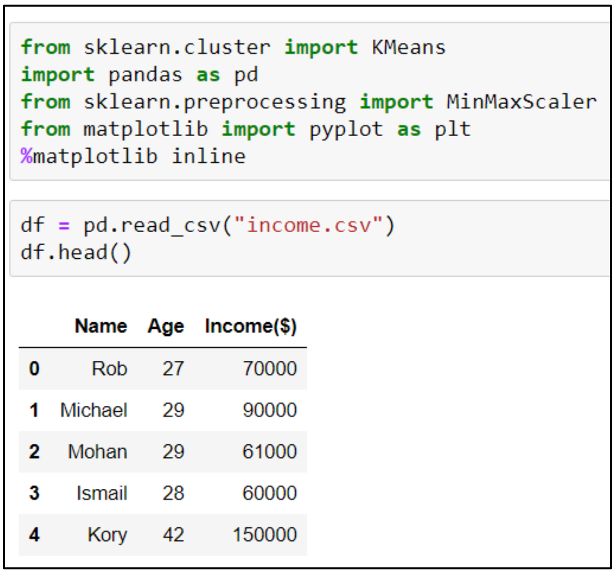
Machine Learning en utilisant Sklearn et Pandas

1. Introduction

Dans ce tutorial, nous allons implémenter des scripts Python en utilisant deux packages Sklearn et Pandas pour effectuer l'apprentissage automatique.

En particulier, nous allons voir des exemples de 3 tâches de base de l'apprentissage automatique : Clustering, Classification et Regression.

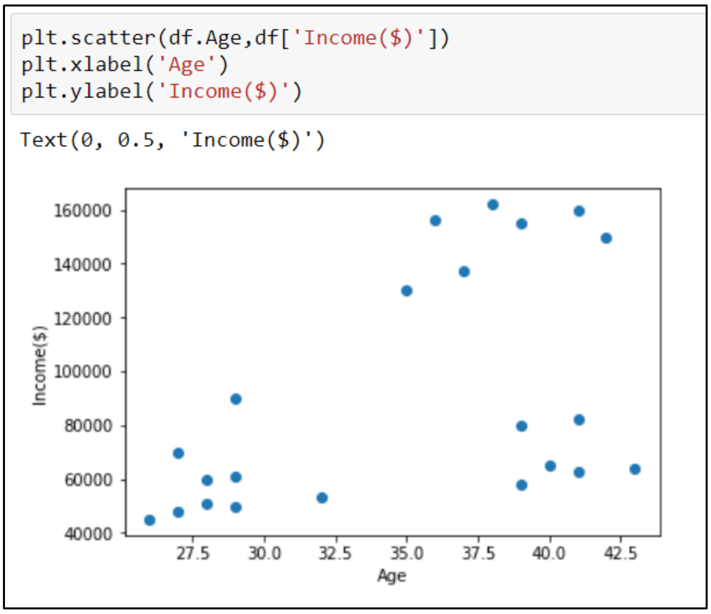
2. Clustering



```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
```

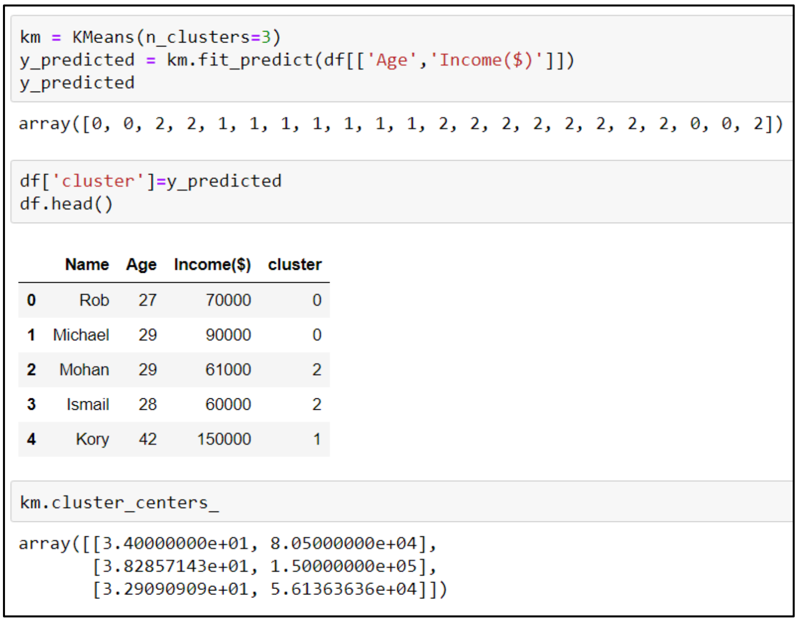
```
df = pd.read_csv("income.csv")
df.head()
```

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000



```
plt.scatter(df.Age,df['Income($)'])
plt.xlabel('Age')
plt.ylabel('Income($)')
```

```
Text(0, 0.5, 'Income($)')
```



```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted
```

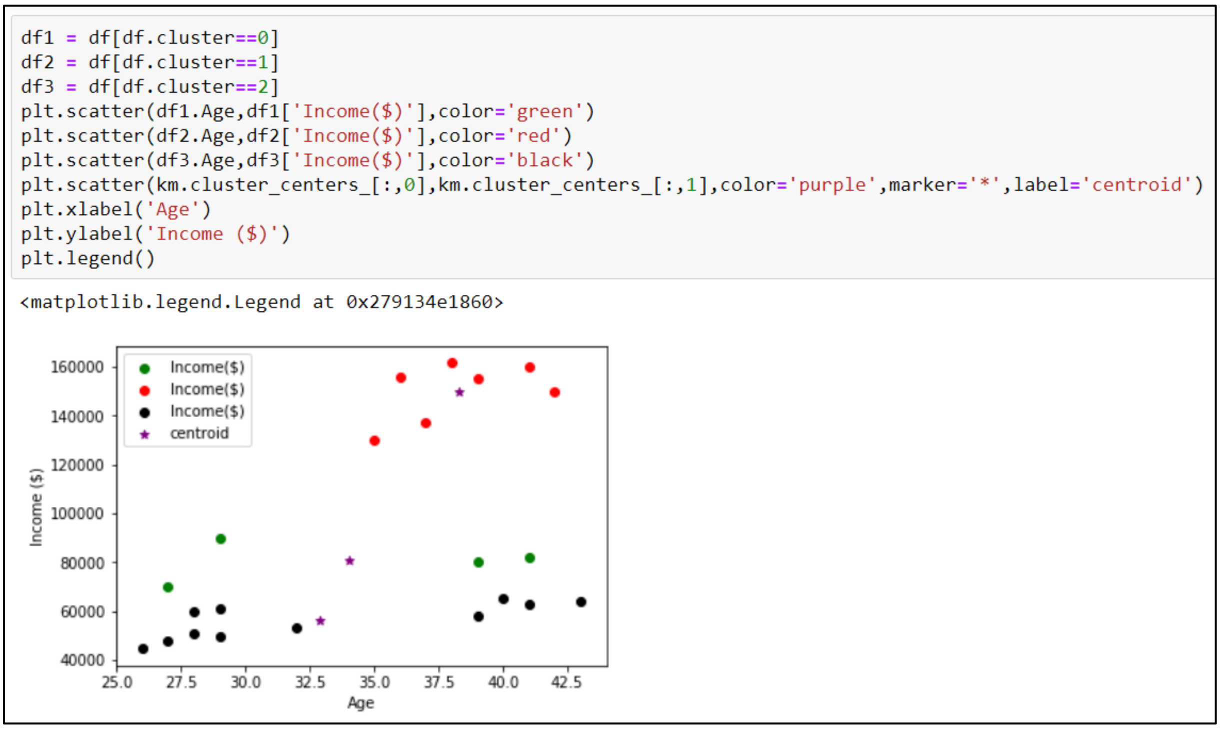
```
array([0, 0, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2])
```

```
df['cluster']=y_predicted
df.head()
```

	Name	Age	Income($)	cluster
0	Rob	27	70000	0
1	Michael	29	90000	0
2	Mohan	29	61000	2
3	Ismail	28	60000	2
4	Kory	42	150000	1

```
km.cluster_centers_
```

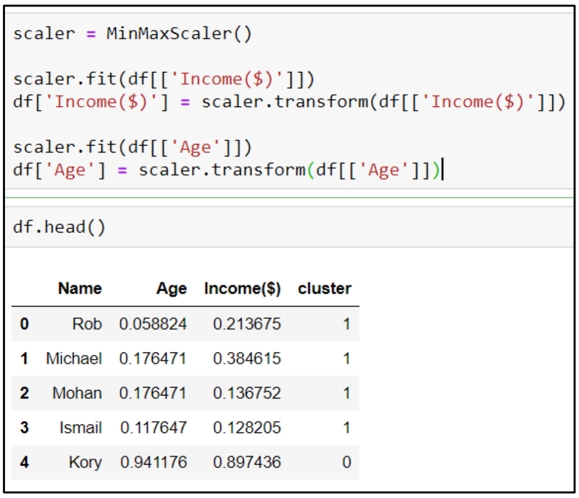
```
array([[3.40000000e+01, 8.05000000e+04],
       [3.82857143e+01, 1.50000000e+05],
       [3.29090909e+01, 5.61363636e+04]])
```



```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
plt.xlabel('Age')
plt.ylabel('Income ($)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x279134e1860>
```

Clustering avec pré-traitement :



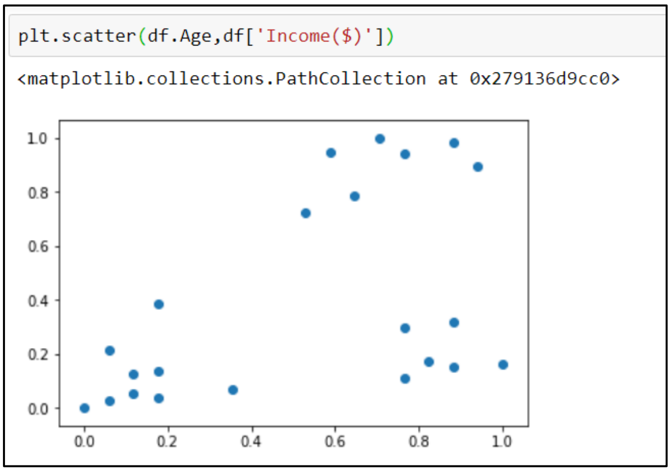
```
scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])
```

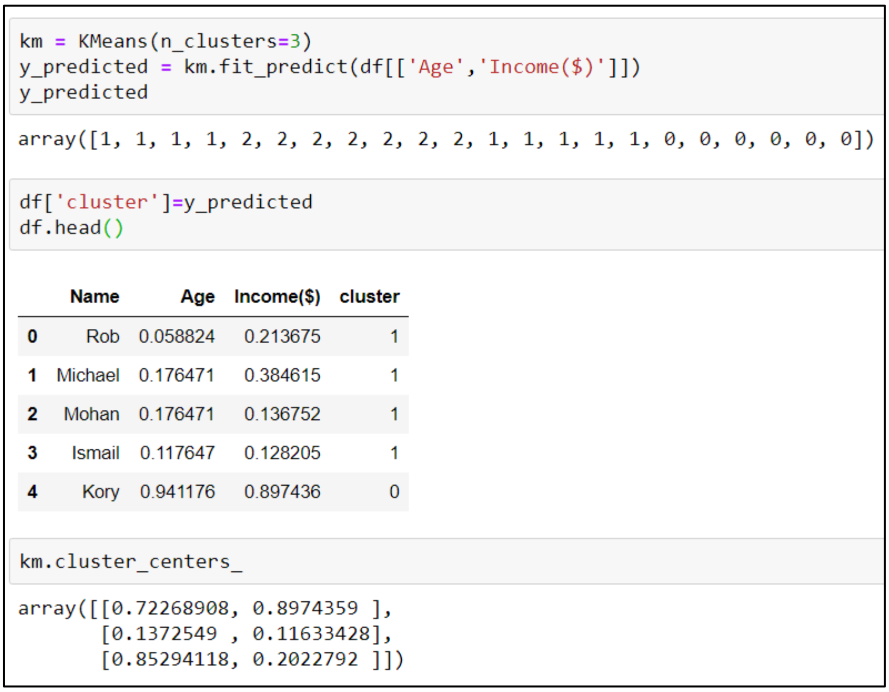
```
df.head()
```

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	1
1	Michael	0.176471	0.384615	1
2	Mohan	0.176471	0.136752	1
3	Ismail	0.117647	0.128205	1
4	Kory	0.941176	0.897436	0



```
plt.scatter(df.Age,df['Income($)'])
```

```
<matplotlib.collections.PathCollection at 0x279136d9cc0>
```



```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted
```

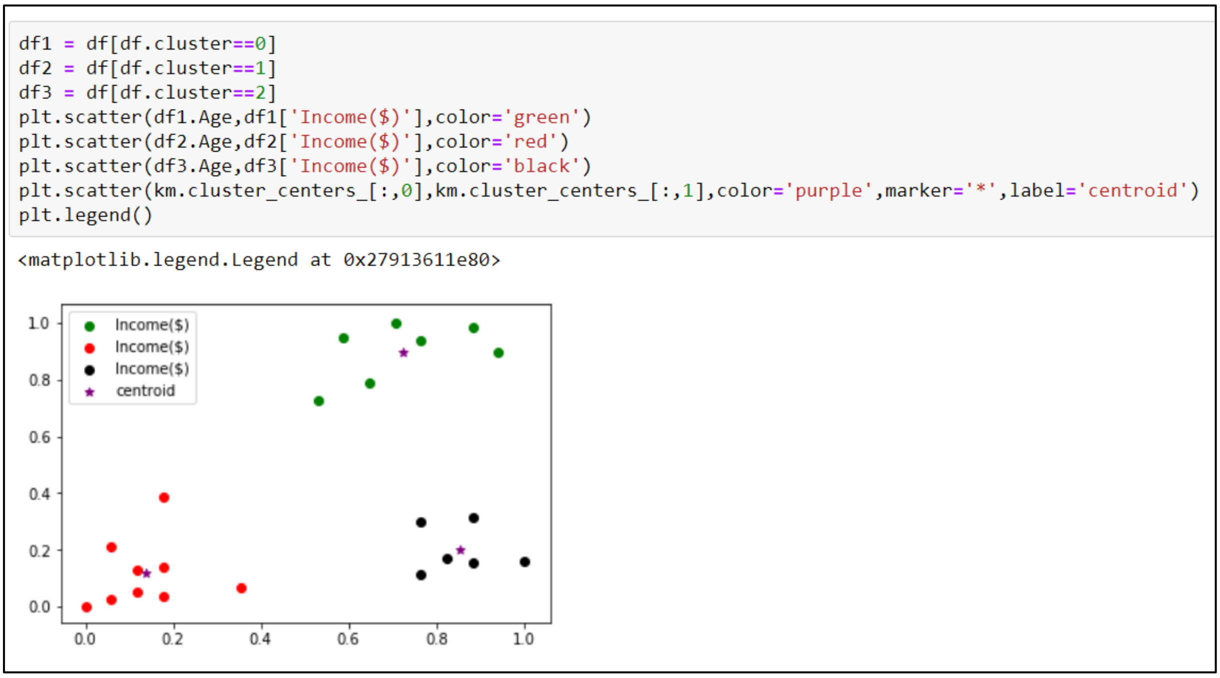
```
array([1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0])
```

```
df['cluster']=y_predicted
df.head()
```

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	1
1	Michael	0.176471	0.384615	1
2	Mohan	0.176471	0.136752	1
3	Ismail	0.117647	0.128205	1
4	Kory	0.941176	0.897436	0

```
km.cluster_centers_
```

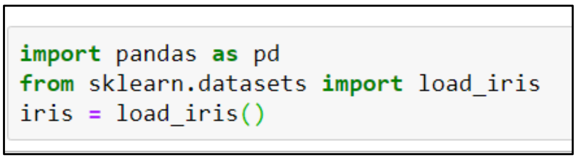
```
array([[0.72268908, 0.8974359 ],
       [0.1372549 , 0.11633428],
       [0.85294118, 0.2022792 ]])
```



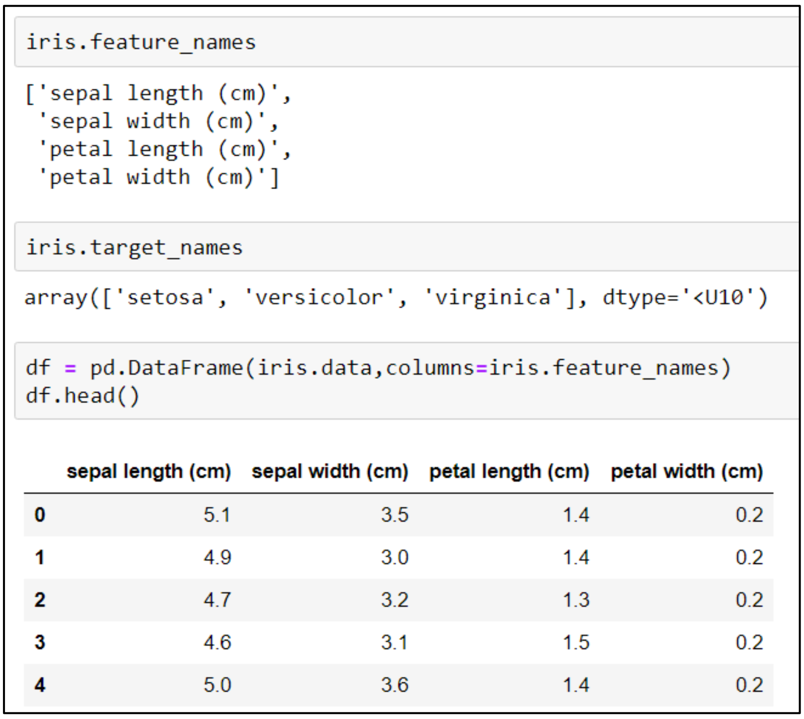
```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x27913611e80>
```

3. Classification



```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```



```
iris.feature_names
```

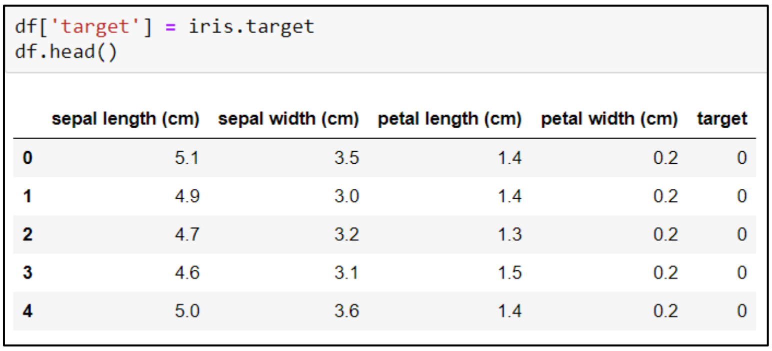
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

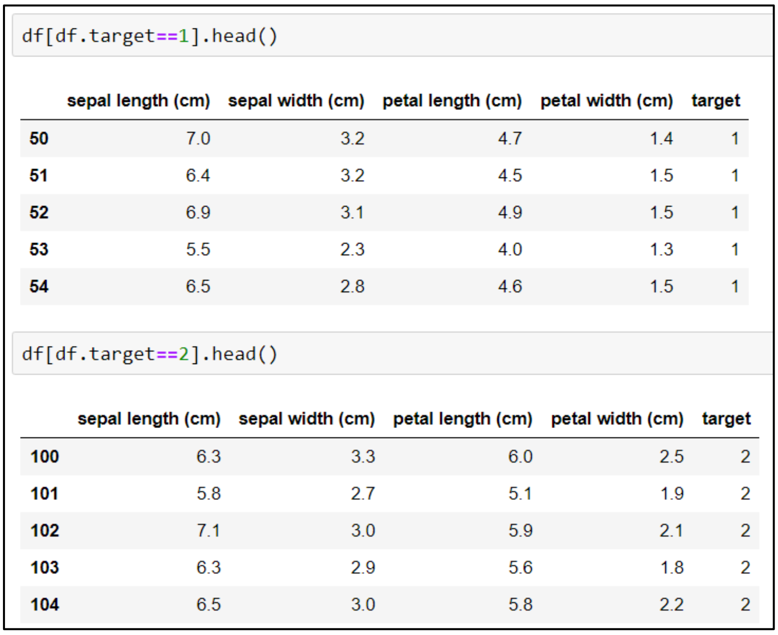
```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2



```
df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

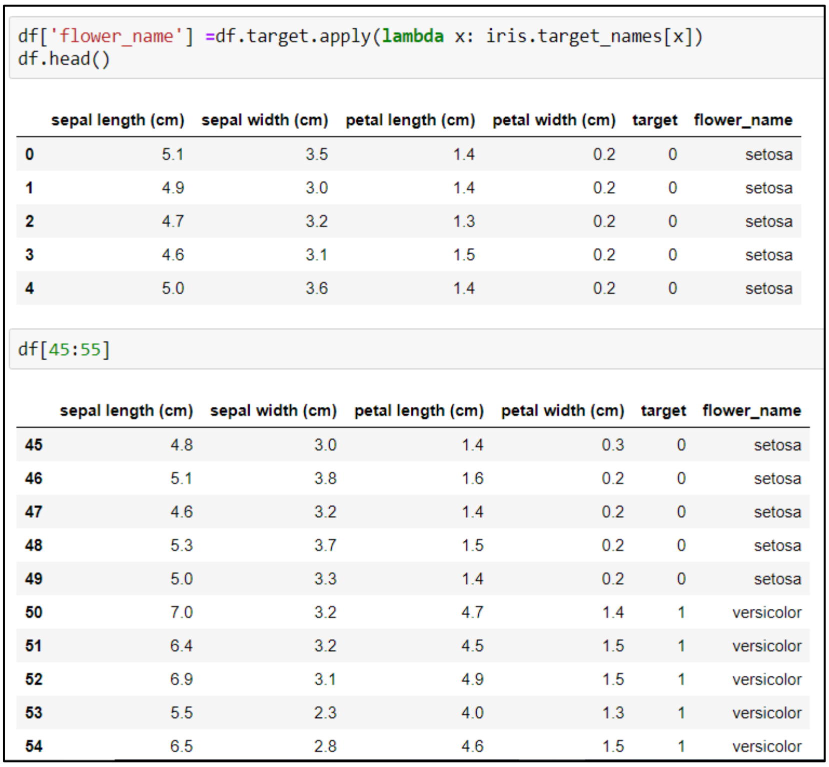


```
df[df.target==1].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

```
df[df.target==2].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

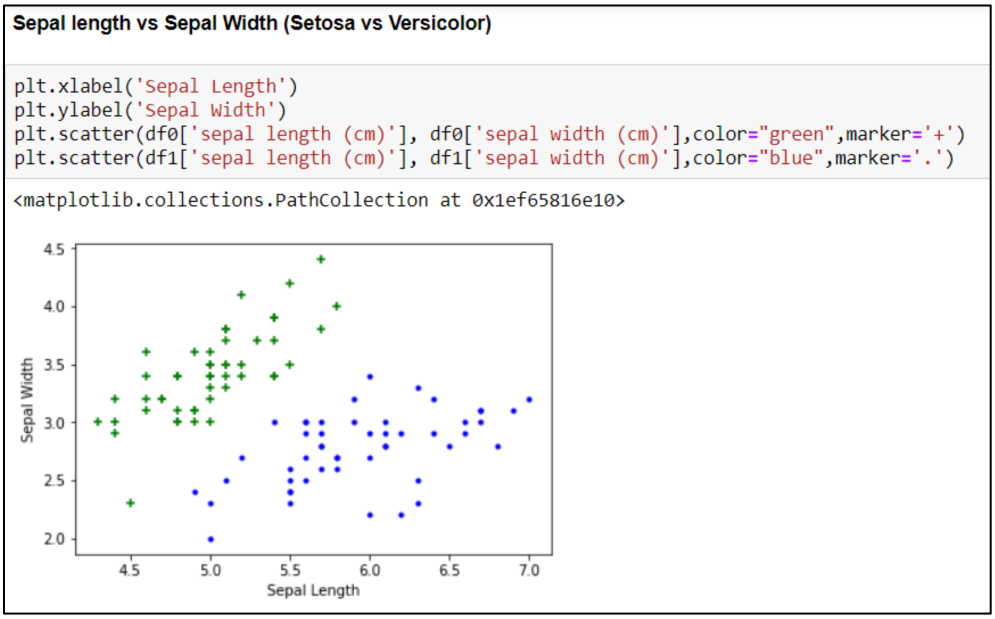


```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
df[45:55]
```

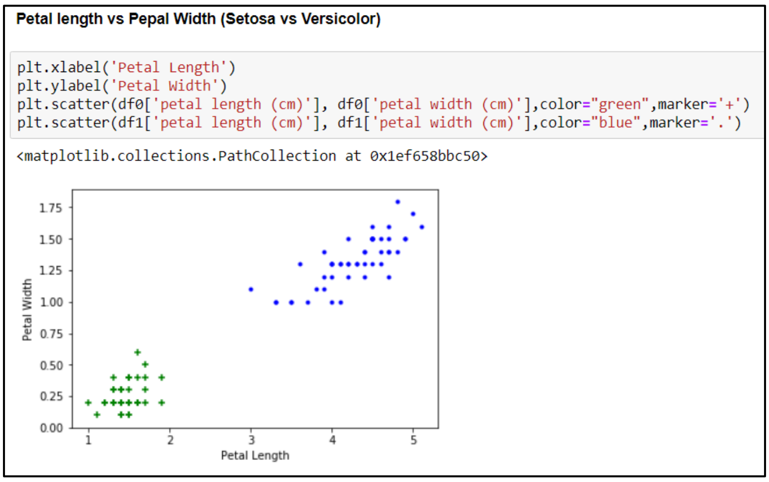
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor



Sepal length vs Sepal Width (Setosa vs Versicolor)

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

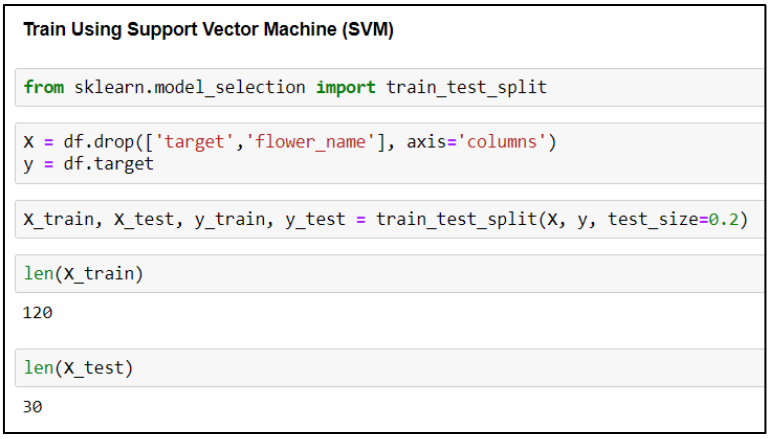
```
<matplotlib.collections.PathCollection at 0x1ef65816e10>
```



Petal length vs Pepal Width (Setosa vs Versicolor)

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x1ef658bbc50>
```



Train Using Support Vector Machine (SVM)

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

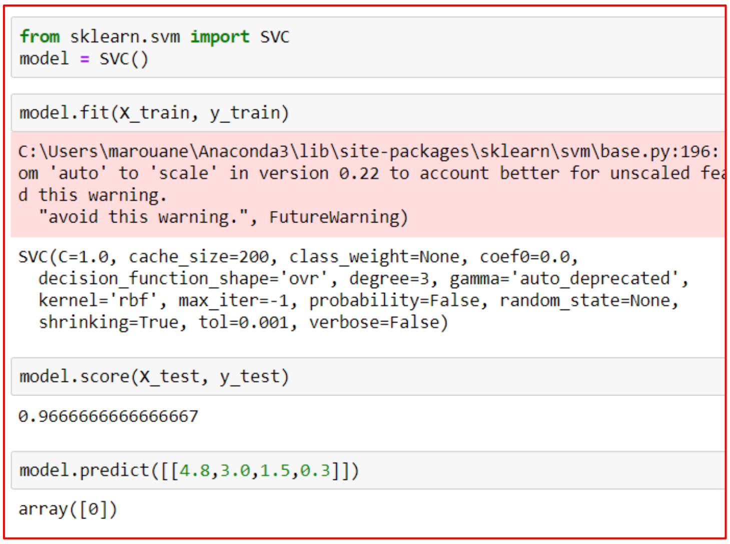
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
len(X_train)
```

```
120
```

```
len(X_test)
```

```
30
```



```
from sklearn.svm import SVC
model = SVC()
```

```
model.fit(X_train, y_train)
```

```
C:\Users\marouane\Anaconda3\lib\site-packages\sklearn\svm\base.py:196:
om 'auto' to 'scale' in version 0.22 to account better for unscaled fea
d this warning.
  "avoid this warning.", FutureWarning)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

```
model.score(X_test, y_test)
```

```
0.9666666666666667
```

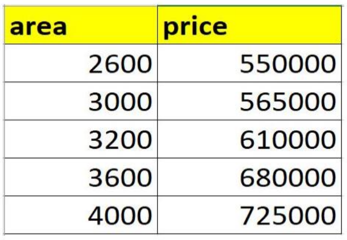
```
model.predict([[4.8,3.0,1.5,0.3]])
```

```
array([0])
```

4. Régression linéaire

4.1. Regression à une seule variable

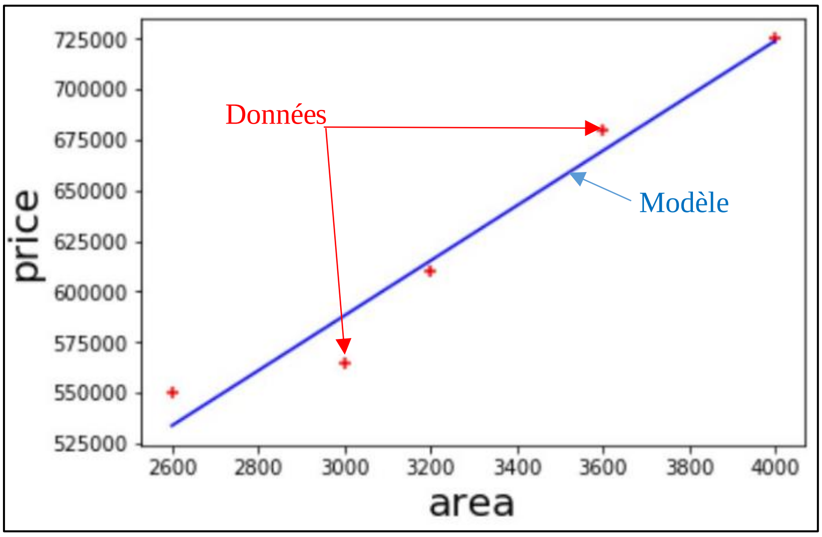
Données



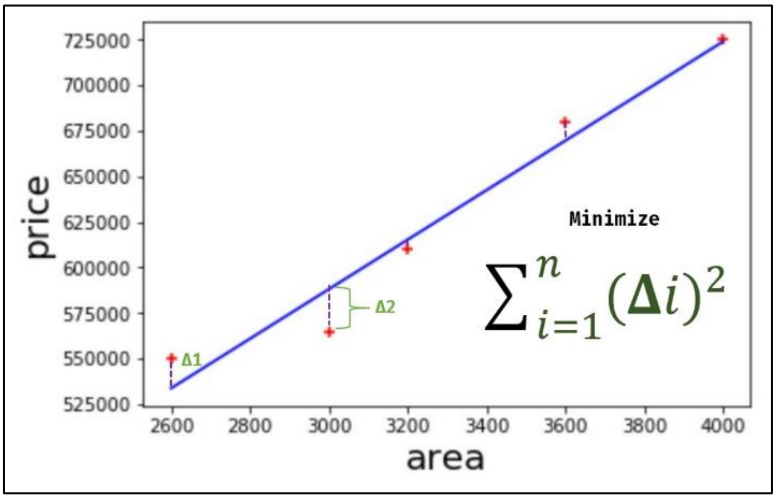
area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Modèle

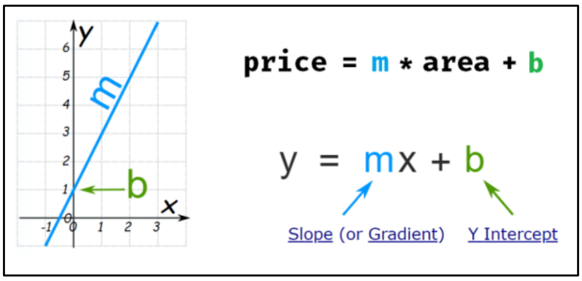
Énoncé du problème: Etant donné les données ci-dessus, nous allons construire un modèle d'apprentissage automatique capable de prévoir le prix des maisons en fonction de la superficie en m2.



Solution :



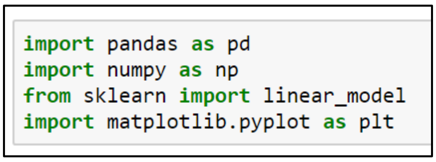
Modèle mathématique:



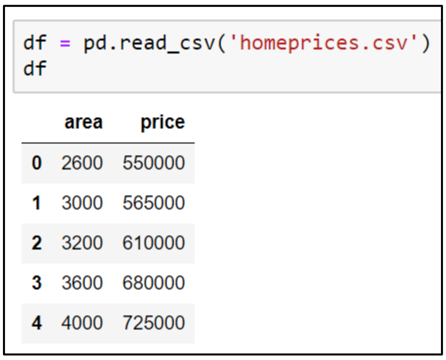
price = m * area + b

$y = mx + b$

Implémentation d’apprentissage:

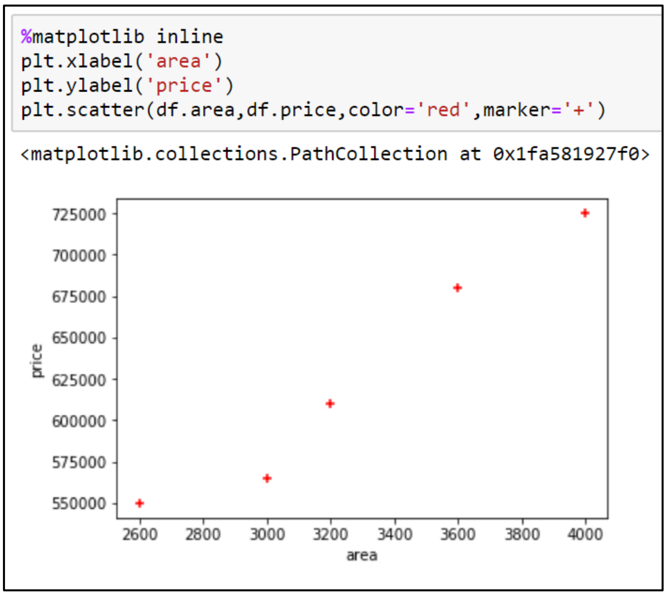


```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
```



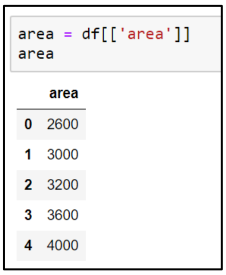
```
df = pd.read_csv('homeprices.csv')
df
```

	area	price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000



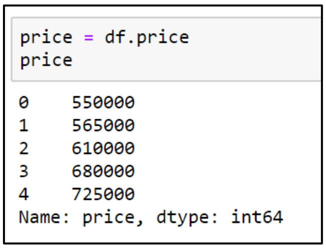
```
%matplotlib inline
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='red',marker='+')
```

```
<matplotlib.collections.PathCollection at 0x1fa581927f0>
```



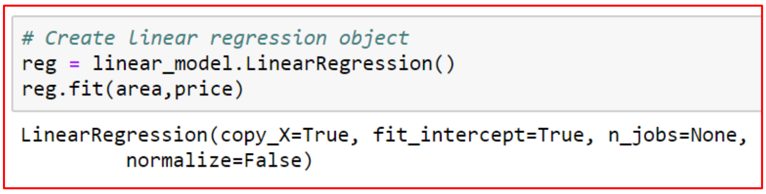
```
area = df[['area']]
area
```

	area
0	2600
1	3000
2	3200
3	3600
4	4000



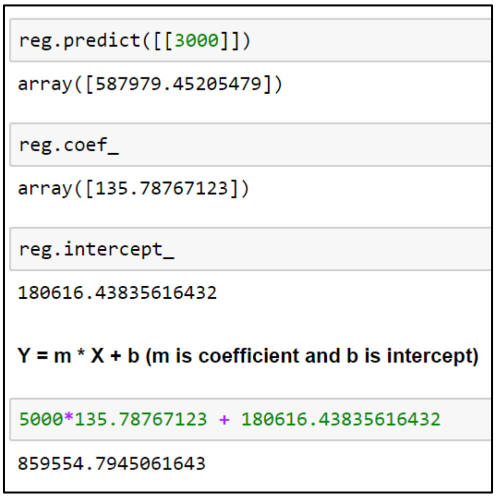
```
price = df.price
price
```

```
0    550000
1    565000
2    610000
3    680000
4    725000
Name: price, dtype: int64
```



```
# Create linear regression object
reg = linear_model.LinearRegression()
reg.fit(area,price)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```



```
reg.predict([[3000]])
```

```
array([587979.45205479])
```

```
reg.coef_
```

```
array([135.78767123])
```

```
reg.intercept_
```

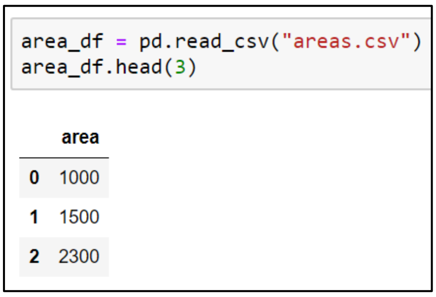
```
180616.43835616432
```

Y = m * X + b (m is coefficient and b is intercept)

```
5000*135.78767123 + 180616.43835616432
```

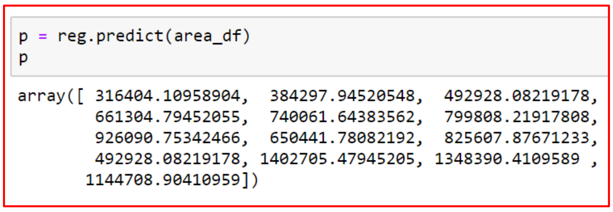
```
859554.7945061643
```

Implémentation du test :



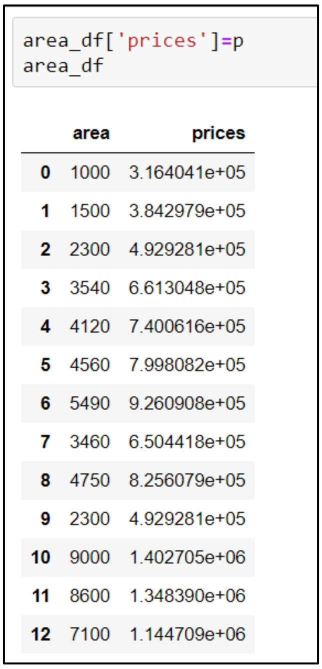
```
area_df = pd.read_csv("areas.csv")
area_df.head(3)
```

	area
0	1000
1	1500
2	2300



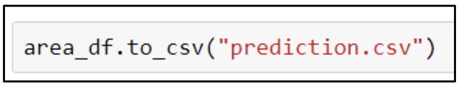
```
p = reg.predict(area_df)
p
```

```
array([ 316404.10958904,  384297.94520548,  492928.08219178,
        661304.79452055,  740061.64383562,  799808.21917808,
        926090.75342466,  650441.78082192,  825607.87671233,
        492928.08219178, 1402705.47945205, 1348390.4109589 ,
       1144708.90410959])
```

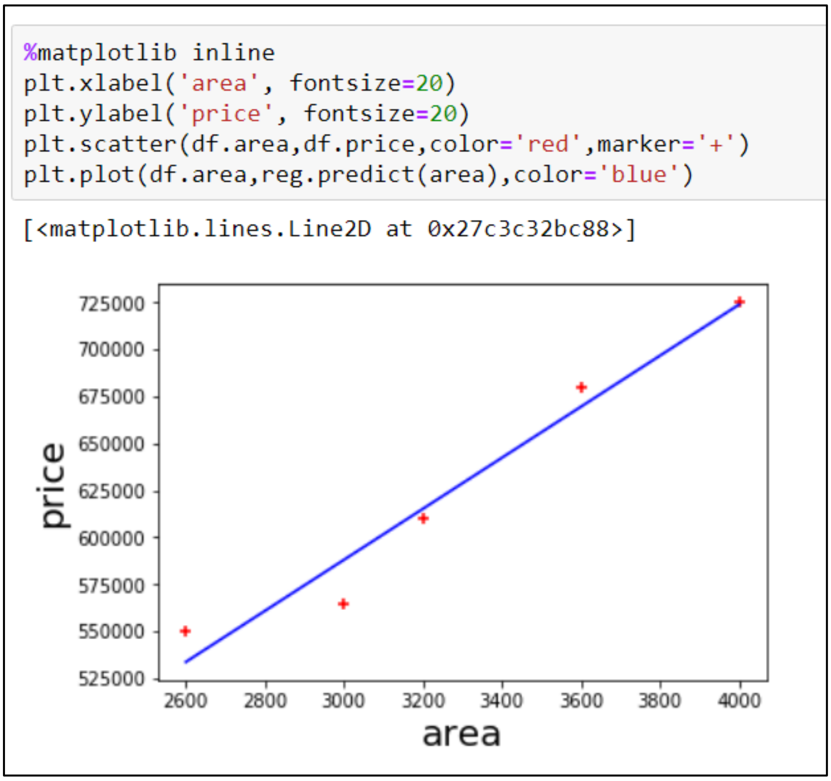


```
area_df['prices']=p
area_df
```

	area	prices
0	1000	3.164041e+05
1	1500	3.842979e+05
2	2300	4.929281e+05
3	3540	6.613048e+05
4	4120	7.400616e+05
5	4560	7.998082e+05
6	5490	9.260908e+05
7	3460	6.504418e+05
8	4750	8.256079e+05
9	2300	4.929281e+05
10	9000	1.402705e+06
11	8600	1.348390e+06
12	7100	1.144709e+06



```
area_df.to_csv("prediction.csv")
```



```
%matplotlib inline
plt.xlabel('area', fontsize=20)
plt.ylabel('price', fontsize=20)
plt.scatter(df.area,df.price,color='red',marker='+')
plt.plot(df.area,reg.predict(area),color='blue')
```

```
[<matplotlib.lines.Line2D at 0x27c3c32bc88>]
```

Exercice :

Enoncé :

Prédictez le revenu par habitant du canada en 2020.

Les données sont enregistrées dans le fichier canada_per_capita_income.csv.

À partir de là :

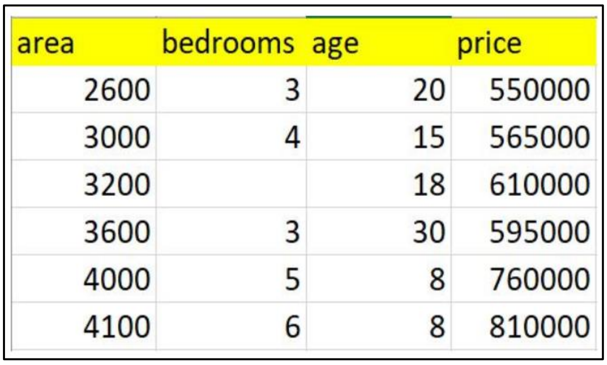
1. On construit un modèle de régression
2. On prédit le revenu par habitant des citoyens canadiens en 2020

Réponse :

41288.69409442

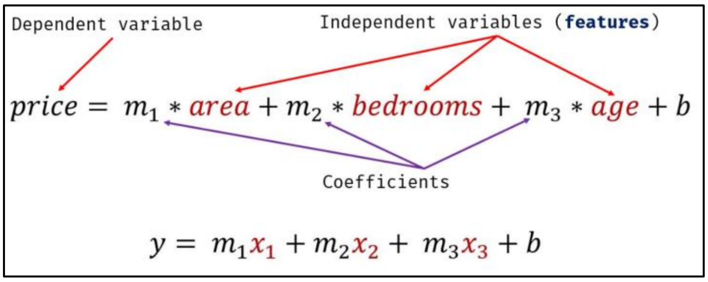
4.2. Regression à variable multiple

Données



area	bedrooms	age	price
2600	3	20	550000
3000	4	15	565000
3200		18	610000
3600	3	30	595000
4000	5	8	760000
4100	6	8	810000

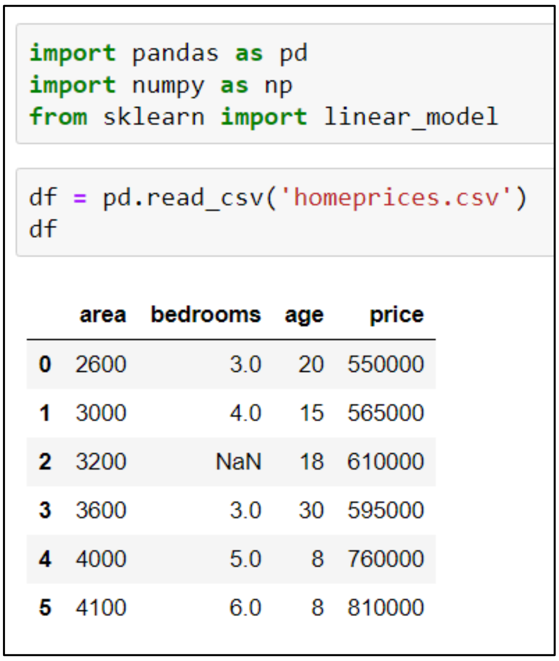
Modèle mathématique



$$price = m_1 * area + m_2 * bedrooms + m_3 * age + b$$

$$y = m_1 x_1 + m_2 x_2 + m_3 x_3 + b$$

Implémentation :



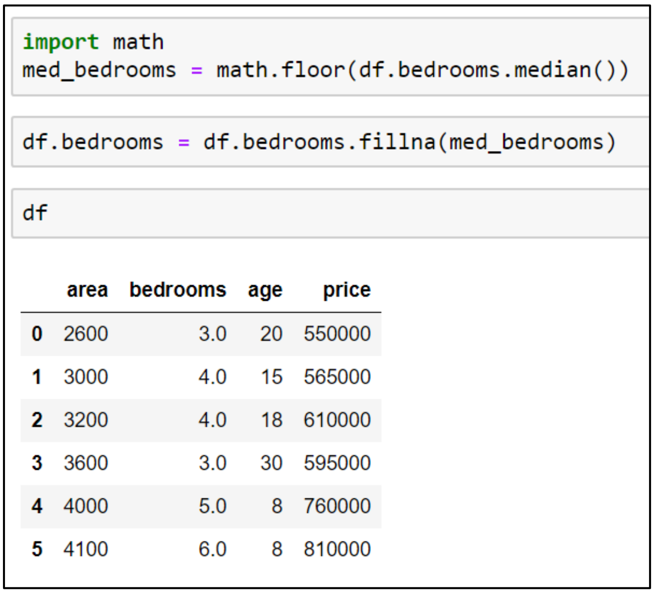
```
import pandas as pd
import numpy as np
from sklearn import linear_model
```

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

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	NaN	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000
5	4100	6.0	8	810000

Pré-traitement des données :

On va remplacer les valeurs NaN (indéfinies) avec la valeur de médiane de la colonne



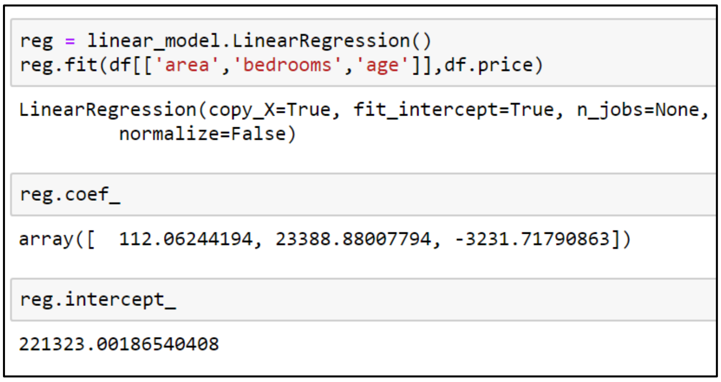
```python
import math
med_bedrooms = math.floor(df.bedrooms.median())
```

```python
df.bedrooms = df.bedrooms.fillna(med_bedrooms)
```

```python
df
```

	area	bedrooms	age	price
0	2600	3.0	20	550000
1	3000	4.0	15	565000
2	3200	4.0	18	610000
3	3600	3.0	30	595000
4	4000	5.0	8	760000
5	4100	6.0	8	810000

Apprentissage



```python
reg = linear_model.LinearRegression()
reg.fit(df[['area','bedrooms','age']],df.price)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

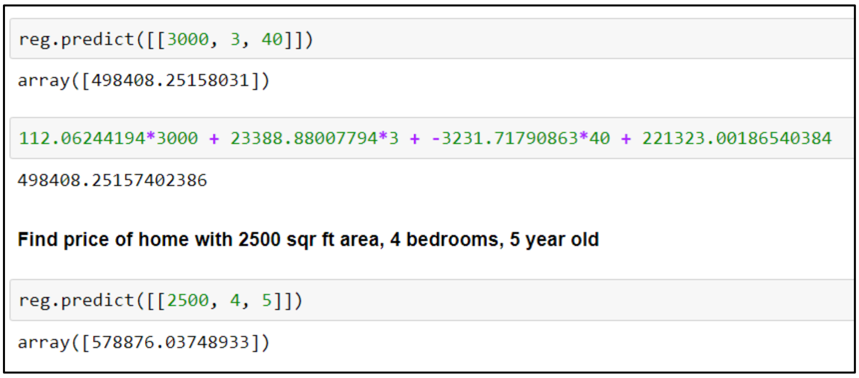
```python
reg.coef_
```

```
array([  112.06244194, 23388.88007794, -3231.71790863])
```

```python
reg.intercept_
```

```
221323.00186540408
```

Test : trouver le prix d'une maison de superficie 3000 m2 , ayant 3 bedrooms et d'âge 40 ans ?



```python
reg.predict([[3000, 3, 40]])
```

```
array([498408.25158031])
```

```python
112.06244194*3000 + 23388.88007794*3 + -3231.71790863*40 + 221323.00186540384
```

```
498408.25157402386
```

Find price of home with 2500 sqr ft area, 4 bedrooms, 5 year old

```python
reg.predict([[2500, 4, 5]])
```

```
array([578876.03748933])
```

Exercice :

<u>Enoncé :</u>

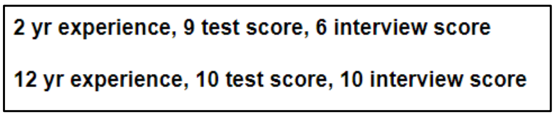
Les données sont enregistrées dans le fichier hiring.csv.

Ce fichier contient des statistiques d’embauche pour une entreprise telles que l’expérience du candidat, sa note au test écrit et sa note pour l’entretien personnel.

Sur la base de ces 3 facteurs, les ressources humaines décideront du salaire.

Compte tenu de ces données, on doit créer un modèle d’apprentissage automatique pour le service des ressources humaines, qui peut l’aider à déterminer les salaires des futurs candidats.

Travail demandé : Prédire les salaires pour les candidats suivants :



2 yr experience, 9 test score, 6 interview score

12 yr experience, 10 test score, 10 interview score

Réponse :

53713.86

93747.79