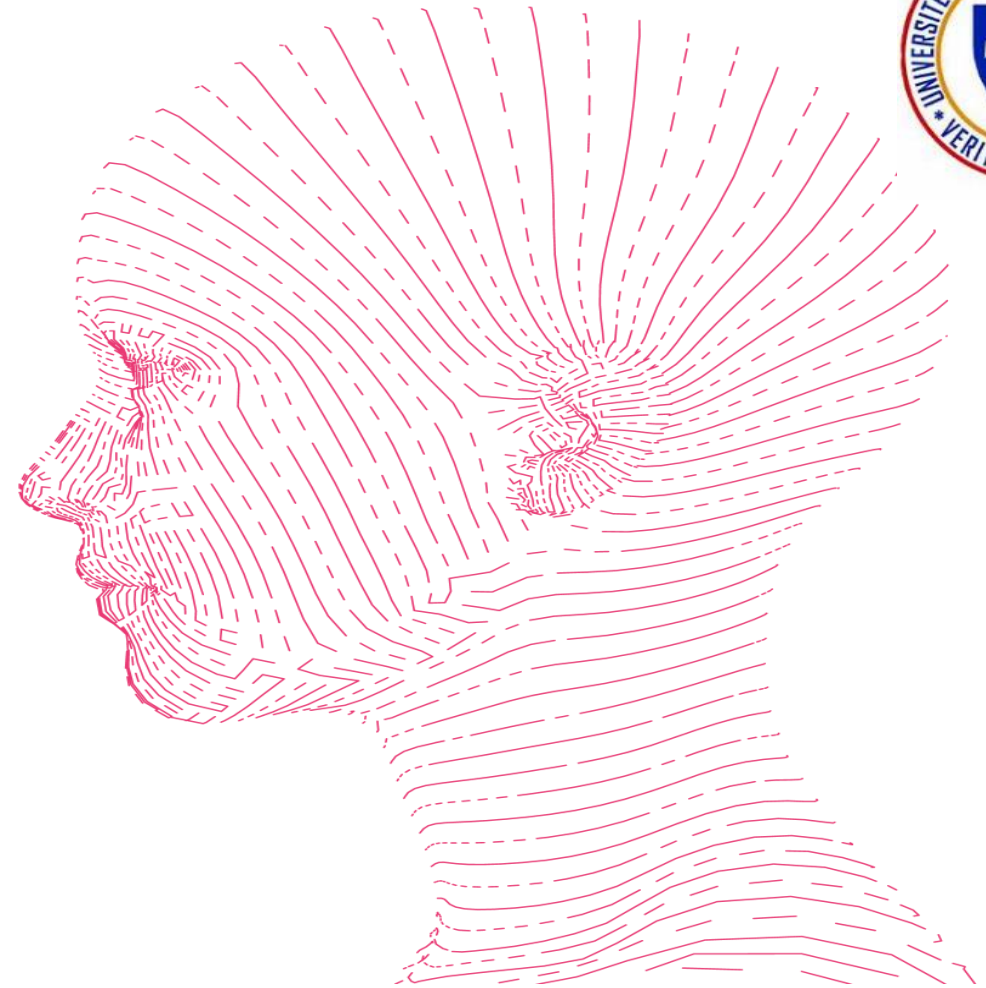


# Machine Learning



- Dispensé par **MWAMBA KASONGO Dahouda**
- Docteur en génie logiciel et systèmes d'information
- Machine and Deep Learning Engineer



Assisté par Ass. **Daniel MBAYA**

- E-mail : [dahouda37@gmail.com](mailto:dahouda37@gmail.com)
- Tel.: **+243 99 66 55 265**

Heure : 10H00 – 12H00

## PLAN DU COURS



### CHAPITRE 3 Apprentissage supervisé : Régression et Classification

#### 3.1 Régression linéaire :

- Introduction à la régression linéaire multiple
- Prédire des valeurs continues (par exemple, prédire les prix des maisons)
- Const Function (Fonction de coût) et Mean Squared Error (Erreur quadratique moyenne)

#### 3.2 Classification :

- Régression logistique pour la classification binaire
- Métriques de classification dans le Machine Learning  
Accuracy (Exactitude), Precision (précision), Recall (rappel), F1-Score
- Prédiction de l'approbation des prêts à l'aide de Machine Learning
- Comparaison de performance des algorithmes

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple: 3.1.1 Introduction à la régression linéaire multiple

- ❑ Dans ce chapitre, nous allons découvrir la régression linéaire multiple à l'aide de scikit-learn dans le langage de programmation Python.
- ❑ La régression est une méthode statistique permettant de déterminer la relation entre des caractéristiques (Features ou variables indépendantes) et une variable de résultat (Target ou variable dépendante) ou un résultat.
- ❑ La régression linéaire multiple, souvent appelée régression multiple, est une méthode statistique qui prédit le résultat d'une variable de réponse en combinant de nombreuses variables explicatives.
- ❑ La régression multiple est une variante de la régression linéaire dans laquelle une seule variable explicative est utilisée  
ici, **y** est la variable dépendante.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \dots$$

$x_1, x_2, x_3, \dots$  sont des variables indépendantes.

$b_0$  = interception de la droite.

$b_1, b_2, \dots$  sont des coefficients.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple: 3.1.1 Introduction à la régression linéaire multiple

□ La régression multiple est une variante de la régression linéaire dans laquelle une seule variable explicative est utilisée

Formulation Mathématique:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \dots$$

The diagram shows the equation  $y = \alpha + \beta_1(x_1) + \beta_2(x_2) + \dots + \beta_n(x_n)$  with the following annotations:

- Predicted value**: points to  $y$ .
- Bias**: points to  $\alpha$ .
- Weight 1**: points to  $\beta_1$ .
- Feature 1**: points to  $x_1$ .
- Weight 2**: points to  $\beta_2$ .
- Feature 2**: points to  $x_2$ .
- Weight n**: points to  $\beta_n$ .
- Feature n**: points to  $x_n$ .

ici, **y** est la variable dépendante.

$x_1, x_2, x_3, \dots$  sont des variables indépendantes.

$b_0$  = interception de la droite.

$b_1, b_2, \dots$  sont des coefficients.

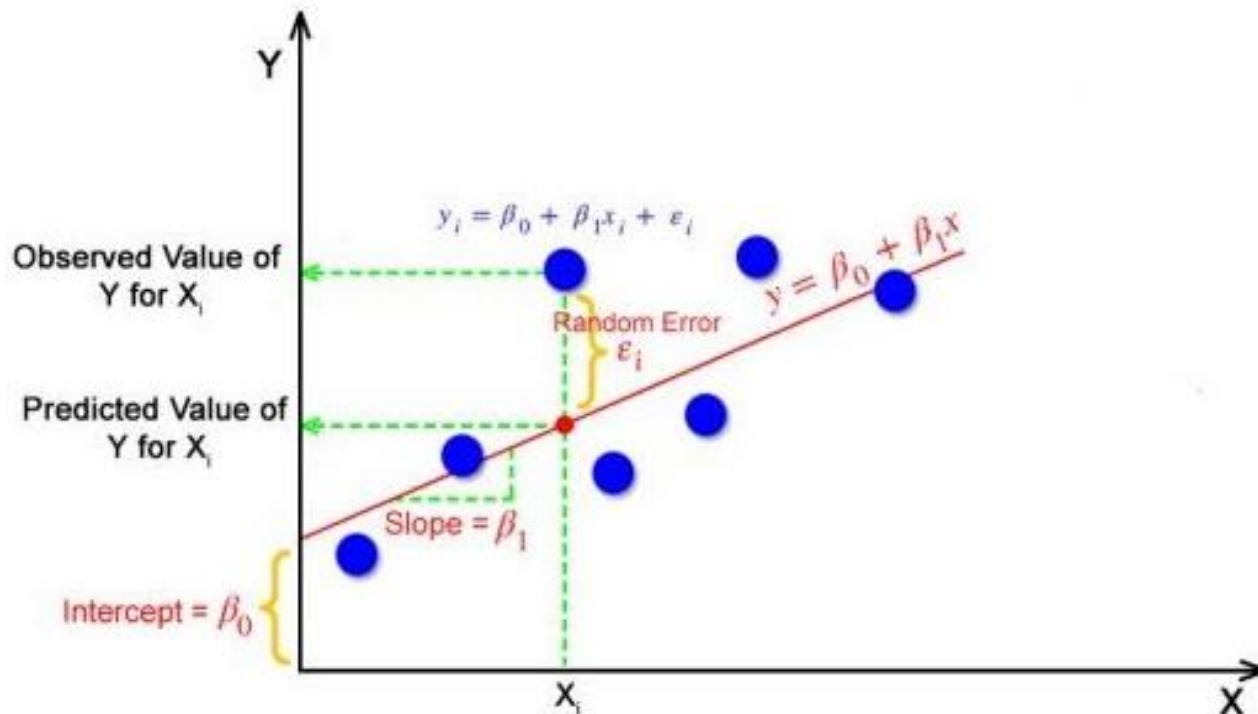
## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.1 Introduction à la régression linéaire multiple

- ❑ La régression linéaire multiple, souvent appelée régression multiple, est une méthode statistique qui prédit le résultat d'une variable de réponse en combinant de nombreuses variables explicatives.



#### Formulation Mathématique:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \dots$$

ici,  $y$  est la variable dépendante.

$x_1, x_2, x_3, \dots$  sont des variables indépendantes.

$b_0$  = interception de la droite.

$b_1, b_2, \dots$  sont des coefficients.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

Dans le contexte de l'apprentissage automatique, une dataset est un ensemble de données qui sert de base à la création et à l'entraînement de modèles.

Une dataset se compose généralement de lignes et de colonnes, où :

- ✓ **Lignes** (échantillons/observations) : chaque ligne représente, un attribut ou une variable.
- ✓ La dernière colonne est souvent l'étiquette cible dans les tâches d'apprentissage supervisé.
- ✓ **Colonnes** (caractéristiques/attributs) : chaque colonne représente une caractéristique d'apprentissage supervisé.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

Voici les principaux composants d'un ensemble de données typique :

- **1. Caractéristiques [Features]** (variables indépendantes)

Il s'agit des variables d'entrée à partir desquelles le modèle d'apprentissage automatique apprendra.

Chaque caractéristique peut représenter une propriété ou une caractéristique mesurable, telle que l'âge, le revenu, la température.

- **2. Cible/étiquette [Target]** (variable dépendante)

Il s'agit de la variable de sortie que le modèle est censé prédire (dans l'apprentissage supervisé).

Dans la régression, il s'agit généralement d'une valeur continue (par exemple, le prix de maison).



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

Labo: Étapes de mise en œuvre de la régression linéaire multiple:

#### Étape 1 : Importer les packages nécessaires

```
1 # Importation de modules et de packages
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 from sklearn.model_selection import train_test_split
7 from sklearn.linear_model import LinearRegression
8 from sklearn.metrics import mean_squared_error, mean_absolute_error
9 from sklearn import preprocessing
10
```



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### Travail Dirige

#### 1. Utiliser un ensemble de données Salary\_data:

- ✓ Appliquer une régression linéaire,
- ✓ Evaluer le modèle,
- ✓ Afficher la ligne de la regression

#### 2. Utiliser un ensemble de données USA\_Housing:

- ✓ Appliquer une régression linéaire,
- ✓ Régression multiple,
- ✓ Evaluer le modèle,
- ✓ Afficher la ligne de la regression

- Tout le code sera stocké dans un dossier portant votre nom complet.
- Mettez le code dans github
- Envoyez-moi le lien de votre repository git hub pour évaluation

#### 3. Utiliser un ensemble de données Real-estate:

- ✓ Appliquer une régression multiple
- ✓ Evaluer le modèle,

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 2 : Importez le fichier CSV

```
1 # Importation d'un ensemble de données
2 df = pd.read_csv("../Data/Real-estate.csv")
3 df.head()
```

[93]:

	No	X1 date de transaction	X2 age de la maison	X3 distance jusqu'a la gare la plus proche	X4 nombre de supermarket de proximite	X5 latitude	X6 longitude	Y prix de la maison par unite de surface
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 2 : Importez le fichier CSV

```
1 df.drop('No', inplace = True, axis=1)
2 df.head()
```

`df.drop('No', inplace=True, axis=1)` :

- **'No'** : nom de la colonne à supprimer (généralement une colonne d'index ou un identifiant inutile).
- **inplace=True** : modifie le DataFrame d'origine df en place, ce qui signifie qu'aucun nouveau DataFrame n'est renvoyé ; celui existant est mis à jour.
- **axis=1** : spécifie que l'opération doit supprimer une colonne. (axis=0 supprimerait des lignes.)

	X1 date de transaction	X2 age de la maison	X3 distance jusqu'a la gare la plus proche	X4 nombre de supermarket de proximite	X5 latitude	X6 longitude	Y prix de la maison par unite de surface
0	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

```
: 1 df.shape
```

```
1 df.columns
```

```
: (414, 7)
```

```
Index(['X1 date de transaction', 'X2 age de la maison',  
      'X3 distance jusqu'a la gare la plus proche',  
      'X4 nombre de supermarket de proximite', 'X5 latitude', 'X6 longitude',  
      'Y prix de la maison par unite de surface'],  
      dtype='object')
```

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



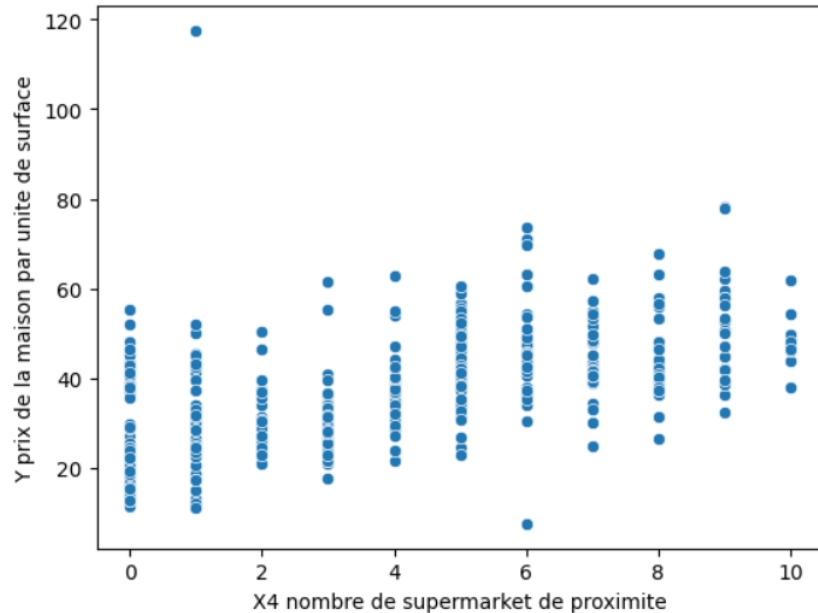
### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

Étape 3 : Créez un nuage de points pour visualiser les données

```
1 # tracer scatterplot
2 sns.scatterplot(x='X4 nombre de supermarket de proximite', y='Y prix de la maison par unite de surface', data=df)
```

[115]: <Axes: xlabel='X4 nombre de supermarket de proximite', ylabel='Y prix de la maison par unite de surface'>



- ✓ **sns.scatterplot** : il s'agit d'une fonction de Seaborn utilisée pour générer des nuages de points.
- ✓ **x='X4 nombre de supermarchés de proximité'** : l'axe des x représente le nombre de supermarchés à proximité.
- ✓ **y='Y prix de la maison par unité de surface'** : l'axe des y représente le prix de la maison par unité de surface.
- ✓ **data=df** : spécifie le DataFrame à partir duquel les données sont extraites

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 4 : Créer des variables de fonctionnalité

- ☐ Pour modéliser les données, nous devons créer des variables de caractéristiques.
- ☐ La variable X contient des variables indépendantes et la variable y contient une variable dépendante.

```
1 # Création de variables de fonctionnalités
2 X = df.drop('Y prix de la maison par unite de surface',axis= 1)
3 y = df['Y prix de la maison par unite de surface']
4 (X)
```

##### ☐ Les variables de caractéristiques X

##### ☐ La variables cible Y

	X1 date de transaction	X2 age de la maison	X3 distance jusqu'a la gare la plus proche	X4 nombre de supermarket de proximite	X5 latitude	X6 longitude
0	2012.917	32.0	84.87882	10	24.98298	121.54024
1	2012.917	19.5	306.59470	9	24.98034	121.53951
2	2013.583	13.3	561.98450	5	24.98746	121.54391
3	2013.500	13.3	561.98450	5	24.98746	121.54391
4	2012.833	5.0	390.56840	5	24.97937	121.54245

1	y
0	37.9
1	42.2
2	47.3
3	54.8
4	43.1

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 5 : Diviser les données en données d'entraînement et de test

```
1 # Création de données d'entraînement et de données de test
2 X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3, random_state=101)
```

```
1 print("X_train:", X_train.shape)
2 print("X_test:", X_test.shape)
3 print("y_train:", y_train.shape)
4 print("y_test:", y_test.shape)
```

X\_train: (289, 6)

X\_test: (125, 6)

y\_train: (289,)

y\_test: (125,)

##### ❑ Paramètres :

- ✓ **test\_size=0.3** : cela signifie que 30 % des données seront allouées à l'ensemble de test (X\_test, y\_test) et 70 % à l'ensemble d'entraînement (X\_train, y\_train).
- ✓ **random\_state=101** : garantit la reproductibilité de la division. Le même état aléatoire produira toujours la même division lorsque vous exécuterez à nouveau le code.

##### ❑ Résultat :

- **X\_train** : 70 % des caractéristiques, qui seront utilisées pour entraîner le modèle.
- **X\_test** : 30 % des caractéristiques, qui seront utilisées pour évaluer les performances du modèle sur des données non vues.
- **y\_train** : 70 % des étiquettes correspondantes, qui seront utilisées pour l'entraînement.
- **y\_test** : 30 % des étiquettes correspondantes, qui seront utilisées pour les tests.

**X** : l'ensemble de caractéristiques (données d'entrée).

**y** : l'ensemble cible (données de sortie).

**train\_test\_split()** : une fonction qui divise les données en sous-ensembles aléatoires d'entraînement et de test.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 6 : Créer un modèle de régression linéaire

- ❑ La classe `LinearRegression()` est utilisée pour créer un modèle de régression multiple, la classe est importée du package `sklearn.linear_model`.

```
7 from sklearn.linear_model import LinearRegression
8 from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
1 # Création d'un modèle de régression
2 multi_reg_linear = LinearRegression()
```

##### Étape 7 : ajuster le modèle avec les données d'entraînement

```
: 1 # Adapter le modèle aux données d'entraînement
  2 multi_reg_linear.fit(X_train,y_train)
```

```
:  LinearRegression ⓘ ⓘ
  LinearRegression()
```

- ✓ **multi\_reg\_linear** : il s'agit de votre objet de modèle de régression linéaire multiple, qui est vraisemblablement une instance de `LinearRegression()` de scikit-learn.
- ✓ **.fit(X\_train, y\_train)** : cette méthode est utilisée pour entraîner le modèle à l'aide des données d'entraînement.
- Le modèle apprendra la relation entre les caractéristiques d'entrée (`X_train`) et les valeurs cibles (`y_train`).



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 8 : Faire des prédictions sur l'ensemble de données de test

- ❑ Dans ce modèle, la méthode **predict()** est utilisée pour faire des prédictions sur les données `X_test`, car les données de test sont des données invisibles et le modèle n'a aucune connaissance des statistiques de l'ensemble de test.

```
1 # Faire la prédiction
2 y_pred = multi_reg_linear.predict(X_test)
```

```
1 y_pred
array([12.63830383, 10.0304461 , 22.98807375, 48.50264837, 32.67140451,
       37.82572669, 36.09178068, 41.05953639, 47.84830793, 40.4574746 ,
       45.0361603 , 32.86533457, 40.48623576, 36.48827849, 44.30595729,
       46.59668235, 38.42798244, 44.26307337, 48.81959723, 45.50409246,
       42.23260833, 54.6526397 , 48.07373298, 37.48194231, 33.57091525,
       48.26293154, 40.23479801, 50.42675437, 47.22333423, 38.99458517,
       48.11033139, 40.47035604, 45.61060308, 43.98441528, 46.54336092,
        8.18725886, 38.08375879, 39.82608171,  8.5339677 , 55.72740213,
       32.17950939, 49.72698264, 24.85604948, 47.64473233, 41.23026871,
       51.17703175, 42.04716292, 37.32689765, 44.24427856, 36.27028988,
       47.55408451, 34.74054504, 43.53329366, 15.95586215, 38.29001222,
       48.93912385, 44.69790471, 44.91934627, 45.33972278, 41.33035787,
       34.39899173, 44.02303164, 41.99940522, 43.9553153 , 53.61420366,
       44.24994361, 24.68926603, 47.06140631, 31.22031534, 40.4930635 ,
       43.12220556, 48.76235412, 15.60855454, 35.80561434, 12.76370021,
```

✓ **multi\_reg\_linear.predict(X\_test)** : cela appelle la méthode **.predict()** du modèle de régression linéaire entraîné (**multi\_reg\_linear**) et génère des valeurs cibles prédites (**y\_pred**) en fonction des caractéristiques de test (**X\_test**).

✓ **X\_test** : l'ensemble de tests des caractéristiques d'entrée qui ont été retenues.

Le modèle utilise ces caractéristiques pour prédire les valeurs cibles correspondantes.

✓ **y\_pred** : les valeurs prédites générées par le modèle, qui peuvent être comparées aux valeurs cibles réelles (**y\_test**) pour évaluer les performances du modèle.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 9 : Évaluer le modèle à l'aide de mesures

- ❑ Le modèle de régression linéaire est évalué avec les mesures **mean\_squared\_error**, **mean\_absolute\_error** et **Score R<sup>2</sup> (coefficient de détermination)**.
- ❑ En comparant avec la moyenne de la variable cible, nous comprendrons dans quelle mesure notre modèle est prédictif.

➤ **mean\_squared\_error** (erreur quadratique moyenne) est la moyenne de la somme des carrés des résidus.

$$1/n \sum_{i=0}^n (y - \bar{y})^2$$

➤ **mean\_absolute\_error** (erreur absolue moyenn) st la moyenne de la somme des valeurs absolues des résidus.

$$1/n \sum_{i=0}^n |y - \bar{y}|$$

- ❑ Moins l'erreur est importante, meilleures sont les performances du modèle.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 9 : Évaluer le modèle à l'aide de mesures

- Le score  $R^2$  (également appelé coefficient de détermination) est une mesure utilisée pour évaluer la performance d'un modèle de régression.

$$R^2 = 1 - \frac{\sum (y_{\text{true}} - y_{\text{pred}})^2}{\sum (y_{\text{true}} - \bar{y})^2}$$

$R^2 = 1$  : Ajustement parfait. Le modèle prédit exactement les points de données.

$R^2 = 0$  : le modèle n'explique aucune variabilité de la variable cible ; les prédictions ne sont pas meilleures que la moyenne de la cible.

$R^2 < 0$  : le modèle est moins performant que la simple prédiction de la moyenne des valeurs cibles.

Cela signifie que les prédictions du modèle sont très médiocres.

Vous pouvez calculer le score  $R^2$  à l'aide de la fonction `r2_score` de **scikit-learn** après avoir généré vos prédictions.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 9 : Évaluer le modèle à l'aide de mesures

```
8 from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```
1 # évaluation du modèle
2 mse = mean_squared_error(y_test, y_pred)
3 mae = mean_absolute_error(y_test, y_pred)
4 r2 = r2_score(y_test, y_pred)
5
6 print(f'Mean Squared Error: {mse}')
7 print(f'Mean Absolute Error: {mae}')
8 print(f'R2 score: {r2}')
```

```
Mean Squared Error: 46.21179783493614
Mean Absolute Error: 5.392293684756542
R2 score: 0.6509058479986625
```

Si vous obtenez un score  $R^2$  de 0,65, cela signifie que 65 % de la variabilité de la variable cible peut être expliquée par le modèle en fonction des caractéristiques d'entrée.  
À l'inverse, 35 % de la variabilité est due à d'autres facteurs non pris en compte par le modèle.

Point clé à retenir :

Un  $R^2$  plus élevé signifie de meilleures performances du modèle.

Un  $R^2$  proche de 1 est idéal, tandis qu'une valeur négative ou 0 indique de mauvaises performances.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 10 : Faire une simple Prediction

```
1 print(multi_reg_linear.predict([[2012.917, 32.0, 84.87882, 10, 24.98298, 121.54024]]))  
[48.05954278]
```

❑ **Remarque importante** : Notez que les valeurs des caractéristiques ont toutes été saisies entre crochets doubles. C'est parce que la méthode « **predict** » attend toujours un tableau 2D comme format de ses entrées. Et mettre nos valeurs entre crochets doubles fait de l'entrée exactement un tableau 2D.

En termes simples :

2012.917, 32.0, 84.87882, 10, 24.98298, 121.54024 → Scalaires

[2012.917, 32.0, 84.87882, 10, 24.98298, 121.54024] → Tableau 1D

[[2012.917, 32.0, 84.87882, 10, 24.98298, 121.54024]] → Tableau 2D

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 10 : Faire une simple Prediction

```
1 print(multi_reg_linear.predict([[2012.917, 32.0, 84.87882, 10, 24.98298, 121.54024]]))  
[48.05954278]
```

- ✓ **multi\_reg\_linear.predict()** : cette méthode permet de réaliser des prédictions avec votre modèle de régression linéaire multiple entraîné.
- ✓ **[[2012.917, 32.0, 84.87882, 10, 24.98298, 121.54024]]** : il s'agit des nouvelles données d'entrée (ensemble de caractéristiques) pour lesquelles vous souhaitez réaliser une prédiction.

❑ Ces valeurs correspondent probablement aux caractéristiques suivantes :

- **Date de transaction** : 2012.917 (date de vente de la propriété).
- **Âge de la maison** : 32,0 ans.
- **Distance jusqu'à la station** : 84,87882 mètres.
- **Nombre de magasins à proximité** : 10 magasins.
- **Latitude** : 24,98298.
- **Longitude** : 121,54024

- ✓ **.predict()** : cette méthode génère la valeur cible prédite pour les caractéristiques d'entrée que vous fournissez.
- ✓ Dans ce cas, il s'agit du prix prévu de la maison par unité de surface.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.1 Régression linéaire Multiple:

#### 3.1.2 Prédire des valeurs continues (par exemple, prédire les prix des maisons)

##### Étape 11 : Obtenir l'équation de régression linéaire finale avec les valeurs des coefficients

- ❑ **Remarque importante** : pour obtenir ces coefficients, nous avons appelé les attributs « `coef_` » et « `intercept_` » de notre objet régresseur

```
1 print("Les coefficient de l'equation sont:",multi_reg_linear.coef_)
2 print("L'intercep est:", multi_reg_linear.intercept_)
```

```
Les coefficient de l'equation sont: [ 4.83926101e+00 -2.74749120e-01 -4.18860818e-03  1.18123112e+00
 2.42384317e+02  2.33991349e+01]
L'intercep est: -18595.055034519715
```

Par conséquent, l'équation de notre modèle de régression linéaire multiple est :

$$Y = 4,839 \times \text{Date de transaction} - 0,275 \times \text{Âge de la maison} - 0,0042 \times \text{Distance} + 1,181 \times \text{Nombre de magasins} + 242,384 \times \text{Latitude} + 23,399 \times \text{Longitude}$$

Les attributs en Python sont différents des méthodes et renvoient généralement une valeur simple ou un tableau de valeurs.



## CHAPITRE 3 APPRENTISSAGE SUPERVISE

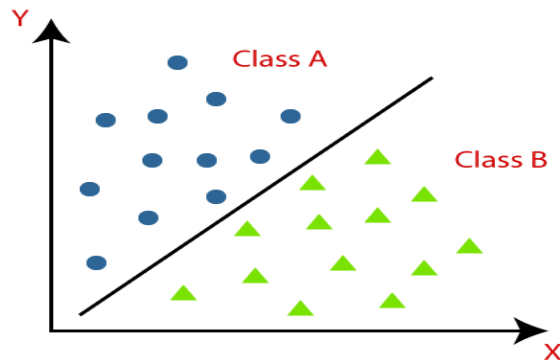


### 3.2 Classification

#### 3.2.1. Introduction à la classification binaire

- ❑ La classification binaire est un type d'apprentissage supervisé dont l'objectif est d'attribuer l'une des deux étiquettes (ou classes) possibles à chaque instance d'entrée.
- ❑ Elle est couramment utilisée dans des tâches telles que la détection de spam, les diagnostics médicaux, l'analyse des sentiments et la détection de fraude, où le résultat est soit « oui » soit « non », « positif » soit « négatif ».
- ❑ Concepts clés de la classification binaire :
  - ✓ **Classes** : Il existe deux catégories distinctes, souvent étiquetées 0 et 1 (ou négatives et positives).
  - ✓ **Apprentissage supervisé** : Vous entraînez le modèle sur des données étiquetées (fonctionnalités d'entrée associées à la classe de sortie correcte).

Donc, le modèle apprend à prédire l'étiquette de classe pour de nouvelles données (test data).



$Y = f(x)$ , ou  $y$  = sortie categorielle ou étiquettes prédites

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### 3.2.1. Introduction à la classification binaire

- ✓ **Algorithmes de classification** : Il existe différents algorithmes pour effectuer une classification binaire.
  1. **Régression logistique [Logistic Regression]** : un modèle statistique qui utilise une fonction logistique pour modéliser un résultat binaire.
  1. **Support Vector Machine (SVM)** : Classe les données en trouvant l'hyperplan qui sépare le mieux les deux classes.
  2. **Arbres de décision [Decision Trees] et Forêts aléatoires [Random Forest]** : Crée des modèles qui divisent les données en fonction des valeurs des caractéristiques.
  1. **Réseaux neuronaux** : Utiles pour les modèles complexes, en particulier dans l'apprentissage profond (Deep Learning).
  2. **K-Nearest Neighbors (k-NN)** : Classe un point de données en fonction de la classe majoritaire de ses k-plus proches voisins.
- ✓ **Mesures d'évaluation** : Comme il s'agit d'une tâche binaire, plusieurs mesures sont utilisées pour évaluer les performances du modèle : **Confusion Matrix, Accuracy, Precision, Recall, F1-Score**.
- ✓ **Limite de décision** : Dans la classification binaire, la limite de décision est la ligne ou la courbe qui sépare les données en deux classes.
  - Dans la régression logistique, par exemple, la limite est déterminée par un seuil de probabilité (par exemple, 0,5)

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning

Les métriques de classification visent à prédire les étiquettes de classe à partir des données d'entrée.

Dans la classification binaire, seules deux classes de sortie possibles existent; mais dans la classification multi-classe, plus de deux classes possibles peuvent être présentes. Nous allons parler uniquement sur la classification binaire.

Il existe de nombreuses façons de mesurer les performances de classification. La matrice de confusion, la précision [Accuracy], et l'AUC-ROC sont quelques-unes des mesures les plus populaires. La précision-rappel [Precision-Recall] est une mesure largement utilisée pour les problèmes de classification.

#### 1. Matrice de Confusion [Confusion Matrix]

Une matrice de confusion est un tableau utilisé pour évaluer les performances d'un modèle de classification, en particulier dans la classification binaire. Il fournit une répartition des prédictions du modèle, montrant la fréquence à laquelle il a correctement ou incorrectement prédit chaque classe.

➤ La matrice est généralement structurée comme suit:

	Prédite : Positive (1)	Prédite : Négatif (0)
Réelle : Positive (1)	True Positive (TP)	False Negative (FN)
Réelle : Négative (0)	False Positive (FP)	True Negative (TN)

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning

##### 1. Matrice de Confusion [Confusion Matrix]

	Prédite : Positive (1)	Prédite : Négatif (0)
Réelle : Positive (1)	True Positive (TP)	False Negative (FN)
Réelle : Négative (0)	False Positive (FP)	True Negative (TN)

➤ La matrice est généralement structurée comme suit:

- ✓ **True Positive (TP)**: Nombre de cas où le modèle a correctement prédit la classe positive (c'est-à-dire que la classe réelle est positive et que la classe prédite est également positive).
- ✓ **True Négatif (TN)** : Nombre de cas où le modèle a correctement prédit la classe négative (c'est-à-dire que la classe réelle est négative et que la classe prédite est également négative).
- ✓ **False Positif (FP)** : Egalement appelé « erreur de type I », cela se produit lorsque le modèle prédit de manière incorrecte la classe positive (c'est-à-dire que la classe réelle est négative, mais que la classe prédite est positive).
- ✓ **Faux Négatif (FN)** : Egalement appelé « erreur de type II », cela se produit lorsque le modèle prédit de manière incorrecte la classe négative (c'est-à-dire que la classe réelle est positive, mais que la classe prédite est négative).

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning



#### 1. Matrice de Confusion [Confusion Matrix]

❑ Exemple :

- Problème: Imaginez que vous construisiez un classificateur binaire pour déterminer si une personne est « diabétique » (1) ou « non diabétique » (0).

Après avoir évalué votre modèle sur un ensemble de données de test, la matrice de confusion pourrait ressembler à ceci :

	Prédite : Diabétique (1)	Prédite : Non Diabétique (0)
Réelle : Diabétique (1)	80 (TP)	10 (FN)
Réelle : Non Diabétique (0)	15 (FP)	95 (TN)

Dans cet exemple :

- 80 personnes ont été correctement classées comme diabétiques (TP).
- 95 personnes ont été correctement classées comme non diabétiques (TN).
- 15 personnes ont été incorrectement classées comme diabétiques (FP).
- 10 personnes ont été incorrectement classées comme non diabétiques (FN).

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning



### 2. Métriques dérivées de la matrice de confusion

✓ **Précision [Accuracy]** : la proportion d'instances correctement prédites.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

	Prédite : Diabétique (1)	Prédite : Non Diabétique (0)
Réelle : Diabétique (1)	80 (TP)	10 (FN)
Réelle : Non Diabétique (0)	15 (FP)	95 (TN)

➤ Sur la base de notre exemple, la **précision [Accuracy]** peut être calculée comme suit :

$$Accuracy = \frac{80 + 95}{80 + 95 + 15 + 10} = \frac{175}{200} = 0.875 \text{ ou } \mathbf{87.5\%}$$

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning



### 2. Métriques dérivées de la matrice de confusion

- ✓ **Precision** : La proportion de vraies positives parmi les positives prédites (c'est-à-dire combien de personnes « diabétiques » prédites étaient réellement diabétiques).

$$Precision = \frac{TP}{TP + FP}$$

	Prédite : Diabétique (1)	Prédite : Non Diabétique (0)
Réelle : Diabétique (1)	80 (TP)	10 (FN)
Réelle : Non Diabétique (0)	15 (FP)	95 (TN)

- Sur la base de notre exemple, la **precision** peut être calculée comme suit :

$$Precision = \frac{80}{80 + 15} = \frac{80}{95} = 0.842 \text{ ou } \mathbf{84.2\%}$$



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning

#### 2. Métriques dérivées de la matrice de confusion

- ✓ **Recall** : La proportion de vraies positives parmi toutes les positives réelles (c'est-à-dire combien de personnes « diabétiques » réelles ont été correctement identifiées).

$$Recall = \frac{TP}{TP + FN}$$

	Prédite : Diabétique (1)	Prédite : Non Diabétique (0)
Réelle : Diabétique (1)	80 (TP)	10 (FN)
Réelle : Non Diabétique (0)	15 (FP)	95 (TN)

- Sur la base de notre exemple, le **recall** peut être calculée comme suit :

$$Recall = \frac{80}{80 + 10} = \frac{80}{90} = 0.889 \text{ ou } \mathbf{88.9\%}$$

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

#### 3.2.2. Métriques de classification dans le Machine Learning



### 2. Métriques dérivées de la matrice de confusion

✓ **F1-Score** : La moyenne harmonique de la précision et du rappel, équilibrant les deux mesures.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

	Prédite : Diabétique (1)	Prédite : Non Diabétique (0)
Réelle : Diabétique (1)	80 (TP)	10 (FN)
Réelle : Non Diabétique (0)	15 (FP)	95 (TN)

➤ Sur la base de notre exemple, le **recall** peut être calculée comme suit :

$$F1 - Score = 2 * \frac{0.842 * 0.889}{0.842 + 0.889} = 2 * \frac{0.748538}{1.731} = 2 * 0.4324 = 0.8648 \text{ ou } 86.48 \text{ ou } \mathbf{86.5\%}$$

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

- ☐ Les prêts sont la principale exigence du monde moderne.
- ☐ Ce n'est que grâce à cela que les banques obtiennent une part importante du bénéfice total.
- ☐ Il est avantageux pour les étudiants de gérer leurs frais d'éducation et de subsistance, et pour les personnes d'acheter tout type de luxe comme des maisons, des voitures, etc.

Mais lorsqu'il s'agit de décider si le profil du candidat est pertinent pour obtenir un prêt ou non, les banques doivent tenir compte de nombreux aspects.

Nous allons donc ici utiliser l'apprentissage automatique avec Python pour faciliter leur travail et prédire si le profil du candidat est pertinent ou non en utilisant des fonctionnalités clés telles que l'état matrimonial, l'éducation, le revenu du candidat, l'historique de crédit, etc.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

□ Étapes à suivre :

1. **Collecte de données** : Collectez des données avec des résultats étiquetés.
2. **Prétraitement des données** : Nettoyez les données, gérez les valeurs manquantes, normalisez les caractéristiques et divisez-les en données d'entraînement et de test.
3. **Sélection du modèle** : choisissez un algorithme de classification binaire.
4. **Entraînement** : Entraînez le modèle à l'aide des données d'entraînement.
5. **Prédiction** : Utilisez le modèle entraîné pour faire des prédictions sur de nouvelles données.
6. **Évaluation** : Utilisez les données de test pour évaluer les performances à l'aide de mesures telles que l'exactitude, la précision, le rappel, etc.

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

##### Étape 1 : Importer les packages nécessaires et la dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

```
# Loading Data
df = pd.read_csv("../Data/LoanApprovalPrediction.csv")
df
```

```
[3]: df.shape
```

```
[3]: (598, 13)
```

```
[4]: df.columns
```

```
[4]: Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
         'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
         'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
         dtype='object')
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

##### Étape 2 : Prétraitement des données

```
df['Loan_Status'].value_counts()
```

```
Loan_Status
Y      411
N      187
Name: count, dtype: int64
```

```
df['Married'].value_counts()
```

```
Married
Yes    388
No     210
Name: count, dtype: int64
```

```
df['Education'].unique()
```

```
array(['Graduate', 'Not Graduate'], dtype=object)
```

```
df['Property_Area'].unique()
```

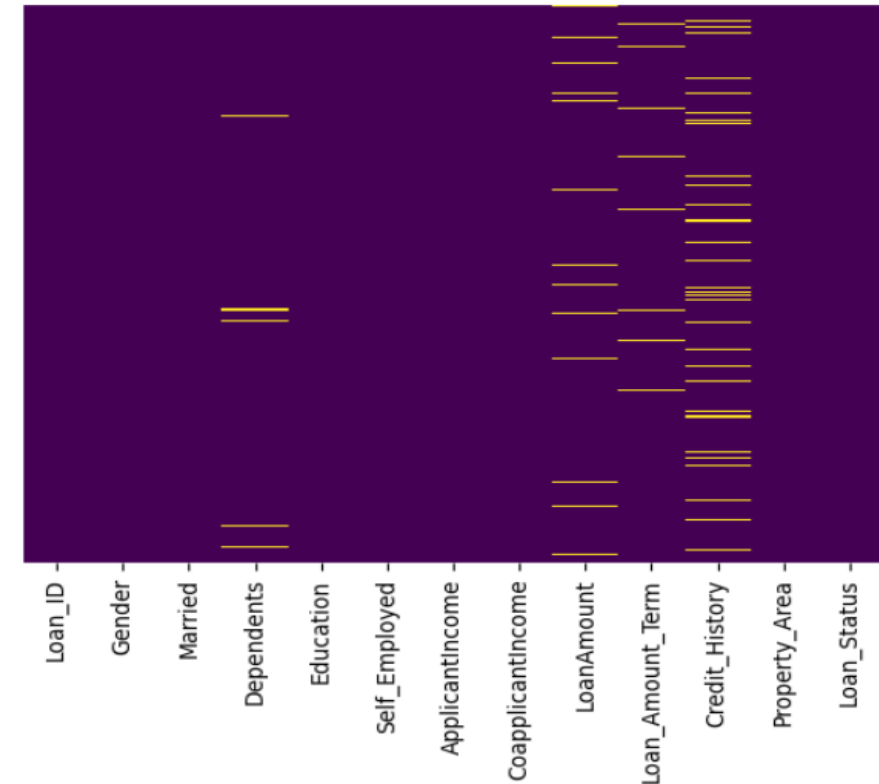
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

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

```
Loan_ID      0
Gender        0
Married       0
Dependents   12
Education     0
Self_Employed 0
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount   21
Loan_Amount_Term 14
Credit_History 49
Property_Area 0
Loan_Status   0
dtype: int64
```

```
[12]: #Vérifions maintenant les valeurs manquantes avec seaborn après imputation
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
[12]: <Axes: >
```



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

##### Étape 2 : Prétraitement des données

```
df.info()
```

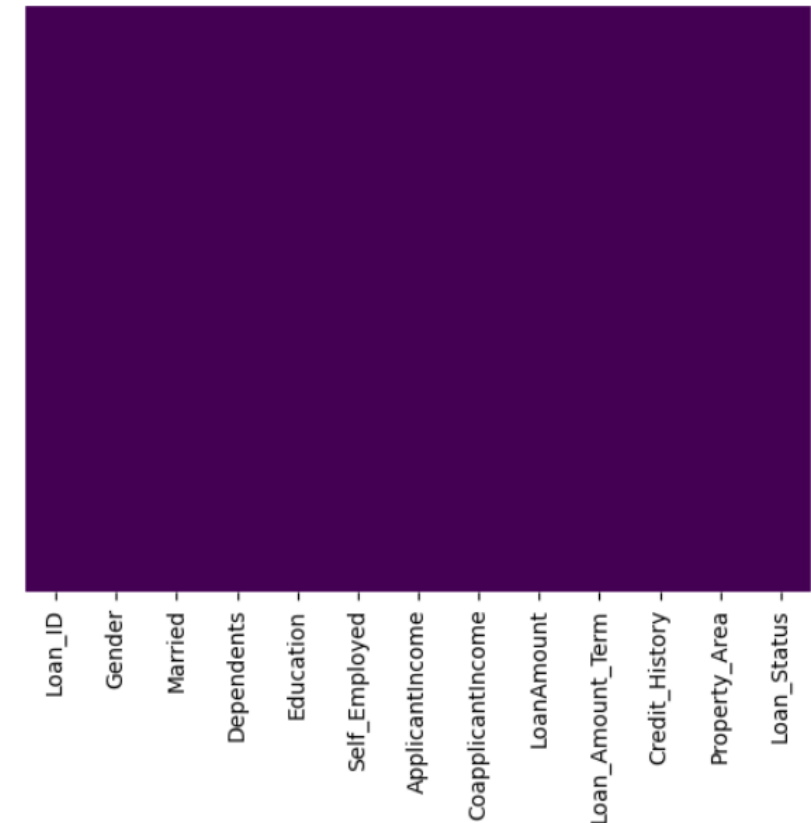
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 598 entries, 0 to 597
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Loan_ID               598 non-null   object 
1   Gender                598 non-null   object 
2   Married               598 non-null   object 
3   Dependents            586 non-null   float64
4   Education             598 non-null   object 
5   Self_Employed         598 non-null   object 
6   ApplicantIncome       598 non-null   int64  
7   CoapplicantIncome     598 non-null   float64
8   LoanAmount            577 non-null   float64
9   Loan_Amount_Term      584 non-null   float64
10  Credit_History         549 non-null   float64
11  Property_Area         598 non-null   object 
12  Loan_Status           598 non-null   object 
dtypes: float64(5), int64(1), object(7)
memory usage: 60.9+ KB
```

*#Nous allons imputer Les valeurs manquantes par la moyenne*

```
df['Dependents'].fillna(df['Dependents'].mean(), inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(), inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mean(), inplace=True)
```

```
: df.isnull().sum()
```

```
: Gender                0
   Married              0
   Dependents           0
   Education            0
   Self_Employed        0
   ApplicantIncome      0
   CoapplicantIncome    0
   LoanAmount           0
   Loan_Amount_Term     0
   Credit_History       0
   Property_Area        0
   Loan_Status          0
dtype: int64
```





## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

#### Étape 2 : Prétraitement des données

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

```
df['Gender'] = np.where(df['Gender'] == 'Male', 1, 0)
df['Married'] = np.where(df['Married'] == 'Yes', 1, 0)
df['Education'] = np.where(df['Education'] == 'Graduate', 1, 0)
df['Self_Employed'] = np.where(df['Self_Employed'] == 'Yes', 1, 0)

df['Property_Area'].unique()

array(['Urban', 'Rural', 'Semiurban'], dtype=object)

# Creation d'un dictionnaire pour le mapping
property_area_mapping = {
    'Urban': 0,
    'Rural': 1,
    'Semiurban': 2
}

# Appliquer le mapping a la colonne 'property_area'
df['Property_Area'] = df['Property_Area'].map(property_area_mapping)
```

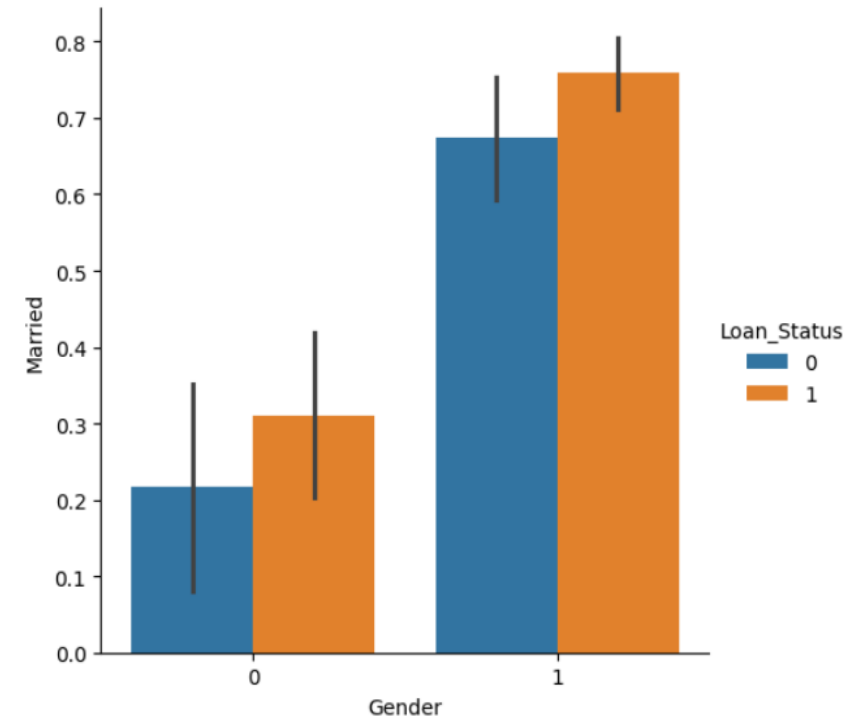
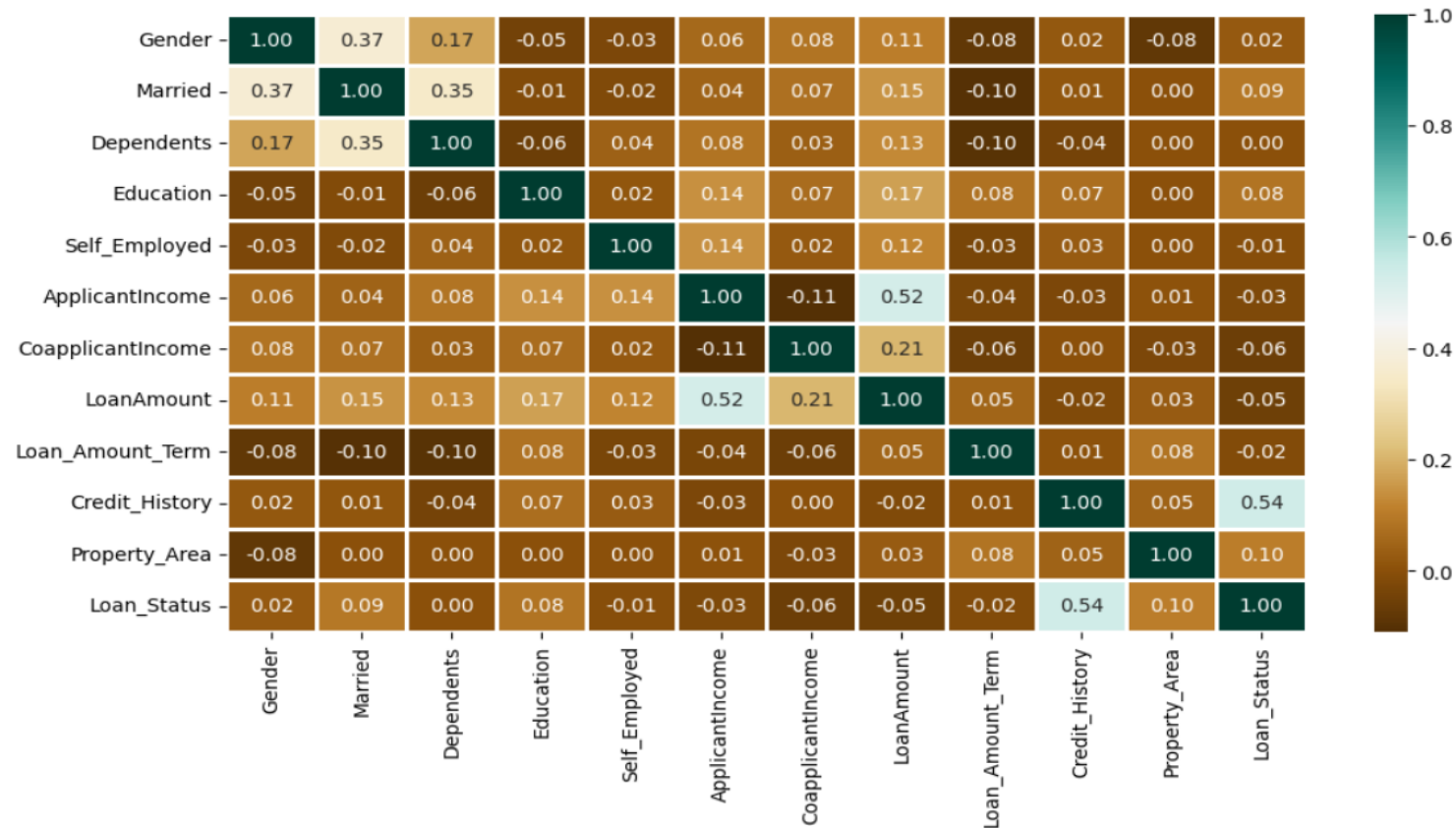
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	1	0	0.0	1	0	5849	0.0	144.968804
1	1	1	1.0	1	0	4583	1508.0	128.000000
2	1	1	0.0	1	1	3000	0.0	66.000000
3	1	1	0.0	0	0	2583	2358.0	120.000000
4	1	0	0.0	1	0	6000	0.0	141.000000

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

#### Étape 2 : Prétraitement des données : Matrice de corrélation des variables



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

Étape 2 : Prétraitement des données : Diviser de l'ensemble de données

```
from sklearn.model_selection import train_test_split

X = df.drop(['Loan_Status'], axis=1)
Y = df['Loan_Status']
X.shape, Y.shape

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4, random_state=1)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", Y_train.shape)
print("y_test:", Y_test.shape)
```

```
X_train: (358, 11)
X_test: (240, 11)
y_train: (358,)
y_test: (240,)
```

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

#### Étape 3 : Sélection du Modèle ou Algorithme

Lors de la sélection d'un algorithme d'apprentissage automatique supervisé, vous devez prendre en compte divers facteurs tels que la nature du problème, la taille et la qualité de l'ensemble de données, ainsi que les mesures de performances les plus pertinentes pour votre cas d'utilisation.

##### ❖ Régression logistique [Logistic Regression]

```
# Sélection du modèle
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
```

#### Étape 4 : Entraînement du Modèle ou Algorithme

```
# Entraînement
lr_model.fit(X_train, Y_train)
```

```
▼ LogisticRegression ⓘ ?
LogisticRegression()
```

#### Étape 5 : Prédiction de la classe

```
# Prédiction
y_pred_lr = lr_model.predict(X_test)
y_pred_lr

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

#### Étape 6 : Évaluation

```
# Évaluation
print("Accuracy de la Régression logistique:", 100 * metrics.accuracy_score(Y_test, y_pred_lr))
print("MSE de la Régression logistique:", mean_squared_error(Y_test, y_pred_lr))
```

```
Accuracy de la Régression logistique: 81.66666666666667
MSE de la Régression logistique: 0.18333333333333332
```

# Machine Learning

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

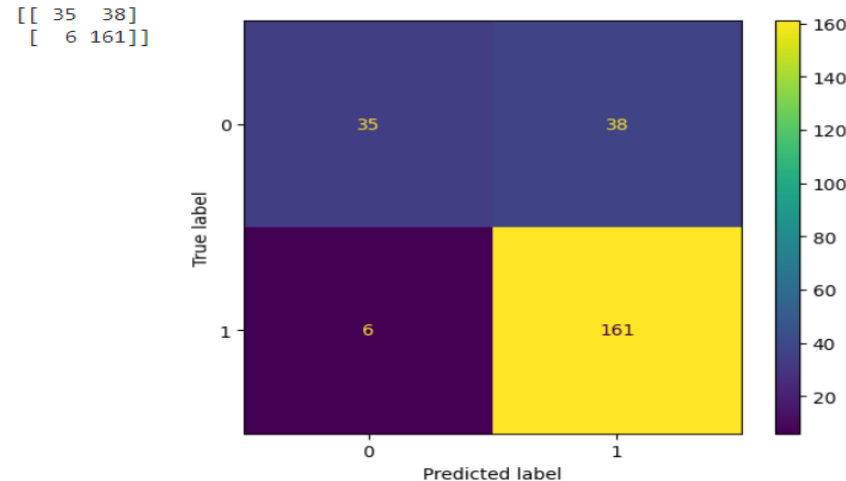
#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

##### Étape 6 : Évaluation : Matrice de Confusion [Confusion Matrix]

Test Data = 35 + 38 + 161 + 6 = 240

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

print(confusion_matrix(Y_test, y_pred_lr))
_ = ConfusionMatrixDisplay.from_estimator(lr_model, X_test, Y_test)
```



```
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", Y_train.shape)
print("y_test:", Y_test.shape)
```

```
X_train: (358, 11)
X_test: (240, 11)
y_train: (358,)
y_test: (240,)
```

Preuve:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

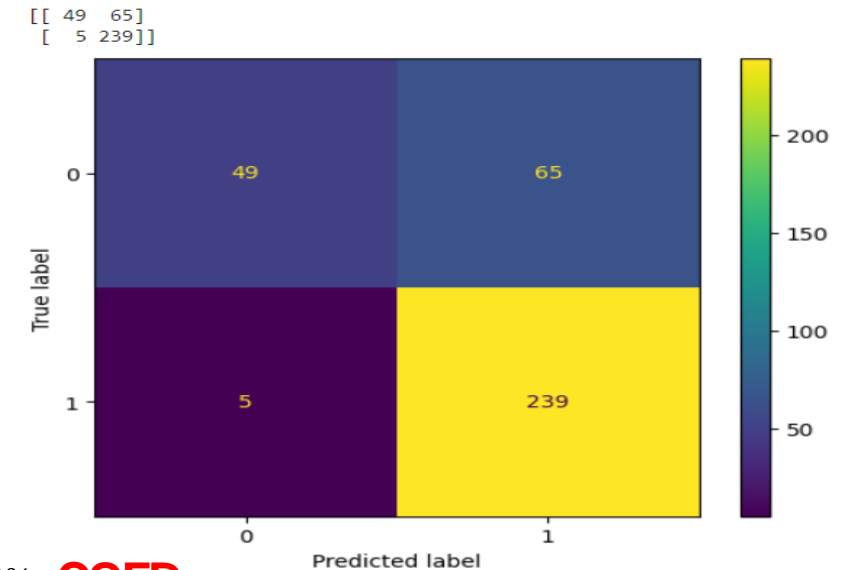
$$Accuracy = \frac{161 + 35}{161 + 35 + 38 + 6} = \frac{196}{240} = 0.816 \text{ ou } 81.6\% \text{ CQFD}$$

Cours-ML-Dahouda M., Ph.D.



Train Data = 49 + 65 + 5 + 239 = 358

```
print(confusion_matrix(Y_train, y_pred_lr_train))
_ = ConfusionMatrixDisplay.from_estimator(lr_model, X_train, Y_train)
```



Reelle

	Négatif (0)	Positive (1)
Négative (0)	35 (TN)	38 (FP)
Positive (1)	6 (FN)	161 (TP)

Prédite

## CHAPITRE 3 APPRENTISSAGE SUPERVISE

### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

##### Étape 6 : Évaluation : Matrice de Confusion [Confusion Matrix]

Reelle \ Prédite	Négatif (0)	Positive (1)
	Négative (0)	Positive (1)
Négative (0)	35 (TN)	38 (FP)
Positive (1)	6 (FN)	161 (TP)

Prédite

$$✓ \text{ Precision} = \frac{TP}{TP+FP} = \frac{161}{161+38} = \frac{161}{199} = 0.809 \text{ ou } 80.9\%$$

$$✓ \text{ Recall} = \frac{TP}{TP+FN} = \frac{161}{161+6} = \frac{161}{167} = 0.964 \text{ ou } 96.4\%$$

$$✓ \text{ F1 - Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = 2 * \frac{0.809 * 0.964}{0.809 + 0.964} = 2 * \frac{0.779876}{1.773} = 2 * 0.438 = 0.879 \text{ ou } 87.9\%$$



```
from sklearn import metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
```

```
print("Precision Score de la Régression logistique:", precision_score(Y_test, y_pred_lr))
print("Recall Score de la Régression logistique:", recall_score(Y_test, y_pred_lr))
print("F1-Score de la Régression logistique:", f1_score(Y_test, y_pred_lr))
```

```
Precision Score de la Régression logistique: 0.8090452261306532
Recall Score de la Régression logistique: 0.9640718562874252
F1-Score de la Régression logistique: 0.8797814207650273
```

**CQFD**

## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.4. Comparaison de performance des algorithmes

##### 1. KNN

```
# Sélection du modèle
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=3)

# Entraînement
knn_model.fit(X_train, Y_train)
```

```
▼ KNeighborsClassifier ⓘ ?
KNeighborsClassifier(n_neighbors=3)
```

```
# Prédiction
y_pred_knn = knn_model.predict(X_test)

# Évaluation
print("Accuracy score of knn model ", 100 * metrics.accuracy_score(Y_test, y_pred_knn))
print("MSE of knn model ", mean_squared_error(Y_test, y_pred_knn))

Accuracy score of knn model  63.74999999999999
MSE of knn model  0.3625
```

##### 2. Random Forest

```
# Sélection du modèle
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators = 7, criterion = 'entropy', random_state = 7)

# Entraînement
rf_model.fit(X_train, Y_train)
```

```
▼ RandomForestClassifier ⓘ ?
RandomForestClassifier(criterion='entropy', n_estimators=7, random_state=7)
```

```
# Prédiction
y_pred_rf = rf_model.predict(X_test)

# Évaluation
print("Accuracy score of rf model ", 100 * metrics.accuracy_score(Y_test, y_pred_rf))
print("MSE of rf model ", mean_squared_error(Y_test, y_pred_rf))

Accuracy score of rf model  78.33333333333333
MSE of rf model  0.21666666666666667
```



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

### Comparaison des algorithmes

#### 3. Support Vector Machine

```
# Sélection du modèle
from sklearn.svm import SVC
svc_model = SVC()

# Entraînement
svc_model.fit(X_train, Y_train)
```

▼ SVC ⓘ ?

SVC()

```
# Prédiction
y_pred_svc = svc_model.predict(X_test)

# Évaluation
print("Accuracy score of svc model ", 100 * metrics.accuracy_score(Y_test, y_pred_svc))
print("MSE of svc model ", mean_squared_error(Y_test, y_pred_svc))
```

Accuracy score of svc model 69.16666666666667  
MSE of svc model 0.30833333333333335

#### 4. Logistic Regression

```
# Sélection du modèle
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
```

```
# Entraînement
lr_model.fit(X_train, Y_train)
```

▼ LogisticRegression ⓘ ?

LogisticRegression()

```
# Prédiction
y_pred_lr = lr_model.predict(X_test)

# Évaluation
print("Accuracy de la Régression logistique:", 100 * metrics.accuracy_score(Y_test, y_pred_lr))
print("MSE de la Régression logistique:", mean_squared_error(Y_test, y_pred_lr))
```

Accuracy de la Régression logistique: 81.66666666666667  
MSE de la Régression logistique: 0.18333333333333332



## CHAPITRE 3 APPRENTISSAGE SUPERVISE



### 3.2 Classification

#### - 3.2.3. Prédiction de l'approbation des prêts à l'aide de Machine Learning

#### Comparaison des algorithmes

Algorithms	Accuracy	Mean Squared Error
k-Nearest Neighbors (KNN)	63.7%	0.3625
Random Forest	78.3%	0.2166
Support Vector Machine (SVM)	69.1%	0.308
Logistic Regression	81.6%	0.1833

#### ❑ Résumé :

Meilleure précision : Logistic Regression (81,6 %)

Meilleur MSE : régression logistique (0,1833)

Deuxième meilleur : Random Forest est également performant avec une précision élevée (78,3 %) et un MSE relativement faible (0,2166).

❑ Compte tenu de cela, la régression logistique semble être l'algorithme le plus performant pour votre ensemble de données, suivi de près par Random Forest.