Intelligence-Artificielle

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

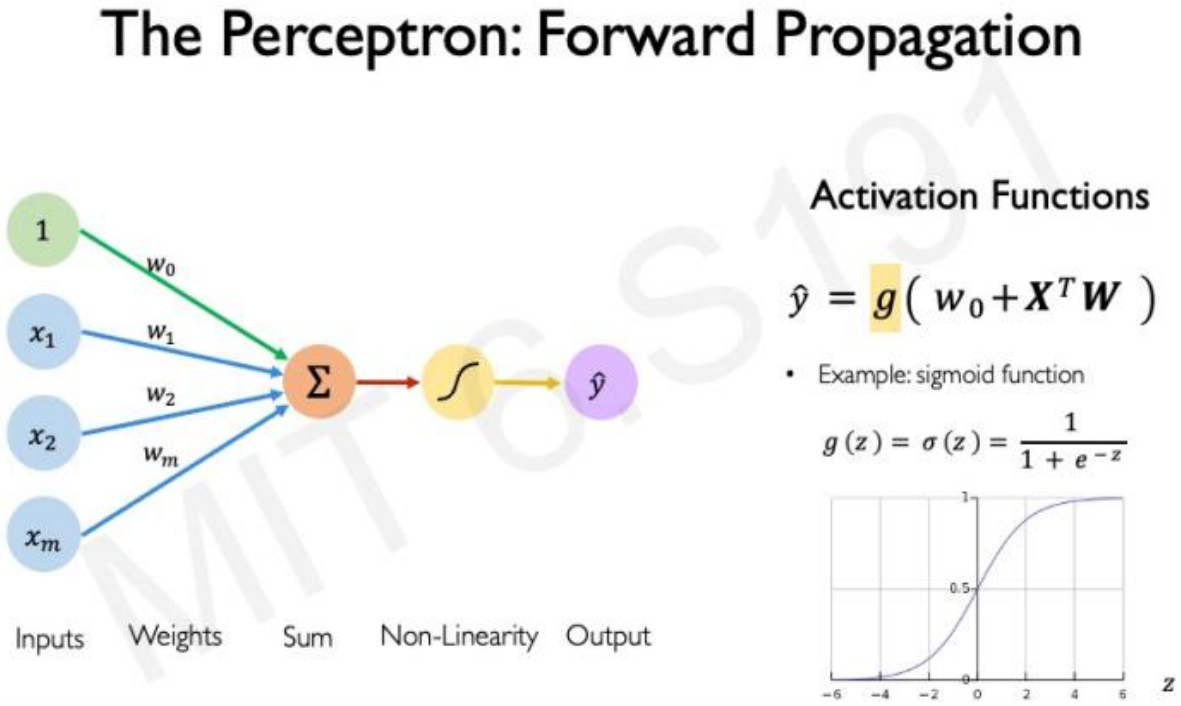
- Assisté par Ass. **Jason MUSA**

Mardi : 8H00 – 12H00

Mercredi : 13H00 – 17H00

Vendredi : 13H00 – 13H00

- E-mail : dahouda37@gmail.com
- Tel.: **+243 99 66 55 265**



CHAPITRE 2 MACHINE LEARNING

2.5. Classification

2.5.1. 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.

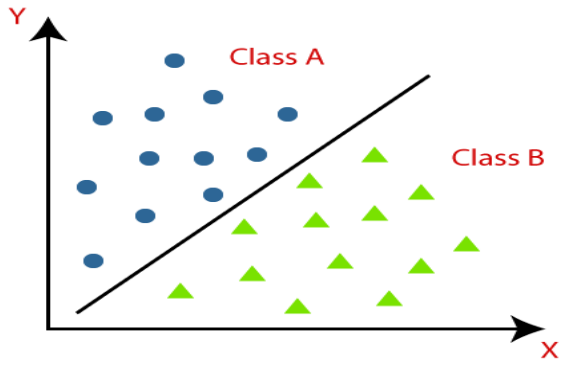
❑ Elle est couramment utilisée dans des tâches telles que la détection de spam, les diagnostics médicaux, l'analyse des sentiments et la détection de fraude, où le résultat est soit « oui » soit « non », « positif » soit « négatif »,

❑ 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 predite

CHAPITRE 2 MACHINE LEARNING

2.5. Classification

2.5.1. 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.

✓ **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 2 MACHINE LEARNING

2.5. Classification

2.5.2. 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 2 MACHINE LEARNING

2.5. Classification

2.5.2. Métriques de classification dans le Machine Learning

1. Matrice de Confusion [Confusion Matrix]

➢ 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)

✓ **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 2 MACHINE LEARNING

2.5. Classification

2.5.2. 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 2 MACHINE LEARNING

2.5. Classification

2.5.2. Métriques de classification dans le Machine Learning

2. Métriques dérivées de la matrice de confusion

✓ **Exactitude [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 2 MACHINE LEARNING

2.5. Classification

2.5.2. 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 :

$$Precision = \frac{80}{80 + 15} = \frac{80}{95} = 0.842$$

CHAPITRE 2 MACHINE LEARNING

2.5. Classification

2.5.2. 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 :

$$Recall = \frac{80}{80 + 10} = \frac{80}{90} = 0.889$$

CHAPITRE 2 MACHINE LEARNING

2.5. Classification

2.5.2. 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 :

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

CHAPITRE 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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.

- 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 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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')
```

	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 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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()
```

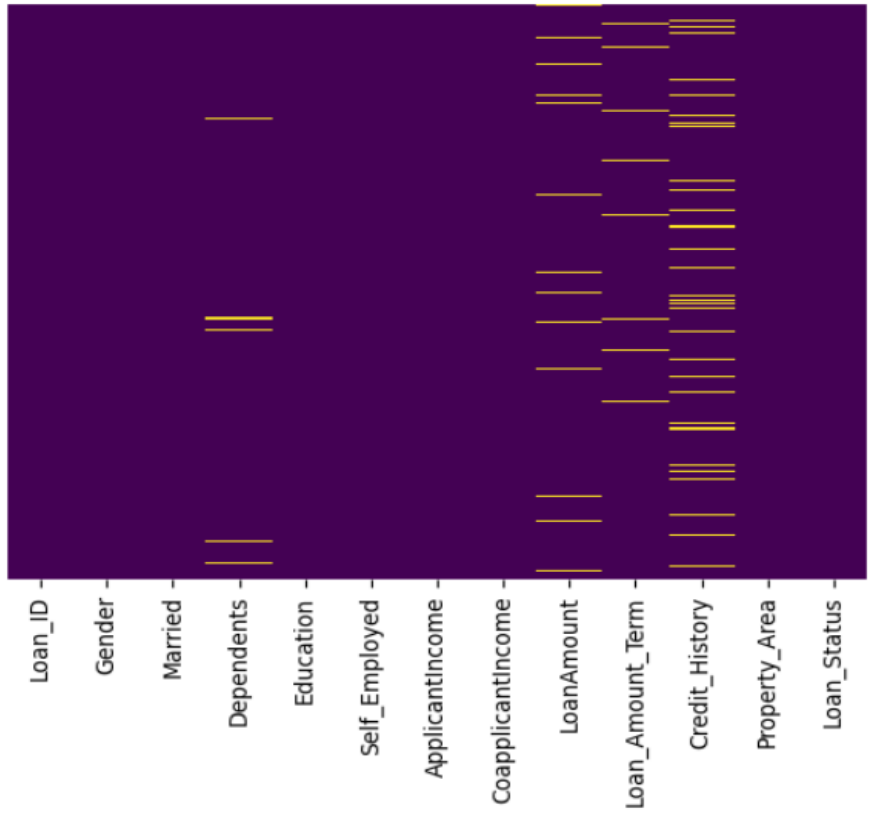
```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

```
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 2 MACHINE LEARNING

2.5 .Classification

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

Étape 2 : Prétraitement des données

```
df.info()
```

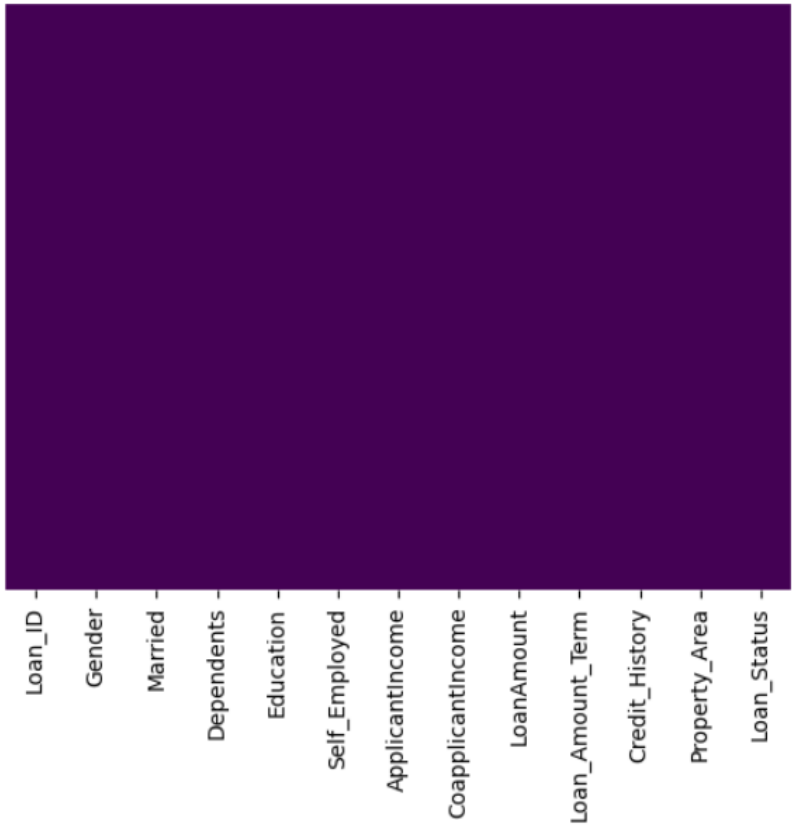
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 598 entries, 0 to 597
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            598 non-null    object
 1   Gender             598 non-null    object
 2   Married            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 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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)
```

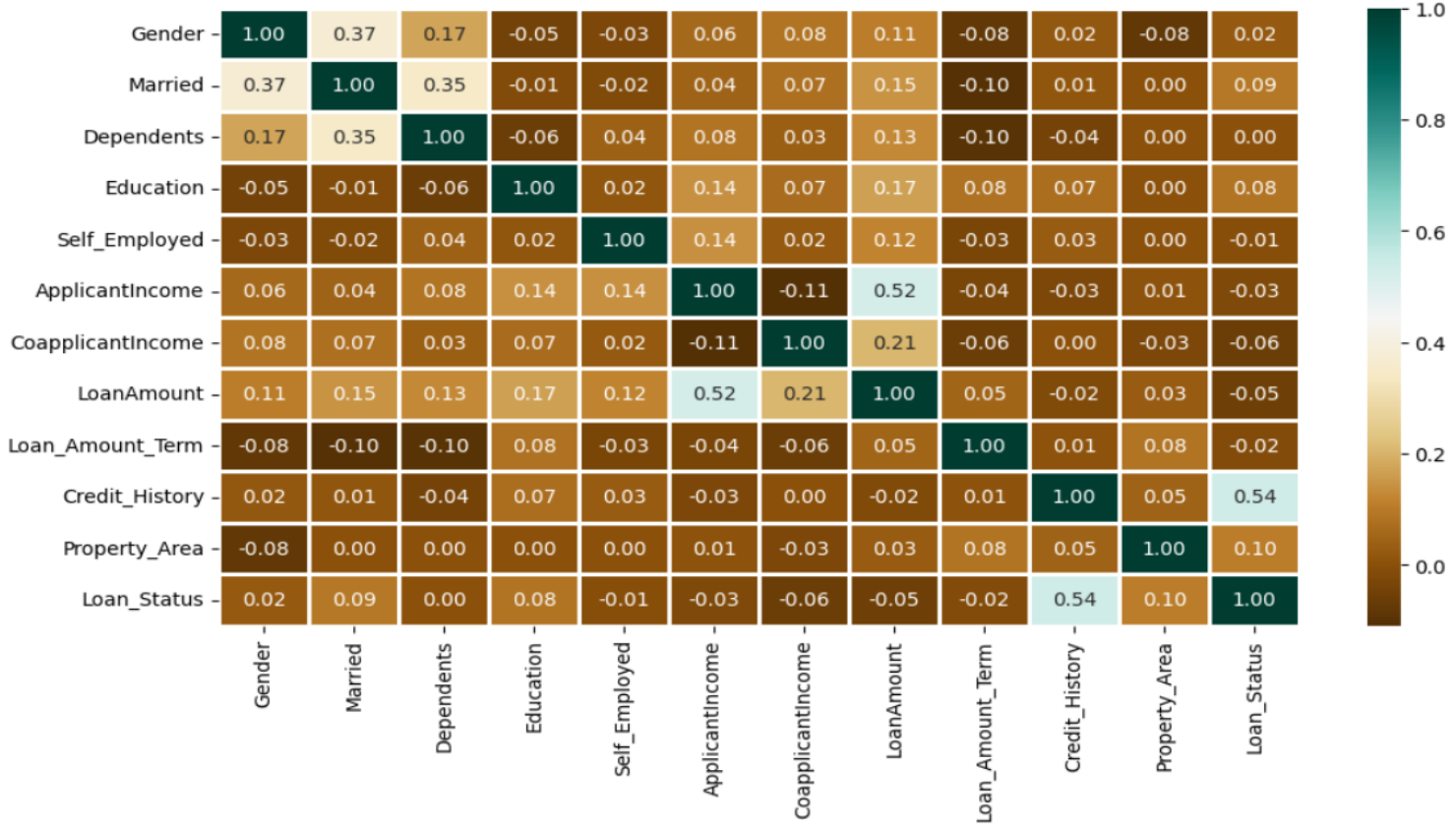
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount
0	1	0	0.0	1	0	5849	0.0	144.968804
1	1	1	1.0	1	0	4583	1508.0	128.000000
2	1	1	0.0	1	1	3000	0.0	66.000000
3	1	1	0.0	0	0	2583	2358.0	120.000000
4	1	0	0.0	1	0	6000	0.0	141.000000

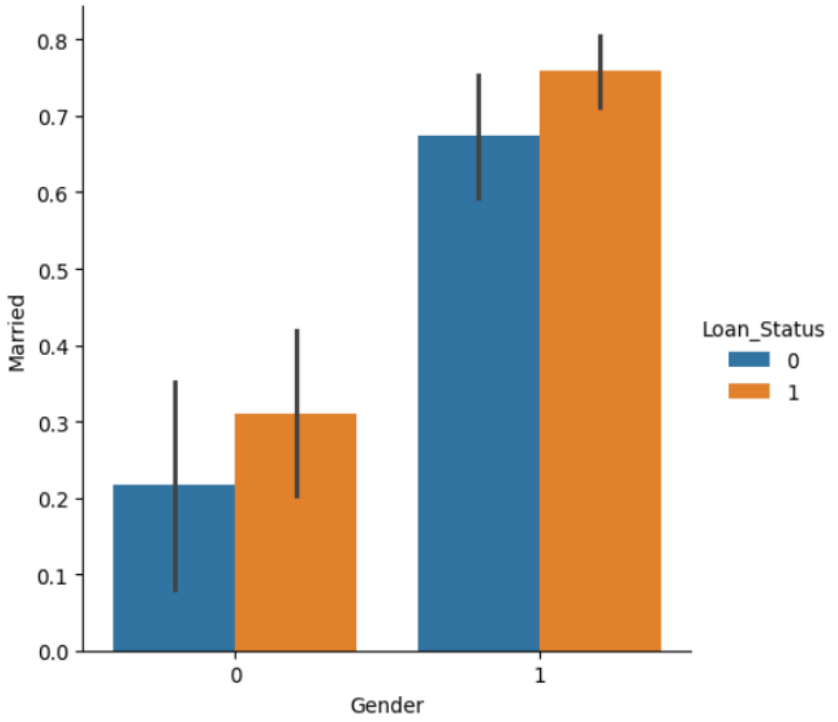
CHAPITRE 2 MACHINE LEARNING

2.5 .Classification

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

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





CHAPITRE 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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 2 MACHINE LEARNING

2.5 .Classification

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

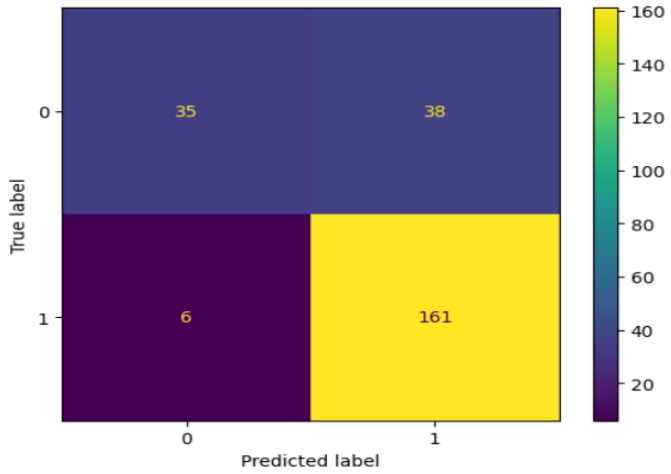
Étape 6 : Évaluation : Matrice de Confusion [Confusion Matrix]

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)
```

```
[[ 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:

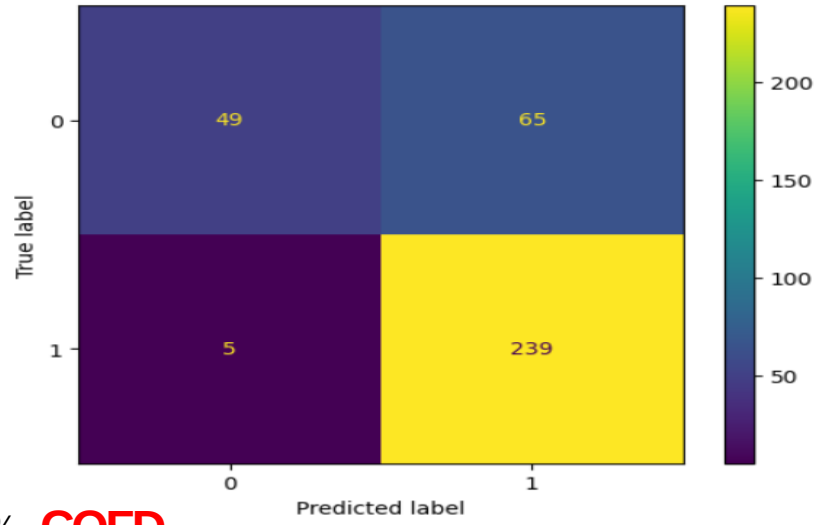
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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 2 MACHINE LEARNING

2.5 .Classification

2.5.3. 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 2 MACHINE LEARNING

2.5 .Classification

2.5.4. 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 2 MACHINE LEARNING

2.5 .Classification

2.5.4. Comparaison de performance 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 2 MACHINE LEARNING

2.5 .Classification

2.5.4. Comparaison de performance 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 2 MACHINE LEARNING

2.6. Apprentissage non supervisé : Clustering

2.6.1 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.

2.6.2 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.

❖ 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.

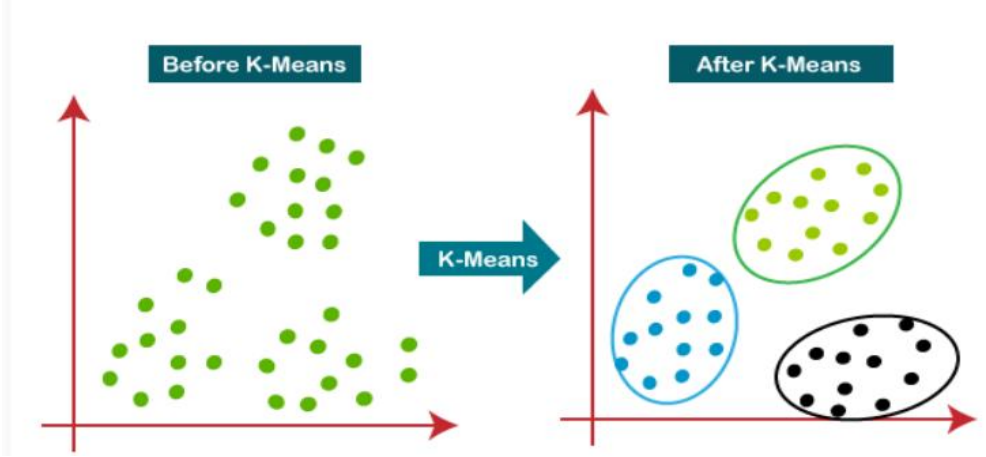
CHAPITRE 2 MACHINE LEARNING

2.6.3 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.

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



CHAPITRE 2 MACHINE LEARNING

2.6.3 Fonctionnement de K-Means Clustering Algorithm

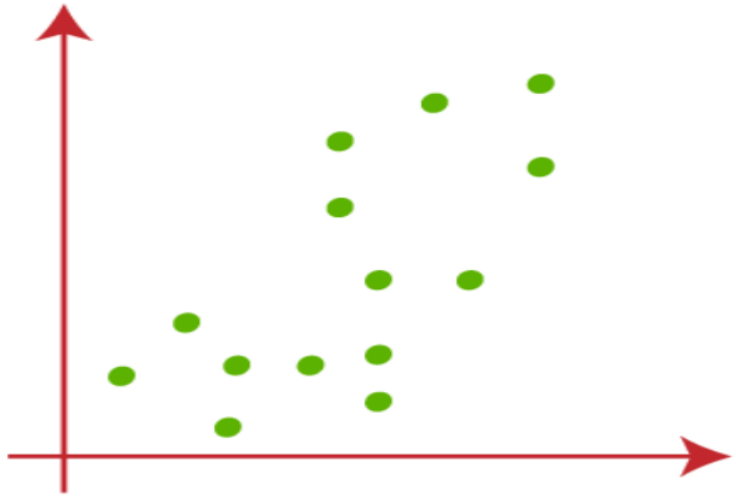
Le fonctionnement de l'algorithme K-Means est expliqué dans les étapes ci-dessous :

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 2 MACHINE LEARNING

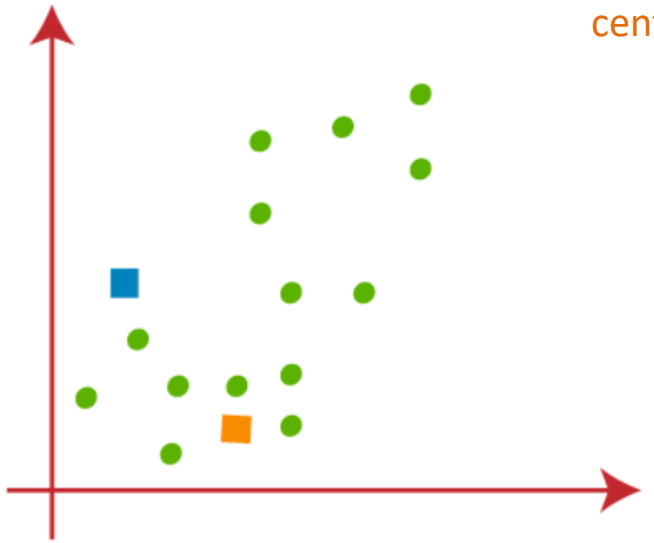
2.6.3 Fonctionnement de K-Means Clustering Algorithm

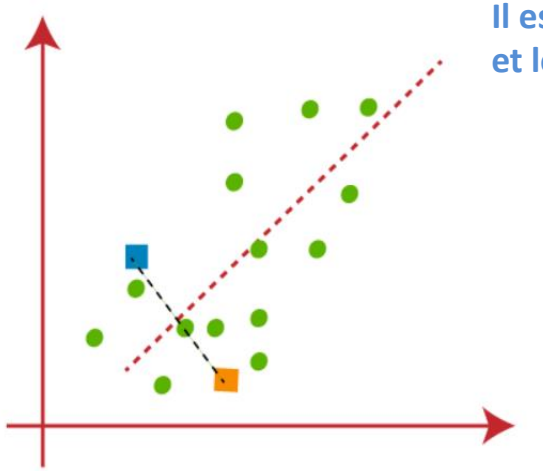
❖ Prenons un nombre k de clusters, c'est-à-dire K=2, pour identifier l'ensemble de données et les placer dans différents clusters.
Cela signifie qu'ici nous allons essayer de regrouper ces ensembles de données en deux clusters différents.

❖ 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.

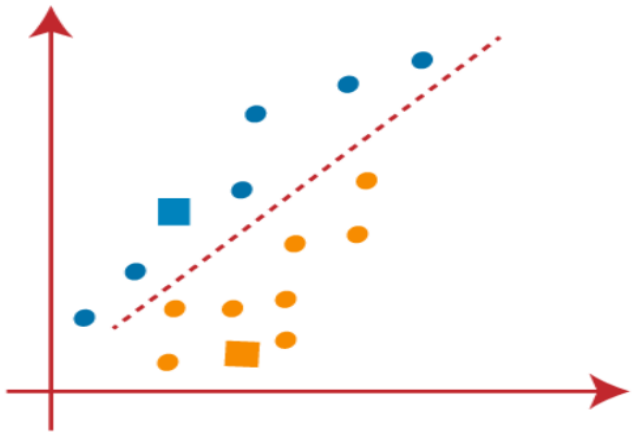
❖ Nous sélectionnons donc ici les deux points ci-dessous comme points k, qui ne font pas partie de notre ensemble de données.

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





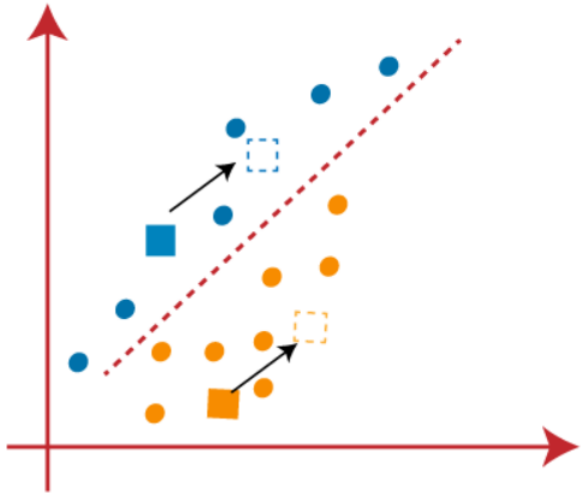
Il est clair que les points à gauche de la ligne sont proches du centroïde K1 ou bleu, et les points à droite de la ligne sont proches du centroïde jaune.



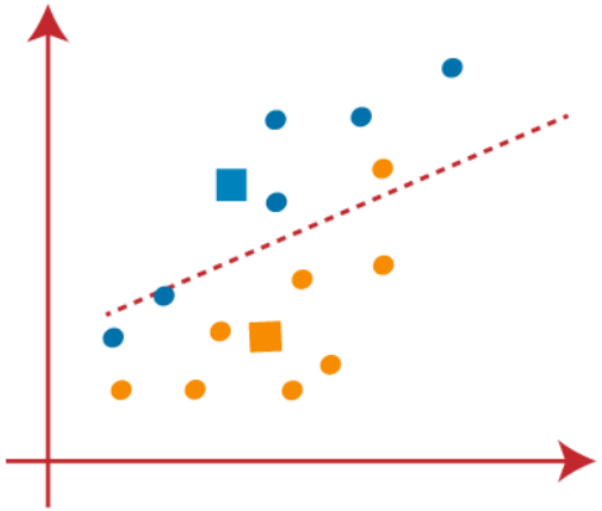
CHAPITRE 2 MACHINE LEARNING

2.6.3 Fonctionnement de K-Means Clustering Algorithm

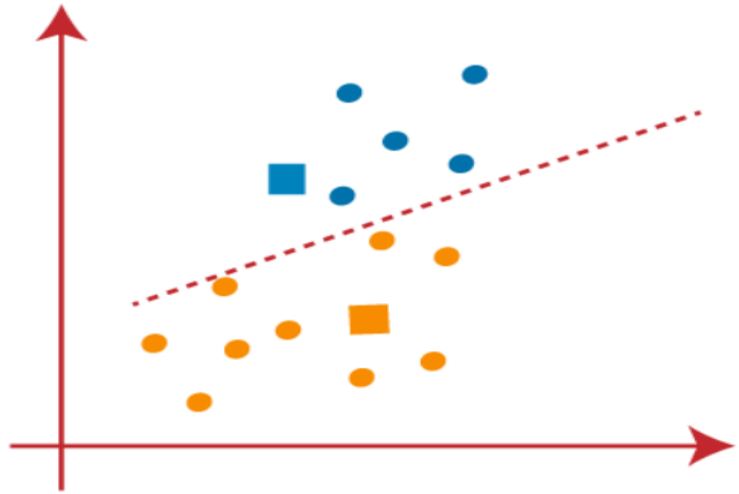
Comme nous devons trouver le cluster le plus proche, nous allons répéter le processus en choisissant un nouveau centroïde.



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:



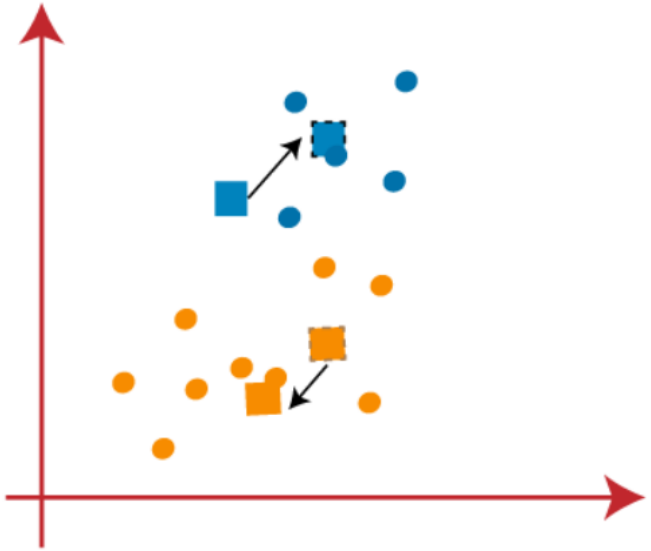
Sur l'image du milieu, nous pouvons voir qu'un point jaune se trouve sur le côté gauche de la ligne et que deux points bleus se trouvent à droite de la ligne.
Ces trois points seront donc attribués à de nouveaux centroïdes.



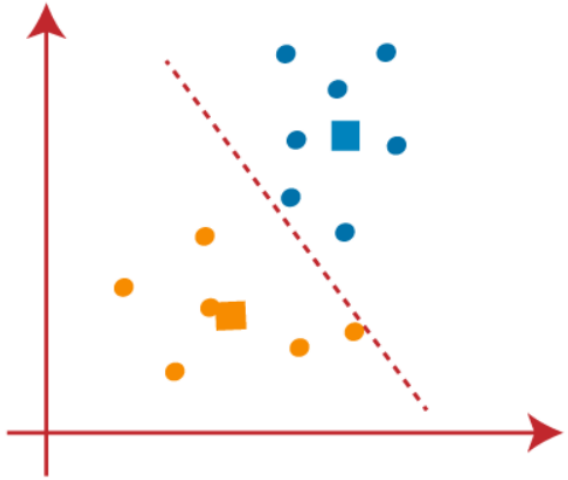
CHAPITRE 2 MACHINE LEARNING

2.6.3 Fonctionnement de K-Means Clustering Algorithm

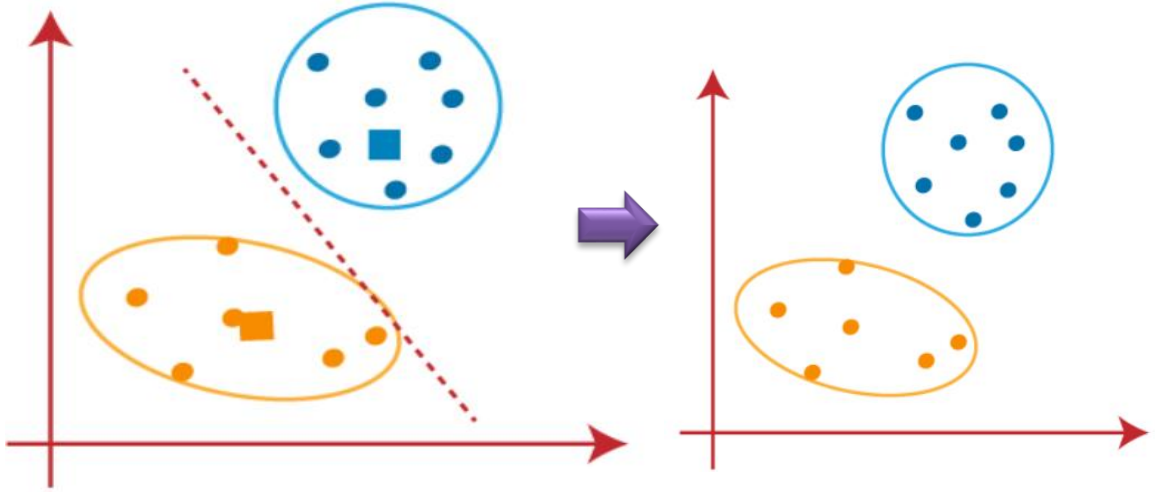
❖ La réaffectation ayant eu lieu, nous allons donc passer à l'étape 4, qui consiste à trouver de nouveaux centroïdes ou points K. Nous allons répéter le processus en trouvant le centre de gravité des centroïdes, de sorte que les nouveaux centroïdes seront tels qu'illustrés dans l'image ci-dessous :



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 2 MACHINE LEARNING

2.6.3 Fonctionnement de K-Means Clustering Algorithm

- 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 2 MACHINE LEARNING

2.6.3 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.

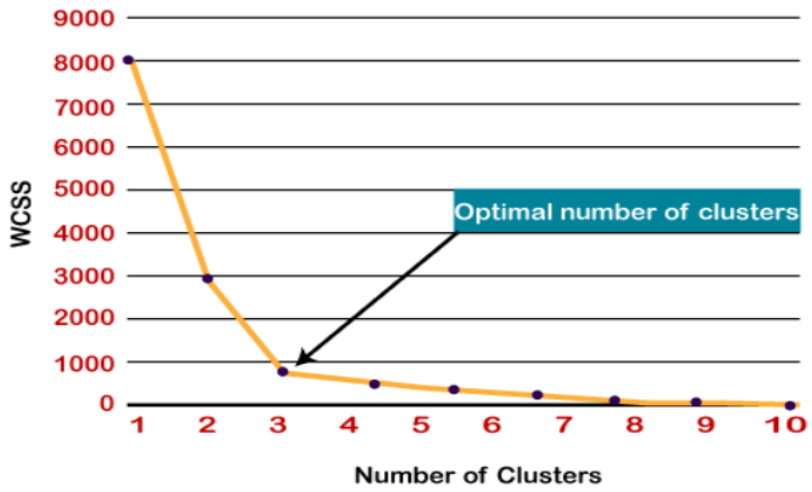
CHAPITRE 2 MACHINE LEARNING

2.6.3 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 2 MACHINE LEARNING

2.6. Introduction à l'apprentissage automatique sur AWS [Lab]

Fin du Chapitre 3 [Lab]