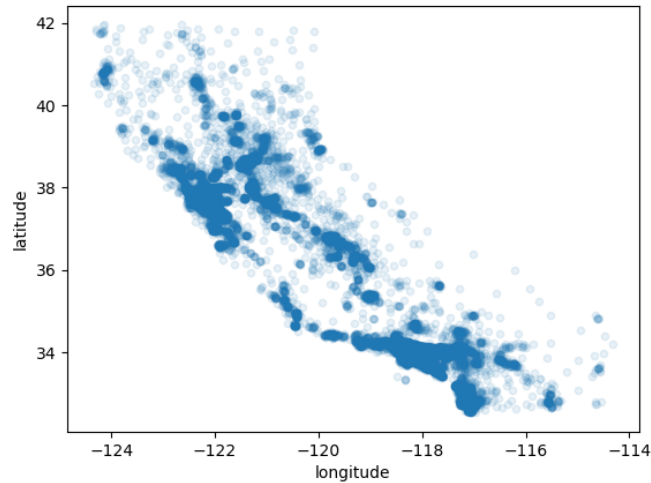
Utiliser la longitude et la latitude pour afficher la concentration des habitations.

Solution :

```
df.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
```


1.5 Recherche de corrélations

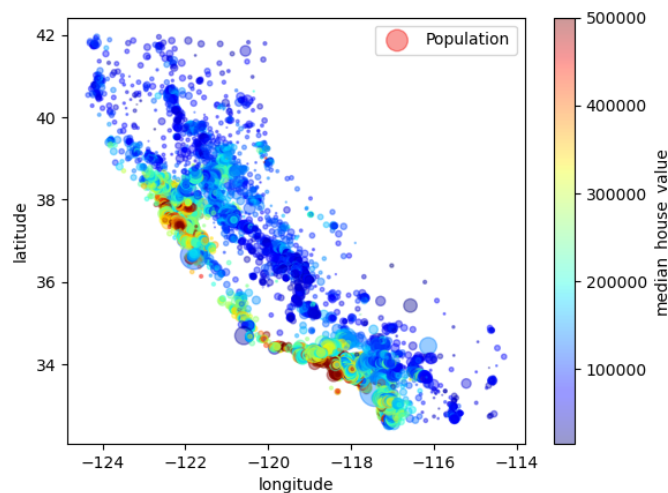
```
plt.savefig('geo.png')
plt.show()
```



Ajouter dans le même graphique la concentration des habitations et le prix de l'immobilier.

Solution :

```
df.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
        s=df["population"]/100, label="Population",
        c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True)
plt.legend()
plt.savefig('geo2.png')
plt.show()
```



1.5 Recherche de corrélations

Calculer le coefficient de corrélation entre chaque couple de variables.

1.6 Expérimenter avec des combinaisons de variables

Solution :

```
df_copy = df.copy()
df_copy.drop(columns=['ocean_proximity'], inplace=True)
corr_matrix = df_copy.corr()
print(corr_matrix)
```

Commentaire :

La colonne *ocean_proximity* est supprimée, car elle n'est pas possible d'effectuer de calcul sur une colonne de type chaîne de caractères.

Résultats :

	longitude	latitude	housing_median_age	total_rooms	\
longitude	1.000000	-0.924664	-0.108197	0.044568	
latitude	-0.924664	1.000000	0.011173	-0.036100	
housing_median_age	-0.108197	0.011173	1.000000	-0.361262	
total_rooms	0.044568	-0.036100	-0.361262	1.000000	
total_bedrooms	0.069608	-0.066983	-0.320451	0.930380	
population	0.099773	-0.108785	-0.296244	0.857126	
households	0.055310	-0.071035	-0.302916	0.918484	
median_income	-0.015176	-0.079809	-0.119034	0.198050	
median_house_value	-0.045967	-0.144160	0.105623	0.134153	

	total_bedrooms	population	households	median_income	\
longitude	0.069608	0.099773	0.055310	-0.015176	
latitude	-0.066983	-0.108785	-0.071035	-0.079809	
housing_median_age	-0.320451	-0.296244	-0.302916	-0.119034	
total_rooms	0.930380	0.857126	0.918484	0.198050	
total_bedrooms	1.000000	0.877747	0.979728	-0.007723	
population	0.877747	1.000000	0.907222	0.004834	
households	0.979728	0.907222	1.000000	0.013033	
median_income	-0.007723	0.004834	0.013033	1.000000	
median_house_value	0.049686	-0.024650	0.065843	0.688075	

	median_house_value
longitude	-0.045967
latitude	-0.144160
housing_median_age	0.105623
total_rooms	0.134153
total_bedrooms	0.049686
population	-0.024650
households	0.065843
median_income	0.688075
median_house_value	1.000000

```
import pandas as pd
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
atts = ["median_house_value", "median_income", "total_rooms", "housing_median_age"]
scatter_matrix(df[atts], figsize=(12, 8))
plt.savefig('matrix_corr.png')
plt.show()
```

1.6 Expérimenter avec des combinaisons de variables

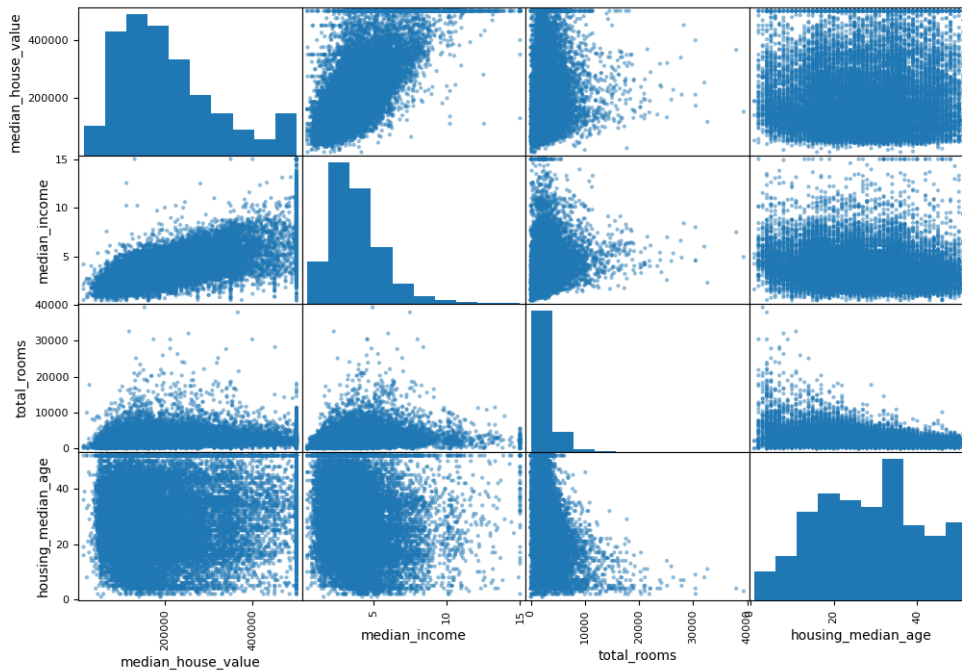
Le nombre total de pièces dans un district n'est pas très utile si vous ne connaissez pas le nombre de logements. Ce qu'on souhaite, c'est d'avoir le nombre de pièces par logement. De même le nombre total de chambres n'est pas très utile en lui-même, mais il sera utile de le comparer au nombre de pièces. Il est aussi intéressant de calculer le nombre de personnes par logement.

Créer de nouvelles colonnes qui expriment ces nouvelles données. Et examiner à nouveau la matrice de corrélation.

Solution :

```
df["rooms_per_household"] = df["total_rooms"] / df["households"]
df["bedrooms_per_room"] = df["total_bedrooms"] / df["total_rooms"]
df["population_per_household"] = df["population"] / df["households"]
```


1.7 Nettoyer les Données



```
df_copy = df.copy()
df_copy.drop(columns=['ocean_proximity'], inplace=True)
corr_matrix = df_copy.corr()
print(corr_matrix["median_house_value"].sort_values(ascending=False))
```

Résultats :

```
median_house_value    1.000000
median_income          0.688075
rooms_per_household    0.151948
total_rooms            0.134153
housing_median_age     0.105623
households             0.065843
total_bedrooms         0.049686
population_per_household -0.023737
population             -0.024650
longitude              -0.045967
latitude               -0.144160
bedrooms_per_room      -0.255880
Name: median_house_value, dtype: float64
```

Commentaire :

La nouvelle variable *bedrooms_per_room* est bien plus corrélée avec la valeur moyenne des habitations que le nombre total de pièces ou de chambres. Les maisons avec un faible ratio chambres/pièces ont tendance à être plus chères.

1.7 Nettoyer les Données

Pour les variables manquantes, trois options s'offrent à nous :

1. Supprimer les districts correspondants.
2. Supprimer cette variable.
3. Mettre la valeur à : (zéro, la moyenne, la médiane, etc.).

1.8 Gérer les variables qualitatives

Solution :

```
#option 1
df.dropna(subset=["total_bedrooms"])
#option 2
df.drop("total_bedrooms", axis=1)
#option 3
median = df["total_bedrooms"].median()
df["total_bedrooms"].fillna(median, inplace=True)
```

Commentaire :

Scikit-Learn fournit la classe *Imputer* qui permet de traiter les valeurs manquantes. Utiliser cette classe pour réaliser chacune des options précédentes.

1.8 Gérer les variables qualitatives

Trouver un moyen de gérer la colonne *ocean_proximity*. Deux options s'offrent à nous :

1. Label encoding
2. One hot encoding

Solution :

```
#Option 1
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
housing_cat = df["ocean_proximity"]
housing_cat_encoded = encoder.fit_transform(housing_cat)
housing_cat_encoded

#Option 2
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
housing_cat_1hot = encoder.fit_transform(housing_cat_encoded.reshape(-1,1))
housing_cat_1hot
```

1.9 Recalibrage des variables

Il existe deux façons de recalibrer les variables :

1. Transformation (*min-max*) ou *Normalisation* : Les valeurs sont décalées afin qu'elles se situent entre 0 et 1.

$$X_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \in [0, 1]$$

2. *Normalisation standard* ou simplement *Standardisation* : tout d'abord, on les soustrait à la valeur moyenne (de sorte que les valeurs normalisées auront toujours une moyenne nulle), puis à les diviser par l'écart-type afin que la distribution qui en résulte ait une variance = 1.

$$X_{standard} = \frac{X - \mu}{\sigma}$$

Utiliser les pipelines offerts par Scikit-Learn pour réaliser une normalisation.

1.10 Séparer les Jeux d'Entraînement, de Validation et de Test

Divisez les données en ensembles d'entraînement, de validation et de test.

Solution :

```
from sklearn.model_selection import train_test_split

X = df.drop(columns=['MedHouseVal'])
y = df['MedHouseVal']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```


2 Exercice : Le dataset IRIS

2.1 Importer les Données

Utilisez la bibliothèque **Scikit-learn** pour charger le dataset Iris. Puis réaliser l'étape de préparation de données en vue de les soumettre à un algorithme d'apprentissage.

1. Examiner la structure des Données.
2. Explorer et visualiser les Données
3. Expérimenter avec des combinaisons de variables
4. Rechercher des corrélations
5. Rédiger une conclusion sur les données
6. Nettoyer les données
7. Gérer les variables qualitatives
8. Recalibrer des variables
9. Séparer les jeux d'entraînement, de validation et de test

3 Dataset : Titanic (Survie des passagers)

Le dataset Titanic est l'un des jeux de données les plus utilisés pour l'apprentissage des concepts de classification. Ce dataset est disponible sur Kaggle.

Réaliser les étapes suivantes :

1. Examiner la structure des Données.
2. Explorer et visualiser les Données
3. Expérimenter avec des combinaisons de variables
4. Rechercher des corrélations
5. Rédiger une conclusion sur les données
6. Nettoyer les données
7. Gérer les variables qualitatives
8. Recalibrer des variables
9. Séparer les jeux d'entraînement, de validation et de test

4 Dataset qualité de l'air à paris

4.1 Importer les Données

Téléchargez le fichier `qualite_air.csv` depuis [data.gouv.fr](https://www.data.gouv.fr/fr/datasets/indice-de-la-qualite-de-lair-quotidien-par-commune-indice-atmo/) et importez-le à l'aide de Pandas. <https://www.data.gouv.fr/fr/datasets/indice-de-la-qualite-de-lair-quotidien-par-commune-indice-atmo/>

Réaliser les étapes suivantes :

1. Examiner la structure des Données.
2. Explorer et visualiser les Données
3. Expérimenter avec des combinaisons de variables
4. Rechercher des corrélations
5. Rédiger une conclusion sur les données
6. Nettoyer les données
7. Gérer les variables qualitatives
8. Recalibrer des variables
9. Séparer les jeux d'entraînement, de validation et de test