Intelligence Artificielle En Réseaux

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

- Assisté Ass. Grace PWELE

- E-mail : dahouda37@gmail.com

Heure : 08H00 – 12H00

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.1 Régression linéaire : Apprentissage supervisé

- 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 : Apprentissage supervisé

- 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

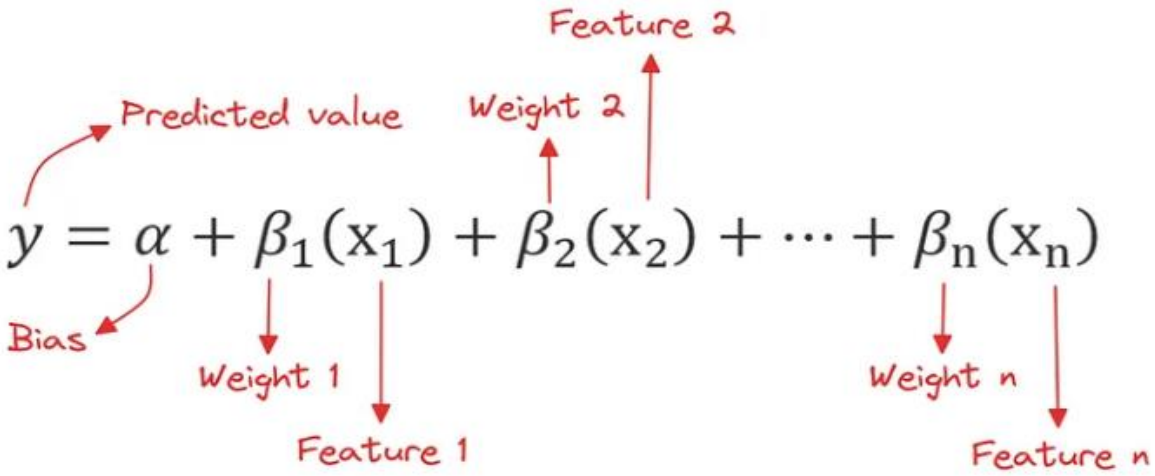
3.3 Clustering : Apprentissage non supervisé :

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 indépendante) 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

Formulation Mathématique:



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

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

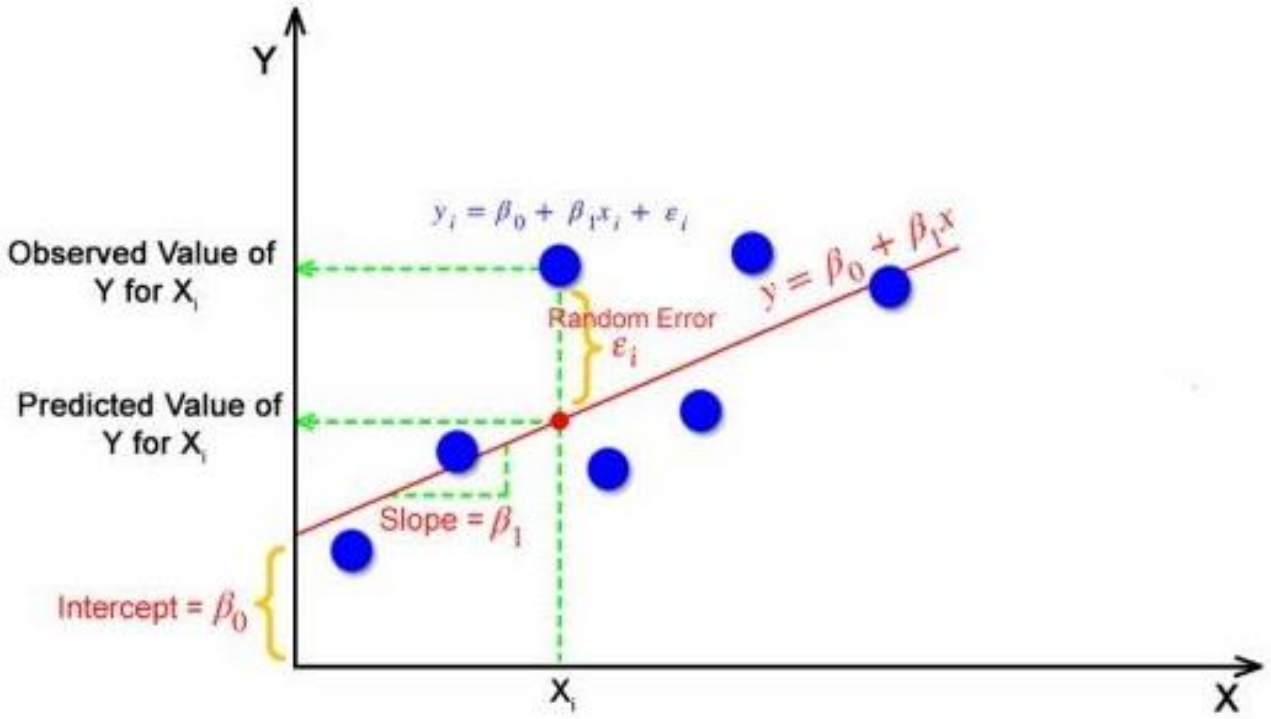
$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$,… sont des variables indépendantes.
$\mathbf{b_0}$ = interception de la droite.
$\mathbf{b_1}$, $\mathbf{b_2}$, … sont des coefficients.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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:

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

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

$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$,… sont des variables indépendantes.
$\mathbf{b_0}$ = interception de la droite.
$\mathbf{b_1}$, $\mathbf{b_2}$, … sont des coefficients.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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'entrainement 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 tâcun échantillon ou une observation individuelle dans l'ensemble de données.
- ✓ **Colonnes** (caractéristiques/attributs) : chaque colonne représente une caractéristiquehes d'apprentissage supervisé.

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 Techniques d'IA utilisées dans le réseaux informatique

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

```
# Importation de modules et de packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing

```

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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

```
# Importation d'un ensemble de données
df = pd.read_csv("../Data/Real-estate.csv")
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 Techniques d'IA utilisées dans le réseaux informatique

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

```
df.drop('No', inplace = True,axis=1)
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
: (414, 7)
```

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

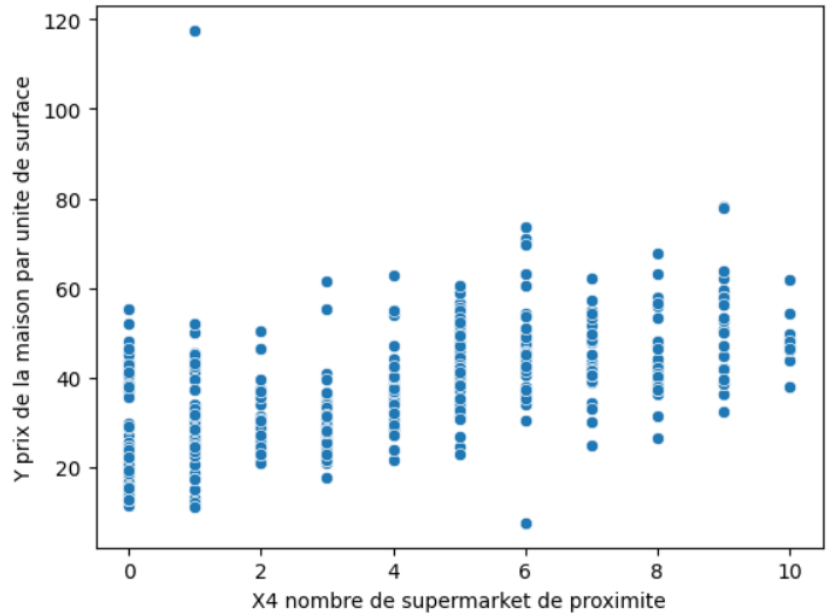
3.1 Régression linéaire Multiple:

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 Techniques d'IA utilisées dans le réseaux informatique

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.

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

- **Les variables de caractéristiques X**

	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

- **La variables cible Y**

	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

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

```
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: (289, 6)
X_test: (125, 6)
y_train: (289,)
y_test: (125,)
```

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.

❑ **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.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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**.

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

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

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

```
# Adapter le modèle aux données d'entraînement
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 Techniques d'IA utilisées dans le réseaux informatique

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 entrainé (**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 Techniques d'IA utilisées dans le réseaux informatique

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 R² (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.

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

- mean_absolute_error (erreur absolue moyenn) st la moyenne de la somme des valeurs absolues des résidus.

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

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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 entrainé.
- ✓ **[[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'agir du prix prévu de la maison par unité de surface.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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×Date de transaction−0,275×Âge de la maison−0,0042×Distance +1,181×Nombre de magasins+242,384×Latitude+23,399×Longitude

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

TD 4

1. Expliquez les termes suivants:

a. Modele de machine Learning

b. Entrainer un modèle

c. Tester un modèle

2. Expliquer cette function : train_test_split(x, y, test_size = 0.24)

3. Soit la dimension suivante (598, 13), Expliquez et diviser en Training data et test Data (80/20)

4. Expliquer l'Accuracy (Exactitude) et la MSE (La perte)

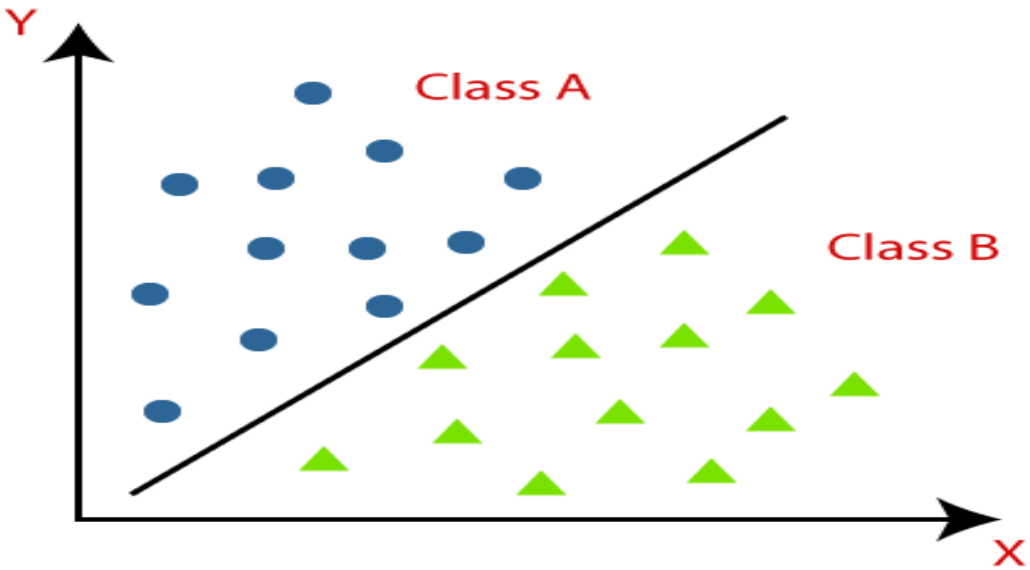
CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 »,



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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

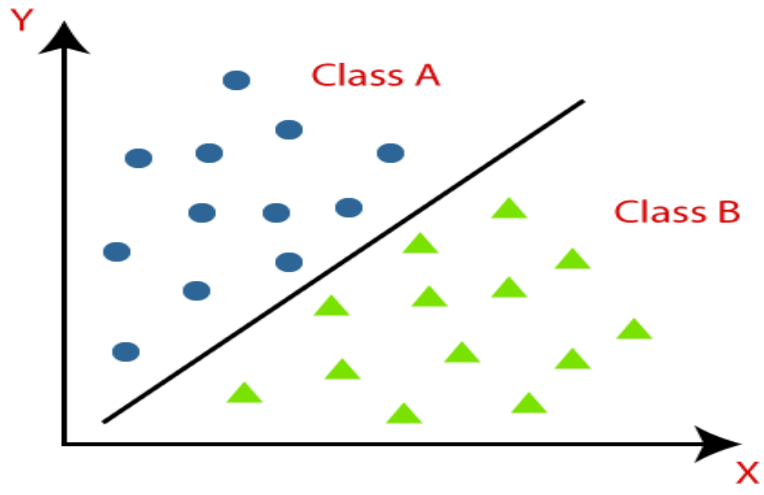
3.2 Classification

3.2.1. Introduction à la classification binaire

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 predite

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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.

2. **Support Vector Machine (SVM)** : Classe les données en trouvant l'hyperplan qui sépare le mieux les deux classes.

3. **Arbres de décision [Decision Trees]** et **Forêts aléatoires [Random Forest]** : Crée des modèles qui divisent les données en fonction des valeurs des caractéristiques.

4. **Réseaux neuronaux** : Utiles pour les modèles complexes, en particulier dans l'apprentissage profond (Deep Learning).

5. **K-Nearest Neighbors (k-NN)** : Classe un point de données en fonction de la classe majoritaire de ses k-plus proches voisins.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.2 Classification

3.2.1. Introduction à la classification binaire

- ✓ **Mesures d'évaluation** :Comme il s'agit d'une tâche binaire, plusieurs mesures sont utilisées pour évaluer les performances du modèle : Accuracy, Precision, Recall, F1-Score, Confusion Matrix.

- ✓ **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)

IA (Réseau et Telecom)-Dahouda, Ph.D

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.2 Classification

3.2.2. Métriques de classification dans le Machine Learning

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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 :

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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 :

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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 :

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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 :

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

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.2 Classification

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

Nous allons donc ici utiliser les algorithmes de Machine Learning avec Python pour faciliter leur travail et prédire si le profil d u 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.

❑ 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 (Training data).
5. **Prédiction** : Utilisez le modèle entraîné pour faire des prédictions sur de nouvelles données (Test data).
6. **Évaluation** : Utilisez les données de test pour évaluer les performances à l'aide de mesures telles que l'exactitude, la précision, le rappel.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 de 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')
```

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 de la dataset

	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

IA (Réseau et Telecom)-Dahouda, Ph.D

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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()
```

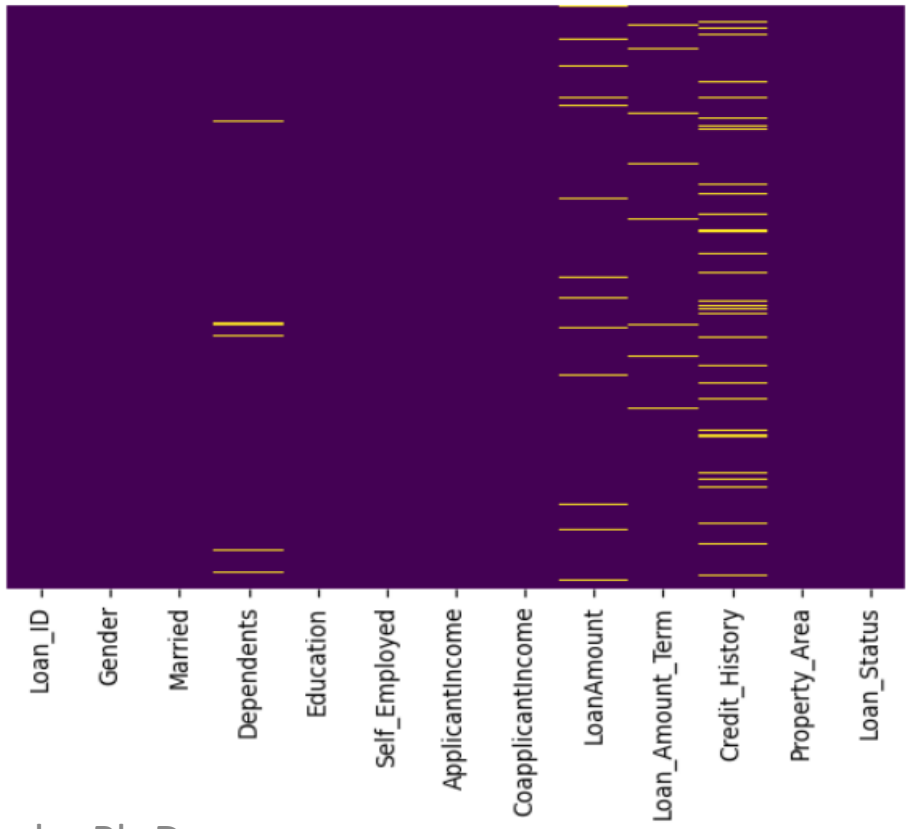
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

```
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 sibornes apres imputation
      sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

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



CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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()
```

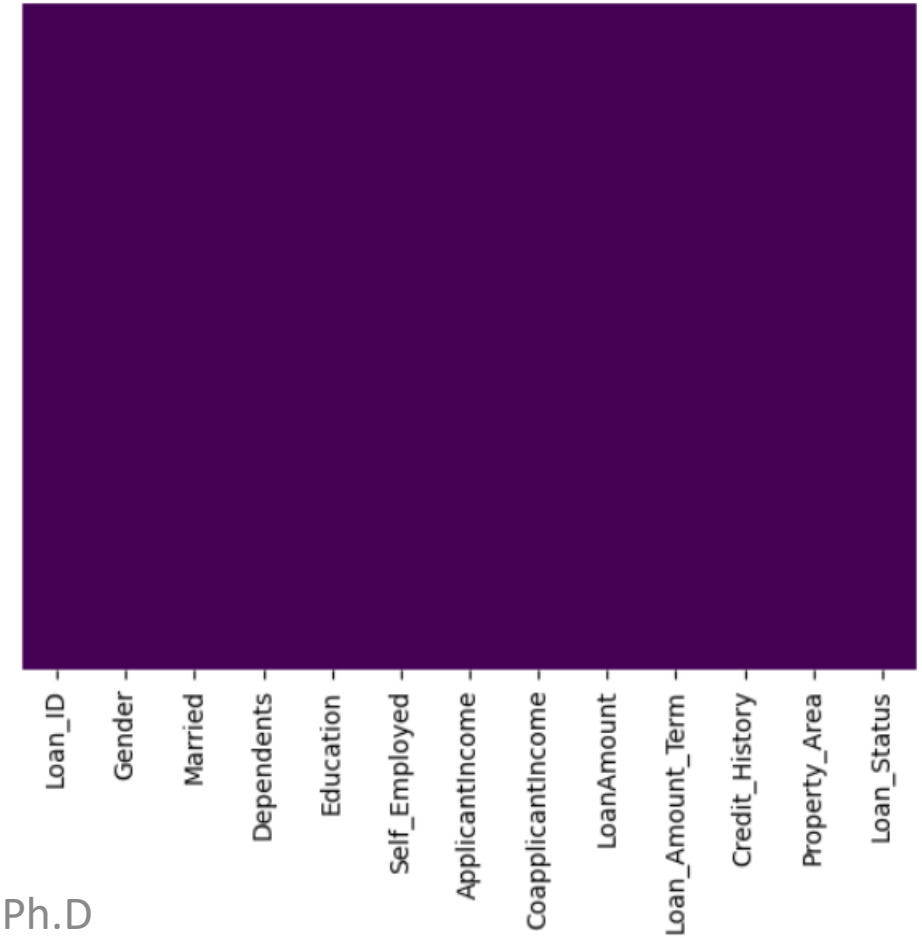
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 598 entries, 0 to 597
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            598 non-null    object
 1   Gender             598 non-null    object
 2   Married            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 Techniques d'IA utilisées dans le réseaux informatique

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 dictionary 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)
```

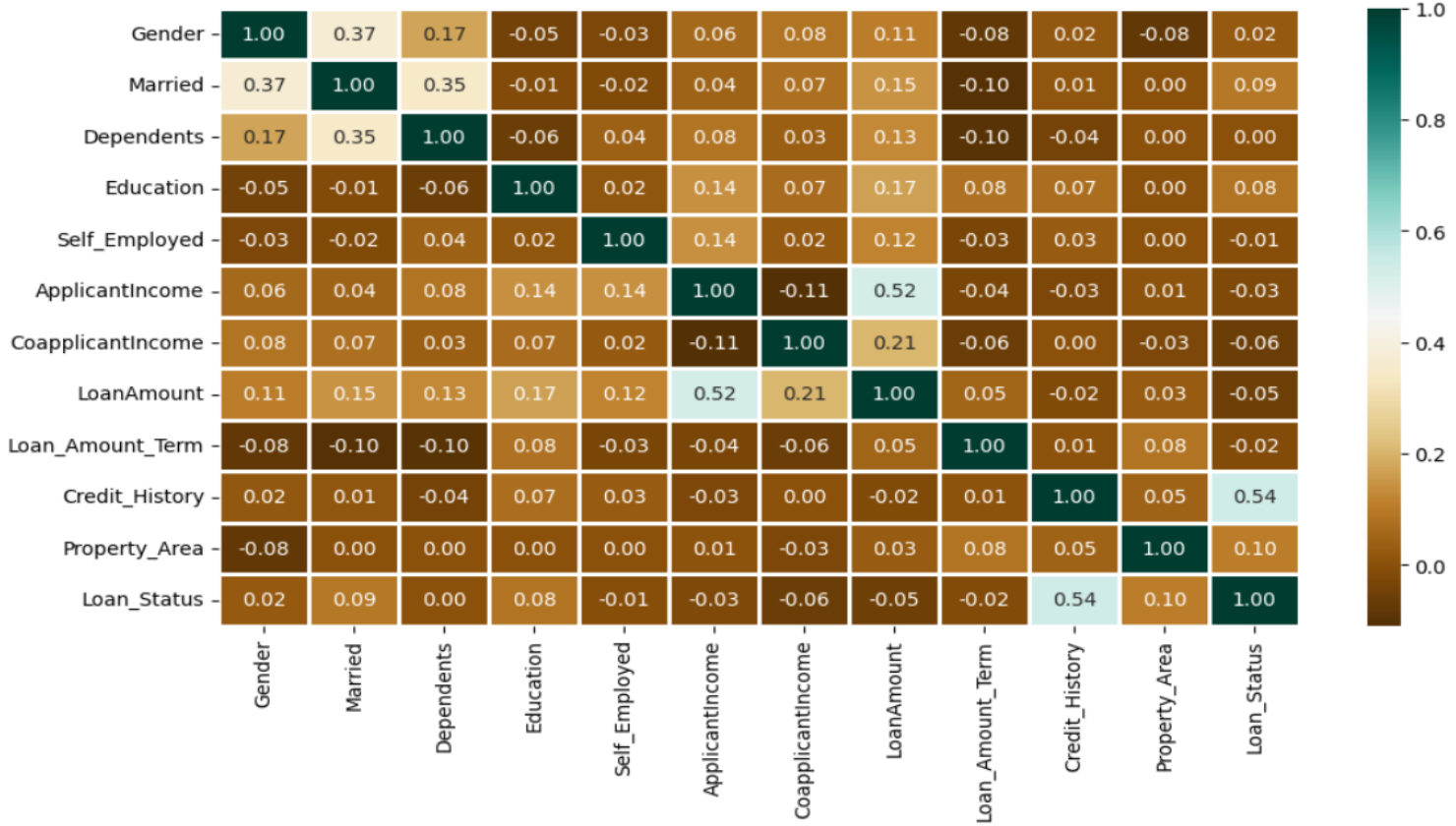
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	1	0	0.0	1	0	5849	0.0	144.968804
1	1	1	1.0	1	0	4583	1508.0	128.000000
2	1	1	0.0	1	1	3000	0.0	66.000000
3	1	1	0.0	0	0	2583	2358.0	120.000000
4	1	0	0.0	1	0	6000	0.0	141.000000

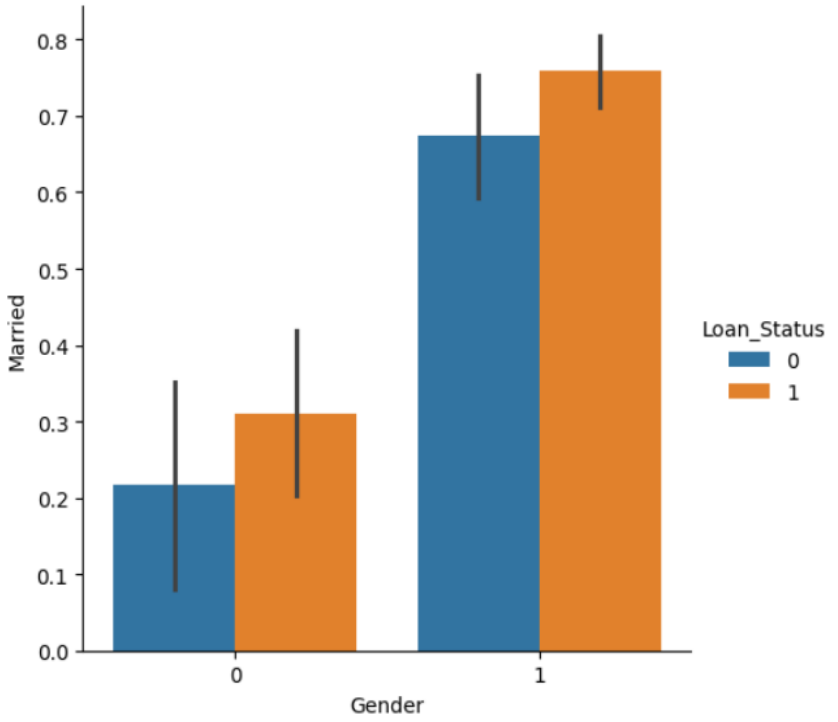
CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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,
```

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
```

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.2 Classification

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

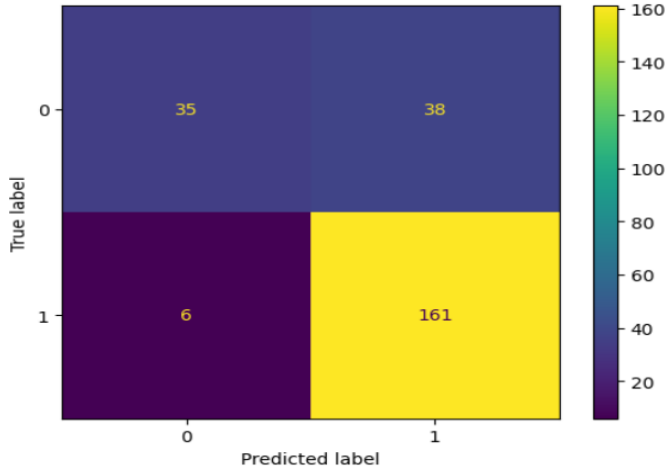
Étape 6 : Évaluation : Matrice de Confusion [Confusion Matrix]

$$\text{Test Data} = 35 + 38 + 161 + 6 = 240$$

```
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)
```

```
[[ 35  38]
 [  6 161]]
```



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

```
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:

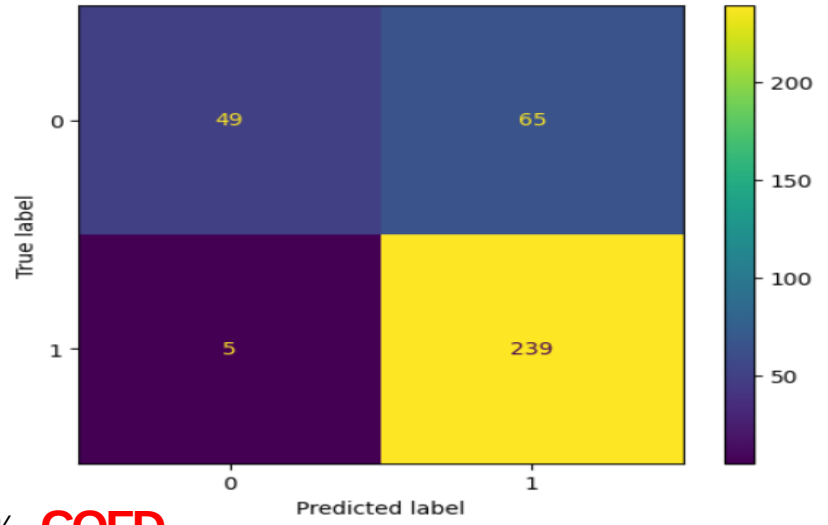
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

$$\text{Train Data} = 49 + 65 + 5 + 239 = 358$$

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

```
[[ 49  65]
 [  5 239]]
```



CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

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

Prédite

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

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

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

```
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 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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 Techniques d'IA utilisées dans le réseaux informatique

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%	01833

❑ 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.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3 Clustering : Apprentissage non supervisé

3.3.1. K-Means Clustering Algorithm

3.3.2. Définition de Clustering k-Means

3.3.3. Fonctionnement de K-Means Clustering Algorithm

- Comment choisir la valeur de « nombre K de clusters » ?
- Elbow Method
- Étapes de la méthode Elbow

3.3.4. Implémentation Python de l'algorithme de clustering K-means

- Segmentation de la clientèle avec l'algorithme de clustering K-means

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3 Clustering : Apprentissage non supervisé

3.3.1 K-Means Clustering Algorithm

Le clustering K-Means est un algorithme d'apprentissage non supervisé utilisé pour résoudre les problèmes de clustering dans le machine learning ou la science des données.

Dans ce chapitre, nous apprendrons ce qu'est l'algorithme de clustering K-means, comment l'algorithme fonctionne, ainsi que l'implémentation Python du clustering K-means.

3.3.2. Définition de Clustering k-Means

- Le clustering K-Means est un algorithme d'apprentissage non supervisé qui regroupe l'ensemble de données non étiqueté en différents clusters.

- Et K définit le nombre de clusters prédéfinis qui doivent être créés dans le processus, si K=2, il y aura deux clusters, et pour K=3, il y aura trois clusters, et ainsi de suite.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3 Clustering : Apprentissage non supervisé

3.3.1 K-Means Clustering Algorithm

3.3.2. Définition de Clustering k-Means

- Il s'agit d'un algorithme itératif qui divise l'ensemble de données non étiqueté en k clusters différents de telle sorte que chaque ensemble de données n'appartienne qu'à un seul groupe ayant des **propriétés similaires**.

Il nous permet de regrouper les données en différents groupes et constitue un moyen pratique de découvrir les catégories de groupes dans l'ensemble de données non étiqueté sans avoir besoin d'aucun entrainement.

- Il s'agit d'un algorithme basé sur le **centroïde**, où chaque cluster est associé à un centroïde.

- L'objectif principal de cet algorithme est de minimiser la somme des distances entre le point de données et leurs clusters correspondants.

IA (Réseau et Telecom)-Dahouda, Ph.D

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

❑ L'algorithme prend l'ensemble de données non étiqueté comme entrée, divise l'ensemble de données en k-nombre de clusters et répète le processus jusqu'à ce qu'il ne trouve pas les meilleurs clusters.

La valeur de k doit être prédéterminée dans cet algorithme.

❑ L'algorithme de clustering k-means effectue principalement deux tâches :

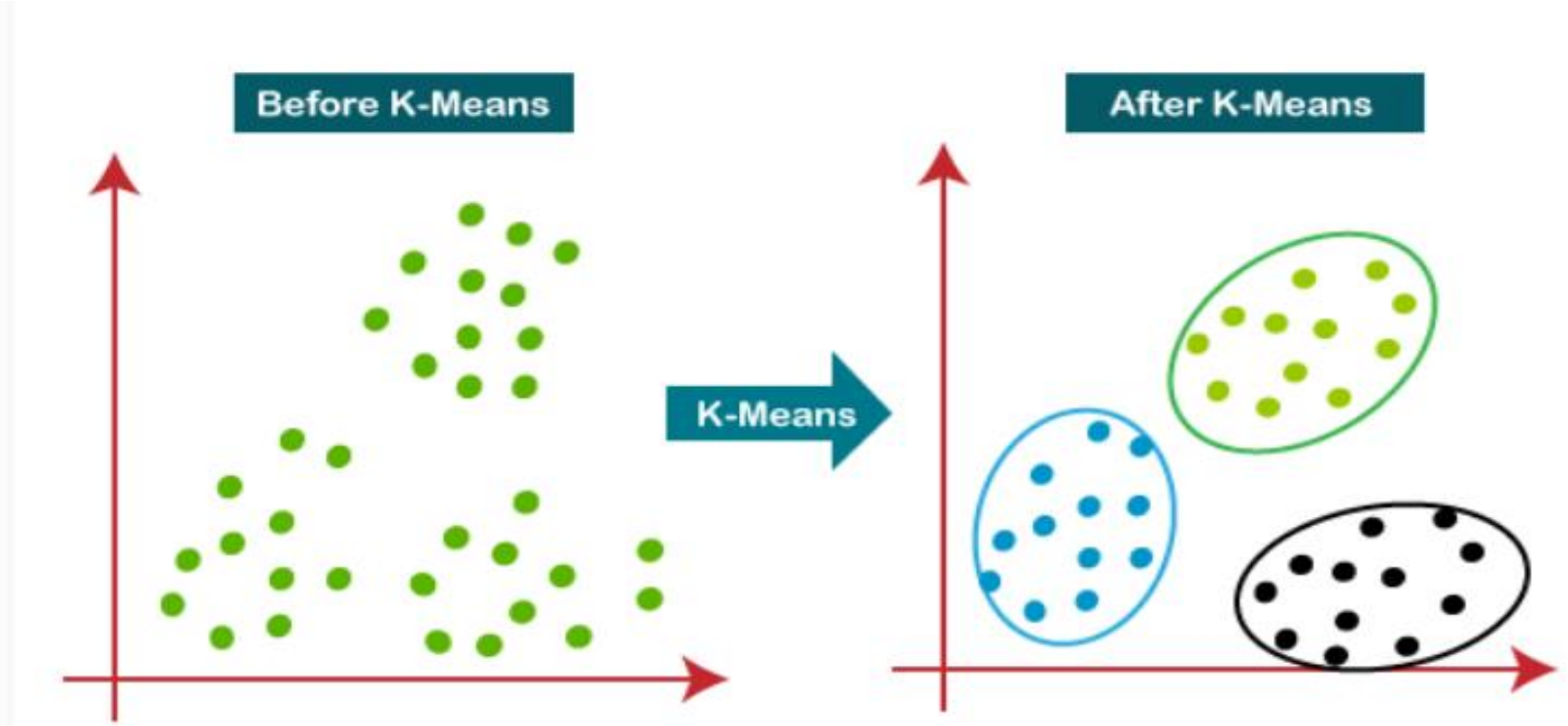
✓ Détermine la meilleure valeur pour K points centraux ou centroïdes par un processus itératif.

✓ Affecte chaque point de données à son centre k le plus proche.
Les points de données qui sont proches du centre k particulier créent un cluster.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

❑ Le diagramme ci-dessous explique le fonctionnement de l'algorithme de clustering K-means :



CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3. Fonctionnement de K-Means Clustering Algorithm

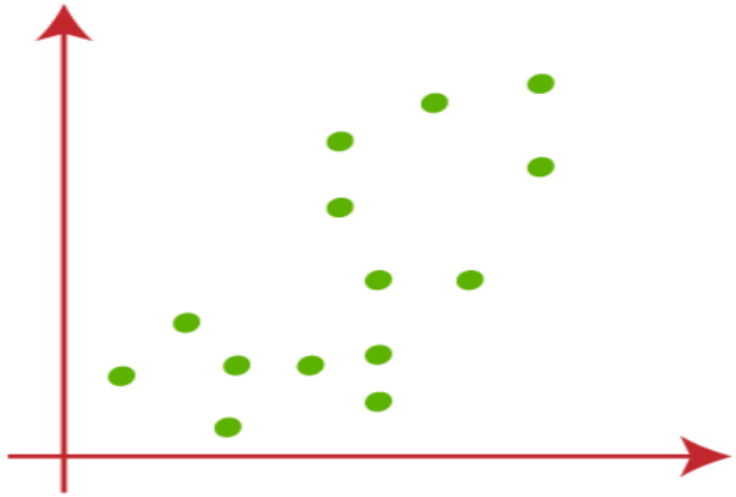
Le fonctionnement de l'algorithme K-Means est expliqué dans les étapes ci-dessous :

Étape 1 : sélectionnez le nombre K pour déterminer le nombre de clusters.
Étape 2 : sélectionnez des points K ou des centroïdes aléatoires. (Il peut s'agir d'autres éléments de l'ensemble de données d'entrée).
Étape 3 : attribuez chaque point de données à son centroïde le plus proche, qui formera les clusters K prédéfinis.
Étape 4 : calculez la variance et placez un nouveau centroïde de chaque cluster.
Étape 5 : répétez les troisièmes étapes, ce qui signifie réaffecter chaque point de données au nouveau centroïde le plus proche de chaque cluster.
Étape 6 : si une réaffectation se produit, passez à l'étape 4, sinon passez à TERMINER.
Étape 7 : le modèle est prêt.

Comprenons les étapes ci-dessus en considérant les tracés visuels :

- Supposons que nous ayons deux variables M1 et M2.

Le diagramme de scatter de l'axe x-y de ces deux variables est le suivant:



CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

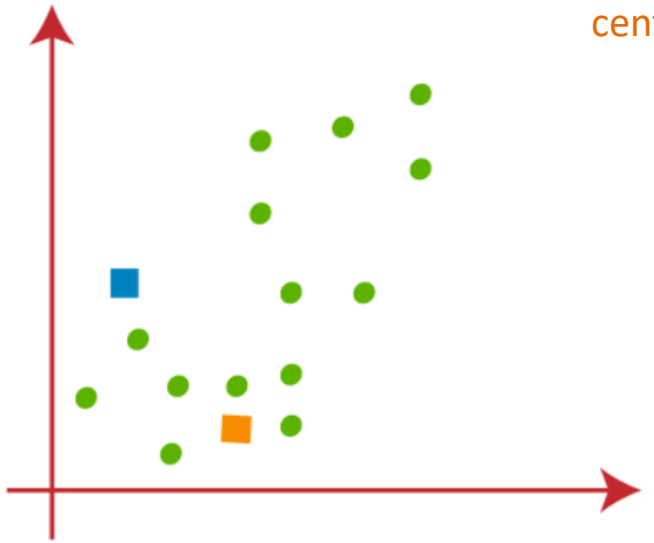
3.3.3 Fonctionnement de K-Means Clustering Algorithm

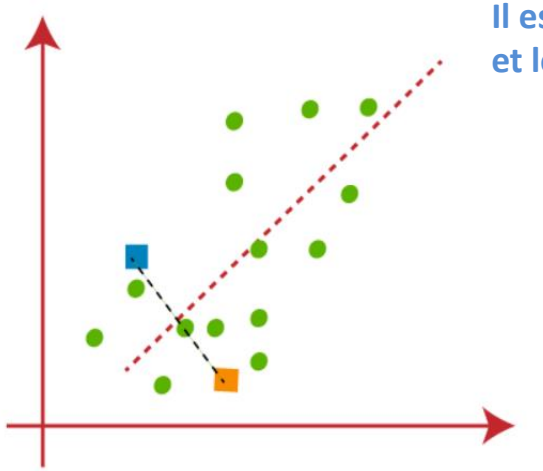
❖ Prenons un nombre k de clusters, c'est-à-dire K=2, pour identifier l'ensemble de données et les placer dans différents clusters.
Cela signifie qu'ici nous allons essayer de regrouper ces ensembles de données en deux clusters différents.

❖ Nous devons choisir des points k aléatoires ou un centroïde pour former le cluster.
Ces points peuvent être soit les points de l'ensemble de données, soit tout autre point.

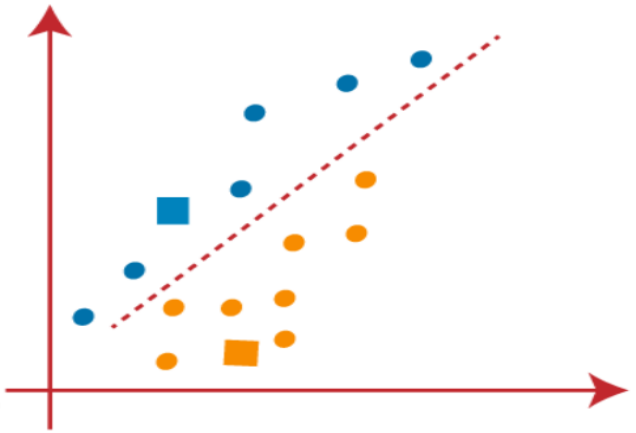
❖ Nous sélectionnons donc ici les deux points ci-dessous comme points k, qui ne font pas partie de notre ensemble de données.

Nous allons maintenant attribuer chaque point de données du nuage de points à son point K ou centroïde le plus proche.





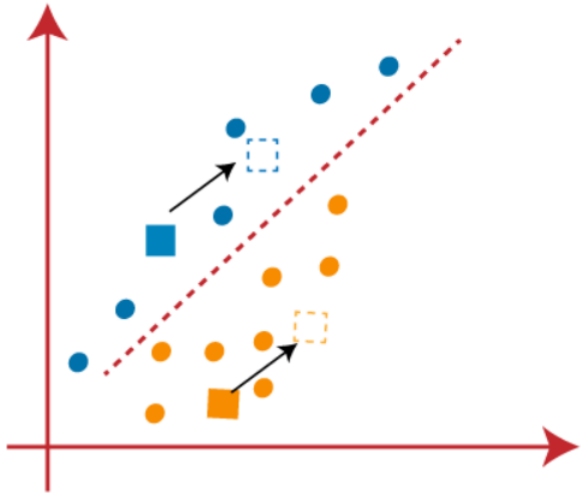
Il est clair que les points à gauche de la ligne sont proches du centroïde K1 ou bleu, et les points à droite de la ligne sont proches du centroïde jaune.



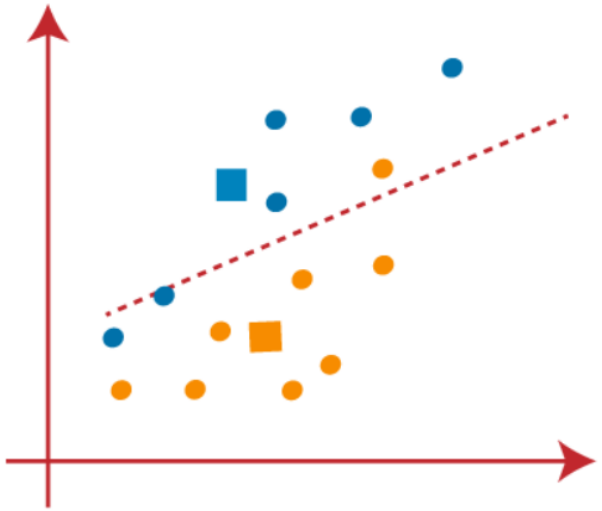
CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

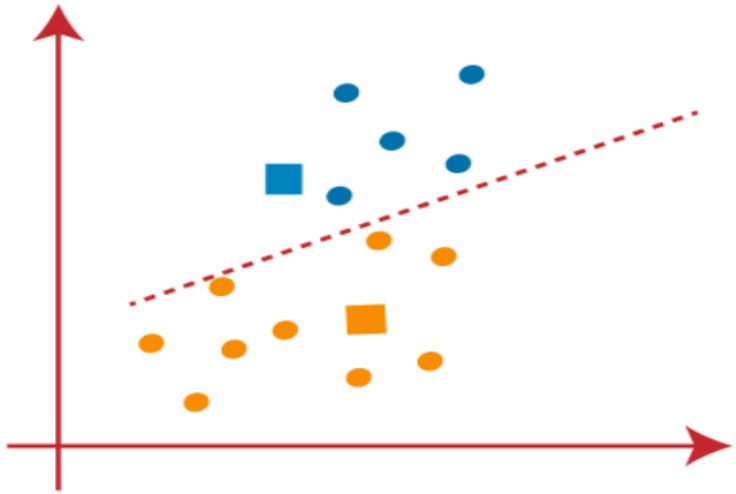
Comme nous devons trouver le cluster le plus proche, nous allons répéter le processus en choisissant un nouveau centroïde.



Ensuite, il va réaffecter chaque point de données au nouveau centroïde.
Pour cela, il va répéter le même processus de recherche d'une ligne médiane.
La médiane sera comme dans l'image ci-dessous:



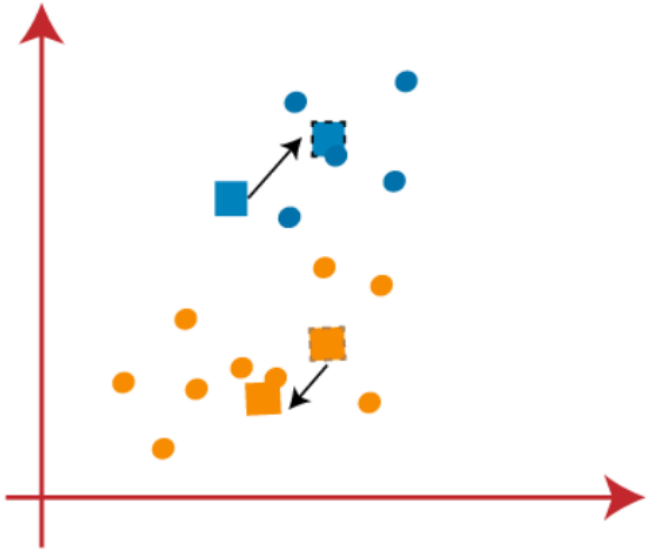
Sur l'image du milieu, nous pouvons voir qu'un point jaune se trouve sur le côté gauche de la ligne et que deux points bleus se trouvent à droite de la ligne.
Ces trois points seront donc attribués à de nouveaux centroïdes.



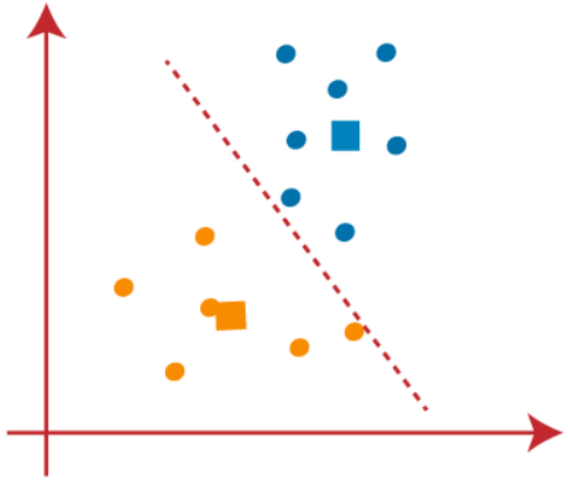
CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

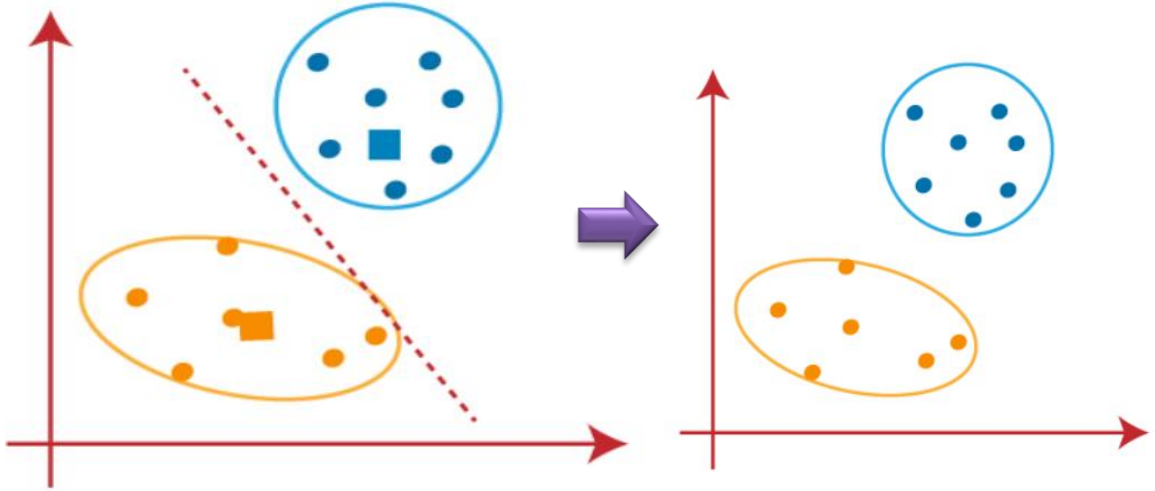
❖ La réaffectation ayant eu lieu, nous allons donc passer à l'étape 4, qui consiste à trouver de nouveaux centroïdes ou points K. Nous allons répéter le processus en trouvant le centre de gravité des centroïdes, de sorte que les nouveaux centroïdes seront tels qu'illustrés dans l'image ci-dessous :



Comme nous avons obtenu les nouveaux centroïdes, nous allons à nouveau tracer la ligne médiane et réaffecter les points de données. L'image sera donc :



Nous pouvons voir dans l'image ci-dessus qu'il n'y a pas de points de données différents de chaque côté de la ligne, ce qui signifie que notre modèle est formé. Considérez l'image ci-dessous.



CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

- Comment choisir la valeur de « nombre K de clusters » ?

❑ Les performances de l'algorithme de clustering K-means dépendent des clusters hautement efficaces qu'il forme.
Mais choisir le nombre optimal de clusters est une tâche difficile.

❑ Il existe différentes manières de trouver le nombre optimal de clusters, mais nous allons ici discuter de la méthode la plus appropriée pour trouver le nombre de clusters ou la valeur de K. La méthode est Elbow Method.

- Elbow Method

❑ La méthode Elbow (méthode du coude) est une technique utilisée en apprentissage non supervisé, notamment dans l'algorithme de k-means clustering, pour déterminer le nombre optimal de clusters k.

❑ Elle vise à trouver le bon compromis entre le nombre de clusters et la variance expliquée (ou compacité des clusters).

❑ Concepts clés :

✓ **Within-Cluster Sum of Squares** (WCSS) [Somme des carrés intra-cluster] : La somme des distances au carré entre chaque point et le centroïde de son cluster. L'objectif est de minimiser cette valeur.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

- Étapes de la méthode Elbow

✓ **Étapes 1: Exécuter k-means pour une plage de valeurs de k** : commencez par un petit nombre de clusters (par exemple,k=1) et augmentez progressivement k (par exemple, jusqu'à k=10).

✓ **Étapes 2: Calculer la somme des carrés intra-cluster WCSS pour chaque k** : Pour chaque nombre de clusters, calculez la somme des carrés intra-cluster. Plus le nombre de clusters augmente, plus WCSS diminue car les points de données sont plus proches de leur centroïde respectif.

❑ La formule du WCSS est : $$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

où C_i représente le i-ème cluster,

x est un point de données, et

μ_i est le centroïde du cluster i.

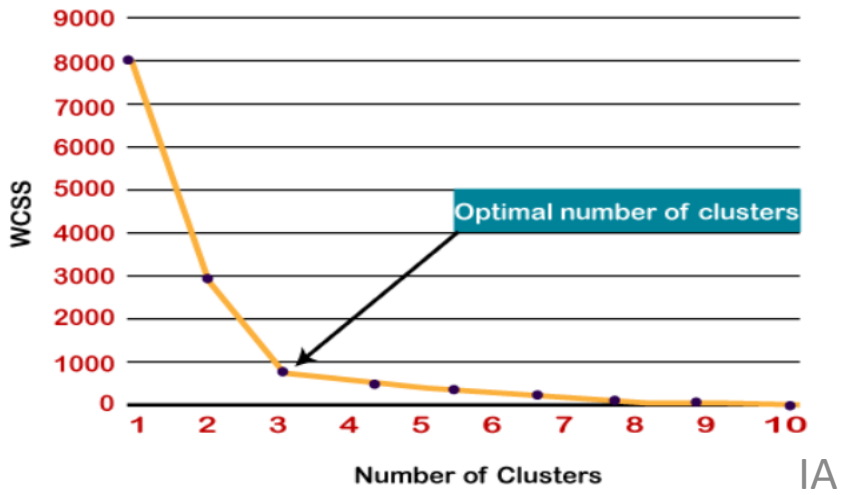
CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.3 Fonctionnement de K-Means Clustering Algorithm

- Étapes de la méthode Elbow

✓ **Étapes 3 : Tracer WCSS contre k** : Créez un graphique en mettant k sur l'axe des abscisses et WCSS sur l'axe des ordonnées.
En général, la somme des carrés diminue rapidement au début, puis la diminution devient plus lente à mesure que k augmente.

✓ **Étapes 4 : Identifier le "coude" (elbow)** : Le point où la diminution de WCSS devient moins significative est appelé le "coude".
C'est l'endroit où l'ajout de clusters supplémentaires n'améliore plus beaucoup la qualité de la partition.
Ce point correspond au nombre optimal de clusters.



CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.4 Implémentation Python de l'algorithme de clustering K-means

1. Comprendre le problème

Nous devons comprendre quel type de problème nous allons résoudre ici. Nous disposons donc d'un ensemble de données (Mall_Customers dataset), qui contient les données des clients qui visitent le centre commercial et y dépensent.

✓ Dans l'ensemble de données, nous avons des attributs suivants :

- Customer_Id [numero du client],
- Gender [Genre],
- Age,
- Annual Income [Revenu annuel] ($) et
- Spending Score (qui est la valeur calculée du montant dépensé par un client dans le centre commercial, plus la valeur est élevée, plus il a dépensé).

✓ Il s'agit d'une méthode non supervisée, nous ne savons donc pas exactement quelles sont les catégories.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.4 Implémentation Python de l'algorithme de clustering K-means

2. segmentation de la clientèle avec l'algorithme de clustering K-means

Étape 1 : Importer les packages nécessaires et de la dataset

```
1 # importation de bibliothèques
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
```

```
1 # Importer la dataset
2 dataset = pd.read_csv('Mall_Customers.csv')
3 dataset
```

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Code Python où on utilises les bibliothèques importées pour appliquer la méthode Elbow avec l'algorithme k-means clustering.

- ✓ **pd.read_csv('Mall_Customers.csv')** indique à pandas de lire le fichier 'Mall_Customers.csv' qui se trouve dans le même répertoire que le script Python.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.4 Implémentation Python de l'algorithme de clustering K-means

2. segmentation de la clientèle avec l'algorithme de clustering K-means

Étape 2 : Étape de prétraitement des données

```
1 dataset['Genre'] = np.where(dataset['Genre'] == 'Male', 1, 0)
```

❖ Le résultat de np.where() remplace la colonne Genre dans le DataFrame.
Ainsi, la colonne Genre qui contenait des chaînes de caractères ('Male', 'Female')
est maintenant transformée en valeurs numériques :1 pour 'Male' , 0 pour 'Female'.

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

```
1 # Extraction de variables indépendantes
2
3 x = dataset.iloc[:, 3:5].values
```

❖ L'instruction **x = dataset.iloc[:, 3:5].values** va créer un tableau NumPy avec les valeurs des colonnes "Annual Income" et "Spending Score" comme ceci :

```
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
```

✓ **.values** : Cette méthode extrait les valeurs du DataFrame sous forme de tableau NumPy. Cela est pratique pour utiliser les données avec des algorithmes de machine learning.

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

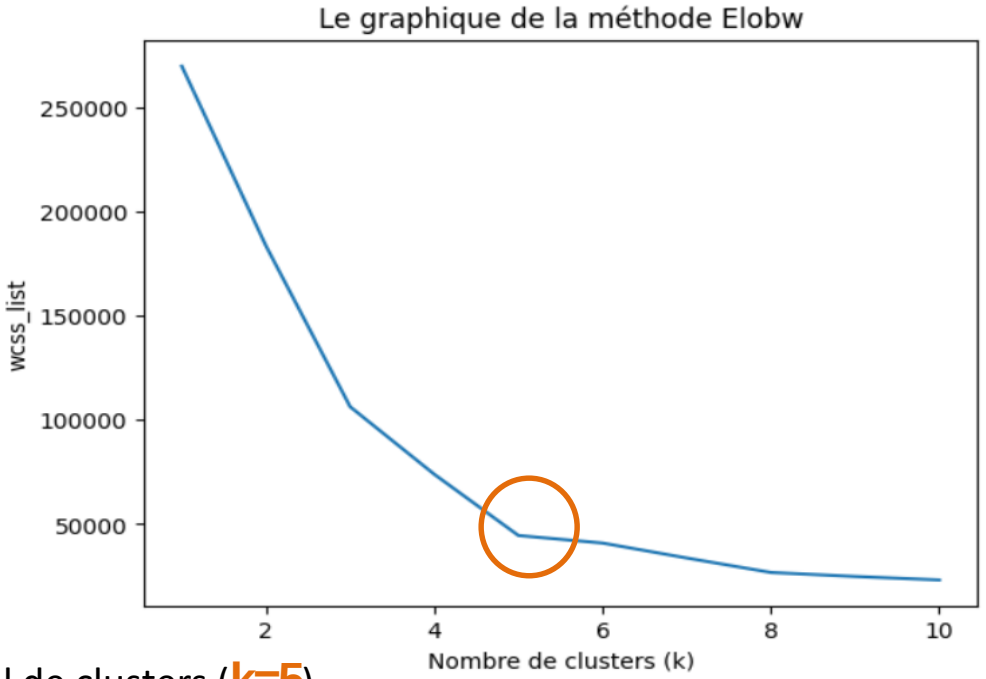
3.3.4 Implémentation Python de l'algorithme de clustering K-means

2. segmentation de la clientèle avec l'algorithme de clustering K-means

Étape 3 : Trouver le nombre optimal de clusters à l'aide de la méthode du coude

```python
#trouver le nombre optimal de clusters en utilisant la méthode du coude
from sklearn.cluster import KMeans
wcss_list= []  #Initialisation de la liste des valeurs de WCSS

#Utilisation de la boucle for pour les itérations de 1 à 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title('Le graphique de la méthode Elobw')
plt.xlabel('Nombre de clusters (k)')
plt.ylabel('wcss_list')
plt.show()
```



- **Identification du coude (elbow)** : Ce point correspond au nombre optimal de clusters (**k=5**).

CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.4 Implémentation Python de l'algorithme de clustering K-means

2. segmentation de la clientèle avec l'algorithme de clustering K-means

Étape 4 : Entraînement de l'algorithme K-means sur les données d'entraînement

```
#Entraînement du modèle K-means sur un ensemble de données
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)

# Prediction de cluster
y_predict= kmeans.fit_predict(x)
```

- ✓ **KMeans(n_clusters=5)** : Crée un objet de l'algorithme K-means avec 5 clusters.
- ✓ **init='k-means++'** : k-means++ est une méthode d'initialisation améliorée pour choisir les centroïdes initiaux.

- ➢ **fit_predict()** combine deux étapes :
- ✓ **fit** : Entraîne le modèle k-means sur les données x.
- ✓ **predict** : Après avoir trouvé les clusters, il attribue chaque point de données à un cluster.

```
y_predict
```

```
array([4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,
       4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 0,
       4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 1, 3, 1, 3, 1,
       0, 1, 3, 1, 3, 1, 3, 1, 3, 1, 0, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1,
       3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1,
       3, 1], dtype=int32)
```

- ✓ **y_predict** : Cette variable contient les étiquettes de cluster pour chaque point de données.

C'est un tableau 1D où chaque élément est un entier représentant le numéro du cluster auquel un point a été attribué (de 0 à 4, car il y a 5 clusters).

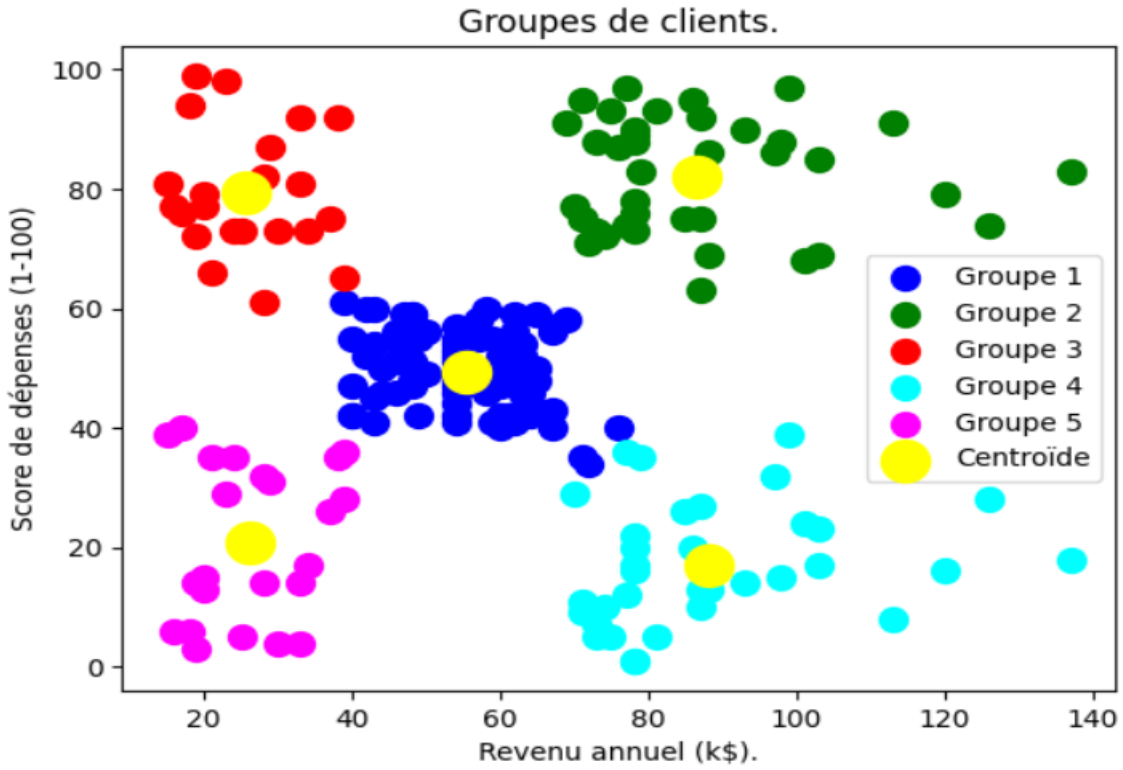
CHAPITRE 3 Techniques d'IA utilisées dans le réseaux informatique

3.3.4 Implémentation Python de l'algorithme de clustering K-means

2. segmentation de la clientèle avec l'algorithme de clustering K-means

Étape 5 : Visualisation des clusters

```
#Visualisation des clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Groupe 1') # Pour le premier groupe.
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Groupe 2') # Pour le deuxième groupe.
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Groupe 3') # Pour le troisième groupe.
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Groupe 4') # Pour le quatrième groupe.
plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Groupe 5') # Pour le cinquième groupe
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroïde')
plt.title('Groupes de clients.')
plt.xlabel('Revenu annuel (k$).')
plt.ylabel('Score de dépenses (1-100)')
plt.legend()
plt.show()
```



TD 4