

Untitled42

November 3, 2025

```
[3]: import pandas as pd
df = pd.DataFrame(
{"a" : [4, 5, 6],
"b" : [7, 8, 9],
"c" : [10, 11, 12]},
index = [1, 2, 3])
print(df)
```

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
[7]: import pandas as pd
data=pd.read_csv("cars.csv")
print(data)
```

	Unnamed: 0	car_name	brand	model	vehicle_age	\
0	0	Maruti Alto	Maruti	Alto	9	
1	1	Hyundai Grand	Hyundai	Grand	5	
2	2	Hyundai i20	Hyundai	i20	11	
3	3	Maruti Alto	Maruti	Alto	9	
4	4	Ford Ecosport	Ford	Ecosport	6	
...	...	...	...	...	...	
15406	19537	Hyundai i10	Hyundai	i10	9	
15407	19540	Maruti Ertiga	Maruti	Ertiga	2	
15408	19541	Skoda Rapid	Skoda	Rapid	6	
15409	19542	Mahindra XUV500	Mahindra	XUV500	5	
15410	19543	Honda City	Honda	City	2	

	km_driven	seller_type	fuel_type	transmission_type	mileage	engine	\
0	120000	Individual	Petrol	Manual	19.70	796	
1	20000	Individual	Petrol	Manual	18.90	1197	
2	60000	Individual	Petrol	Manual	17.00	1197	
3	37000	Individual	Petrol	Manual	20.92	998	
4	30000	Dealer	Diesel	Manual	22.77	1498	
...	...	...	...	...	...	...	
15406	10723	Dealer	Petrol	Manual	19.81	1086	
15407	18000	Dealer	Petrol	Manual	17.50	1373	

15408	67000	Dealer	Diesel	Manual	21.14	1498
15409	3800000	Dealer	Diesel	Manual	16.00	2179
15410	13000	Dealer	Petrol	Automatic	18.00	1497

	max_power	seats	price
0	46.30	5	120000
1	82.00	5	550000
2	80.00	5	215000
3	67.10	5	226000
4	98.59	5	570000
...	...	...	...
15406	68.05	5	250000
15407	91.10	7	925000
15408	103.52	5	425000
15409	140.00	7	1225000
15410	117.60	5	1200000

[15411 rows x 14 columns]

```
[11]: my_data=pd.DataFrame(data,columns=['price','brand'])
      print(my_data)
```

	price	brand
0	120000	Maruti
1	550000	Hyundai
2	215000	Hyundai
3	226000	Maruti
4	570000	Ford
...	...	...
15406	250000	Hyundai
15407	925000	Maruti
15408	425000	Skoda
15409	1225000	Mahindra
15410	1200000	Honda

[15411 rows x 2 columns]

```
[19]: #Using head() method with an argument which helps us to restrict the number of
      ↪initial records that should be displayed
      data.head(n=2)
      #Using.tail() method with an argument which helps us to restrict the number of
      ↪initial records that should be displayed
      data.tail(n=2)
```

```
[19]: Unnamed: 0      car_name      brand      model      vehicle_age      km_driven  \
15409      19542  Mahindra XUV500  Mahindra  XUV500           5      3800000
15410      19543    Honda City    Honda    City           2       13000
```

	seller_type	fuel_type	transmission_type	mileage	engine	max_power	\
15409	Dealer	Diesel	Manual	16.0	2179	140.0	
15410	Dealer	Petrol	Automatic	18.0	1497	117.6	

	seats	price
15409	7	1225000
15410	5	1200000

```
[21]: #information about the data
print(data.describe())
#assigning the 'col_i' column as target
target=data['price']
data.head(n=2)
```

	Unnamed: 0	vehicle_age	km_driven	mileage	engine	\
count	15411.000000	15411.000000	1.541100e+04	15411.000000	15411.000000	
mean	9811.857699	6.036338	5.561648e+04	19.701151	1486.057751	
std	5643.418542	3.013291	5.161855e+04	4.171265	521.106696	
min	0.000000	0.000000	1.000000e+02	4.000000	793.000000	
25%	4906.500000	4.000000	3.000000e+04	17.000000	1197.000000	
50%	9872.000000	6.000000	5.000000e+04	19.670000	1248.000000	
75%	14668.500000	8.000000	7.000000e+04	22.700000	1582.000000	
max	19543.000000	29.000000	3.800000e+06	33.540000	6592.000000	

	max_power	seats	price
count	15411.000000	15411.000000	1.541100e+04
mean	100.588254	5.325482	7.749711e+05
std	42.972979	0.807628	8.941284e+05
min	38.400000	0.000000	4.000000e+04
25%	74.000000	5.000000	3.850000e+05
50%	88.500000	5.000000	5.560000e+05
75%	117.300000	5.000000	8.250000e+05
max	626.000000	9.000000	3.950000e+07

```
[21]: Unnamed: 0    car_name    brand    model    vehicle_age    km_driven    \
0          0    Maruti Alto    Maruti    Alto          9        120000
1          1  Hyundai Grand    Hyundai    Grand          5         20000
```

	seller_type	fuel_type	transmission_type	mileage	engine	max_power	seats	\
0	Individual	Petrol	Manual	19.7	796	46.3	5	
1	Individual	Petrol	Manual	18.9	1197	82.0	5	

	price
0	120000
1	550000

```
[25]: df=pd.read_csv('cars.csv')
df.to_csv('my_data.csv')
```

```
[32]: data_copy=data.copy()
train_set=data_copy.sample(frac=0.80,random_state=0)
test_set =data_copy.drop(train_set.index)
```

```
[40]: #Use 'pop' to extract the labels
train_set_labels=train_set.pop('price')
test_set_labels=test_set.pop('price')
print("train_set:\n",train_set)
print("train_set_labels:\n",train_set_labels)
```

```
train_set:
```

	Unnamed: 0	car_name	brand	model	vehicle_age	\
5122	6501	Renault KWID	Renault	KWID	1	
9792	12542	Hyundai i10	Hyundai	i10	11	
8379	10768	Tata Safari	Tata	Safari	8	
10718	13645	Honda City	Honda	City	6	
641	815	Maruti Swift	Maruti	Swift	2	
...	...	...	...	...	...	
5974	7635	Mahindra Bolero	Mahindra	Bolero	5	
8333	10709	Maruti Wagon R	Maruti	Wagon R	4	
11698	14849	Maruti Swift Dzire	Maruti	Swift Dzire	13	
2910	3697	Renault KWID	Renault	KWID	5	
6096	7787	Hyundai Verna	Hyundai	Verna	9	

	km_driven	seller_type	fuel_type	transmission_type	mileage	\
5122	1200	Individual	Petrol	Automatic	22.00	
9792	60000	Individual	Petrol	Manual	20.36	
8379	60000	Individual	Diesel	Manual	11.57	
10718	57000	Dealer	Diesel	Manual	26.00	
641	14499	Trustmark Dealer	Petrol	Manual	22.00	
...	...	...	...	...	...	
5974	76000	Dealer	Diesel	Manual	13.60	
8333	35000	Dealer	Petrol	Manual	21.79	
11698	110000	Individual	Diesel	Manual	19.30	
2910	35000	Individual	Petrol	Manual	25.17	
6096	96888	Dealer	Diesel	Automatic	22.32	

	engine	max_power	seats
5122	999	67.00	5
9792	1197	78.90	5
8379	2179	138.10	7
10718	1498	98.60	5
641	1197	81.80	5
...	...	...	...
5974	2523	63.00	9
8333	998	67.05	5
11698	1248	73.90	5
2910	799	53.30	5

```
6096      1582      126.30      5
```

```
[12329 rows x 13 columns]
```

```
train_set_labels:
```

```
5122      450000
9792      165000
8379      350000
10718     525000
641       535000
```

```
...
```

```
5974      590000
8333      350000
11698     325000
2910      250000
6096      550000
```

```
Name: price, Length: 12329, dtype: int64
```

```
[68]: from glob import glob

# Returns a list of pathnames in list files
pathnames = glob(r"C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/
↳Codes_LOG-POLAR_HMAX/HMAX_model/data/*")
#r"..." ensures that backslashes are not treated as escape characters.
for path in pathnames:
    print(path)
```

```
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\1.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\10.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\100.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\101.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\102.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\103.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\104.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\105.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\106.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\107.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\108.jpg
```



```

C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\88.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\89.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\9.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\90.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\91.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\92.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\93.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\94.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\95.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\96.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\97.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\98.jpg
C:/Users/Thinkpad/OneDrive/Documents/projectdanemarkitaly/Codes_LOG-
POLAR_HMAX/HMAX_model/data\99.jpg

```

```

[74]: from sklearn import datasets
      dat=datasets.load_breast_cancer()
      print("Examples=",dat.data.shape,"Labels=",dat.target.shape)

```

Examples= (569, 30) Labels= (569,)

```

[76]: from sklearn import datasets
      dat=datasets.fetch_20newsgroups(subset='train')
      from pprint import pprint
      pprint(list(dat.target_names))

```

```

['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',

```

```
'sci.crypt',
'sci.electronics',
'sci.med',
'sci.space',
'soc.religion.christian',
'talk.politics.guns',
'talk.politics.mideast',
'talk.politics.misc',
'talk.religion.misc']
```

```
[78]: from sklearn.datasets import fetch_openml
mice=fetch_openml(name='miceprotein',version=4)
mice.data.shape
```

```
[78]: (1080, 77)
```

```
[80]: mice.target.shape
```

```
[80]: (1080,)
```

```
[84]: import numpy as np
np.unique(mice.target)
```

```
[84]: array(['c-CS-m', 'c-CS-s', 'c-SC-m', 'c-SC-s', 't-CS-m', 't-CS-s',
't-SC-m', 't-SC-s'], dtype=object)
```

```
[86]: mice.url
```

```
[86]: 'https://www.openml.org/d/40966'
```

```
[88]: mice.details['version']
```

```
[88]: '4'
```

```
[98]: from sklearn.datasets import make_classification
X, y = make_classification(
    n_samples=10000,    # nombre total d'exemples
    n_features=5,       # 5 variables (dimensions)
    flip_y=0.1,         # 10% des labels aléatoirement inversés (introduit du
↳ bruit)
    class_sep=0.1       # séparation très faible entre les classes (données peu
↳ séparables)
)
```

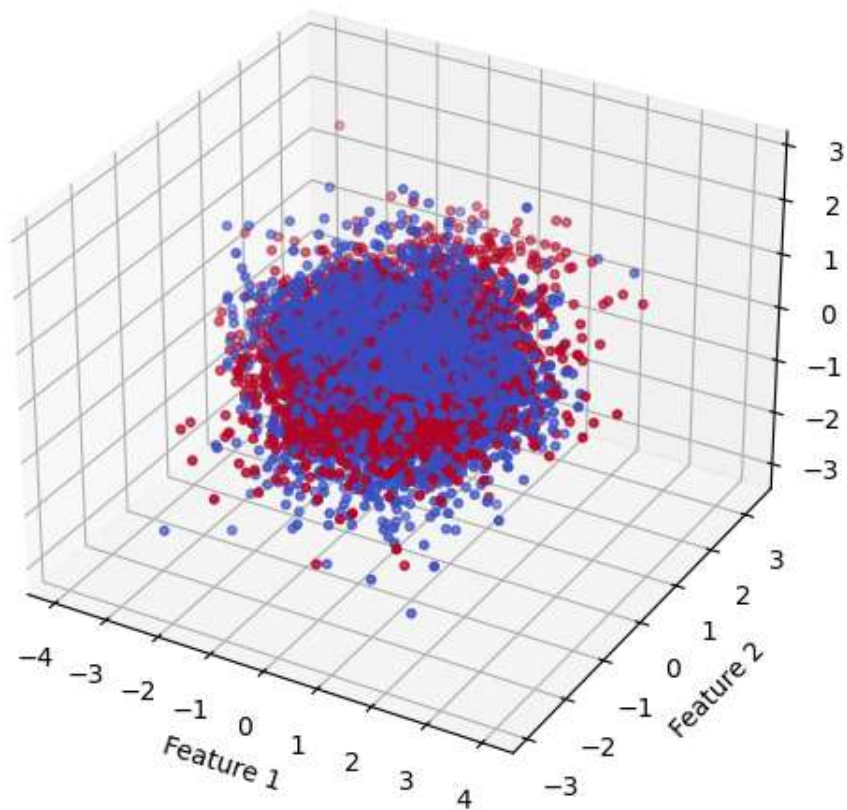
```
[100]: print("X shape =", X.shape)
print("len y =", len(y))
print(y)
```

```
X shape = (10000, 5)
len y = 10000
[1 1 0 ... 0 1 1]
```

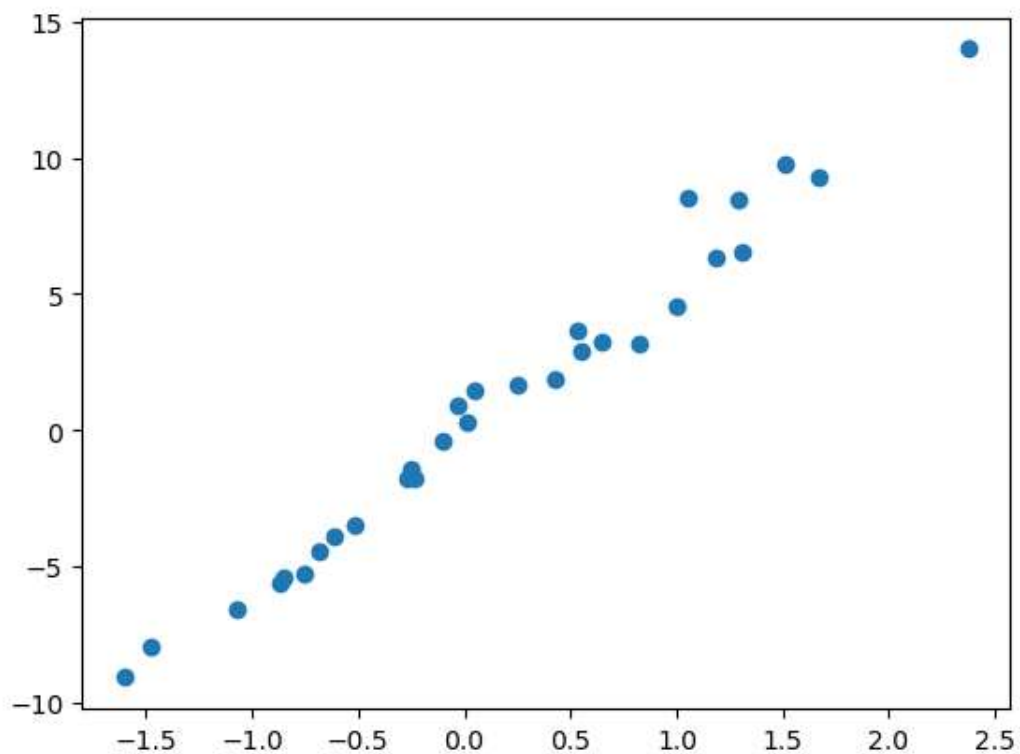
```
[114]: import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Visualisation 3D
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0], X[:,1], X[:,2], c=y, cmap='coolwarm', s=10)
ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.set_zlabel('Feature 3')
plt.title('Visualisation 3D du jeu de données')
plt.show()
```

Visualisation 3D du jeu de données



```
[118]: from sklearn import datasets
from matplotlib import pyplot as plt
x,y=datasets.make_regression(
    n_samples=30,
    n_features=1,
    noise=0.8)
plt.scatter(x,y)
plt.show()
```



```
[122]: from sklearn.datasets import make_blobs
from matplotlib import pyplot as plt
import pandas as pd

# Génération des données
X, y = make_blobs(n_samples=200, centers=4, n_features=2, random_state=42)

# Transformation en DataFrame pour faciliter la manipulation
df = pd.DataFrame({
    'x1': X[:, 0],
    'x2': X[:, 1],
    'label': y
})
```

```

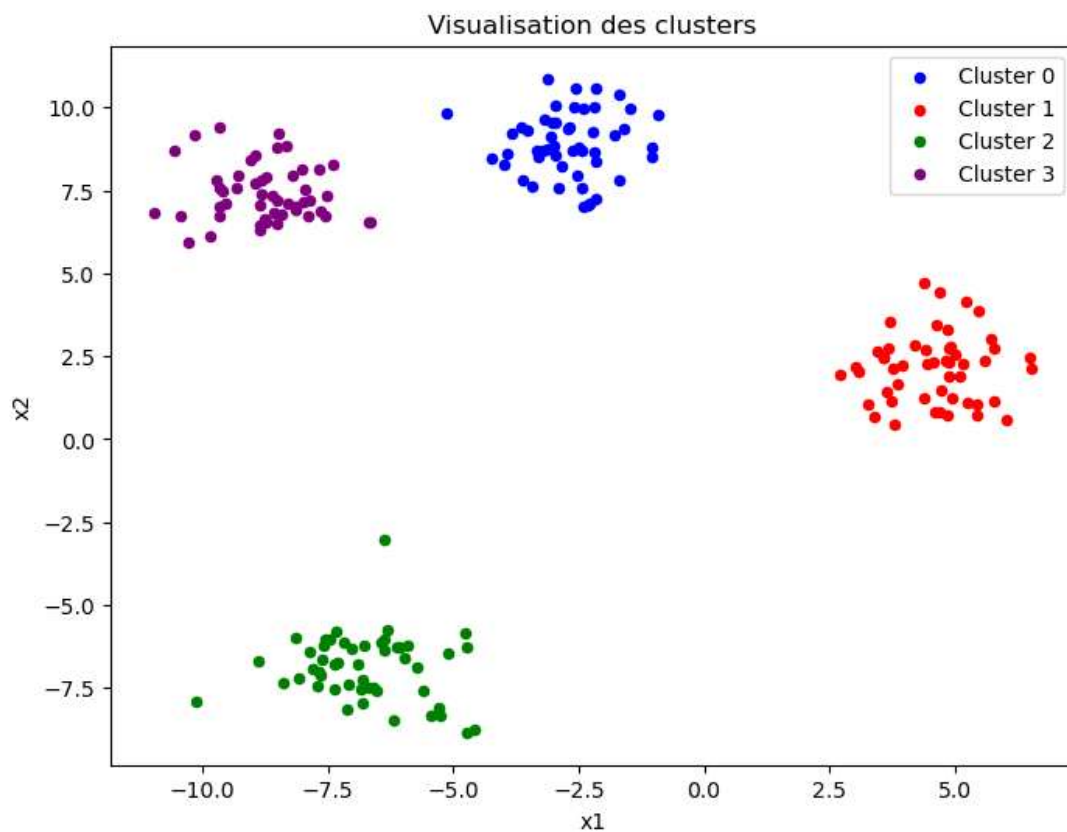
# Grouper les données par cluster
groups = df.groupby('label')

# Création du graphique
fig, ax = plt.subplots(figsize=(8,6))
colors = ["blue", "red", "green", "purple"]

# Boucle pour tracer chaque groupe
for idx, group in groups:
    group.plot(ax=ax, kind='scatter', x='x1', y='x2', label=f'Cluster {idx}',
    ↪color=colors[idx])

# Affichage
plt.title('Visualisation des clusters')
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()

```




```
[128]: import pandas as pd
import numpy as np
# dictionary of lists
dictionary = {"Name":["Alex", "Mike", "John", "Dave", "Joey"],
              "Height(m)": [1.75, 1.65, 1.73, np.nan, 1.82],
              "Test Score": [70, np.nan, 8, 62, 73]}
# creating a dataframe from dict
df = pd.DataFrame(dictionary)
# using isnull() function this function return dataframe of Boolean values
# which are True for NaN values.
print(df.isnull())
print("SUM : \n",df.isnull().sum())
```

	Name	Height(m)	Test Score
0	False	False	False
1	False	False	True
2	False	False	False
3	False	True	False
4	False	False	False

SUM :

Name	0
Height(m)	1
Test Score	1

dtype: int64

```
[136]: # Convertir en numérique
df['Test Score'] = pd.to_numeric(df['Test Score'], errors='coerce')
df['Height(m)'] = pd.to_numeric(df['Height(m)'], errors='coerce')

# Remplir les NaN par la moyenne
df['Test Score'] = df['Test Score'].fillna(df['Test Score'].mean())
df['Height(m)'] = df['Height(m)'].fillna(df['Height(m)'].mean())

# Supprimer lignes restantes avec NaN (optionnel)
df = df.dropna()
```

```
[138]: df
```

	Name	Height(m)	Test Score
0	Alex	1.7500	70.00
1	Mike	1.6500	53.25
2	John	1.7300	8.00
3	Dave	1.7375	62.00
4	Joey	1.8200	73.00

```
[140]: import pandas as pd
import numpy as np
```

```

# Création du dictionnaire avec des valeurs manquantes non standard
dictionary_1 = {
    "Name": ["Alex", "Mike", "John", "Dave", "Joey"],
    "Height(m)": [1.75, 1.65, "-", "na", 1.82],
    "Test Score": [70, np.nan, 8, 62, 73]
}

# Création du DataFrame
df_1 = pd.DataFrame(dictionary_1)
print("Original DataFrame:\n", df_1)

# Détecter les valeurs manquantes standard (NaN)
print("\nIsNull:\n", df_1.isnull())

# Remplacer les valeurs non standard par NaN
df_1 = df_1.replace(["-", "na"], np.nan)
print("\nAfter replacing non-standard missing values:\n", df_1)

# Remplir les NaN par 0
df_1 = df_1.fillna(0)
print("\nAfter fillna(0):\n", df_1)

```

Original DataFrame:

	Name	Height(m)	Test Score
0	Alex	1.75	70.0
1	Mike	1.65	NaN
2	John	-	8.0
3	Dave	na	62.0
4	Joey	1.82	73.0

IsNull:

	Name	Height(m)	Test Score
0	False	False	False
1	False	False	True
2	False	False	False
3	False	False	False
4	False	False	False

After replacing non-standard missing values:

	Name	Height(m)	Test Score
0	Alex	1.75	70.0
1	Mike	1.65	NaN
2	John	NaN	8.0
3	Dave	NaN	62.0
4	Joey	1.82	73.0

After fillna(0):

	Name	Height(m)	Test Score
--	------	-----------	------------

0	Alex	1.75	70.0
1	Mike	1.65	0.0
2	John	0.00	8.0
3	Dave	0.00	62.0
4	Joey	1.82	73.0

C:\Users\Thinkpad\AppData\Local\Temp\ipykernel_28472\2380661644.py:19:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer\_objects(copy=False)`. To opt-in to the future behavior, set `pd.set\_option('future.no\_silent\_downcasting', True)`
df_1 = df_1.replace(["-", "na"], np.nan)

```
[142]: import numpy as np
from sklearn.impute import SimpleImputer
X=[[np.nan,2],[6,np.nan],[7,6]]
# mean, median, most_frequent, constant(fill_value=)
imp=SimpleImputer(missing_values=np.nan,strategy='mean')
data=imp.fit_transform(X)
print(data)
```

```
[[6.5 2. ]
 [6.  4. ]
 [7.  6. ]]
```

```
[144]: from sklearn.datasets import make_classification
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # pour visualisation 3D

# Génération des données
X, y = make_classification(
    n_samples=10000,
    n_features=4,
    flip_y=0.1,
    class_sep=0.1,
    random_state=42
)

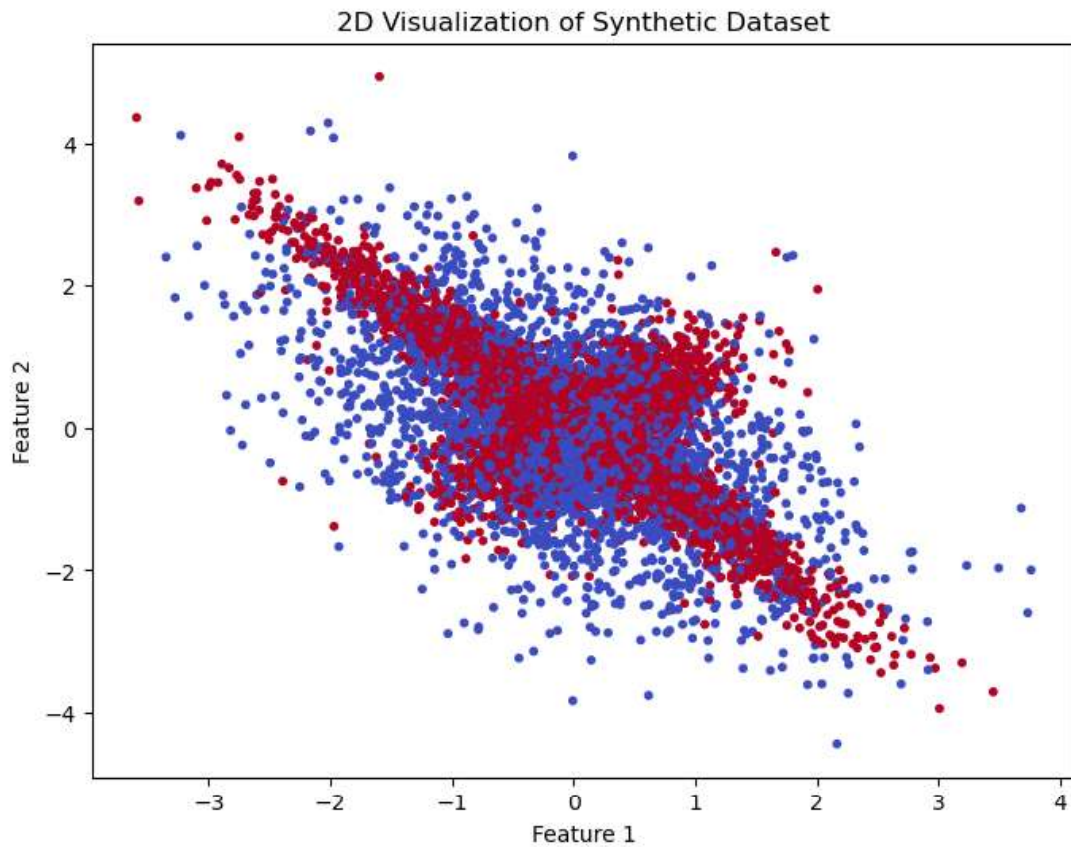
print("X shape =", X.shape)
print("len y =", len(y))
print("y unique classes:", set(y))

# Visualisation 2D (en utilisant les deux premières features)
plt.figure(figsize=(8,6))
plt.scatter(X[:,0], X[:,1], c=y, cmap='coolwarm', s=10)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('2D Visualization of Synthetic Dataset')
```

```
plt.show()

# Visualisation 3D (en utilisant les trois premières features)
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:,0], X[:,1], X[:,2], c=y, cmap='coolwarm', s=10)
ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.set_zlabel('Feature 3')
plt.title('3D Visualization of Synthetic Dataset')
plt.show()
```

X shape = (10000, 4)
 len y = 10000
 y unique classes: {0, 1}



3D Visualization of Synthetic Dataset

