Matière : **Machine Learning**

DSI 3

Chapitre III : La classification

Enseignantes :

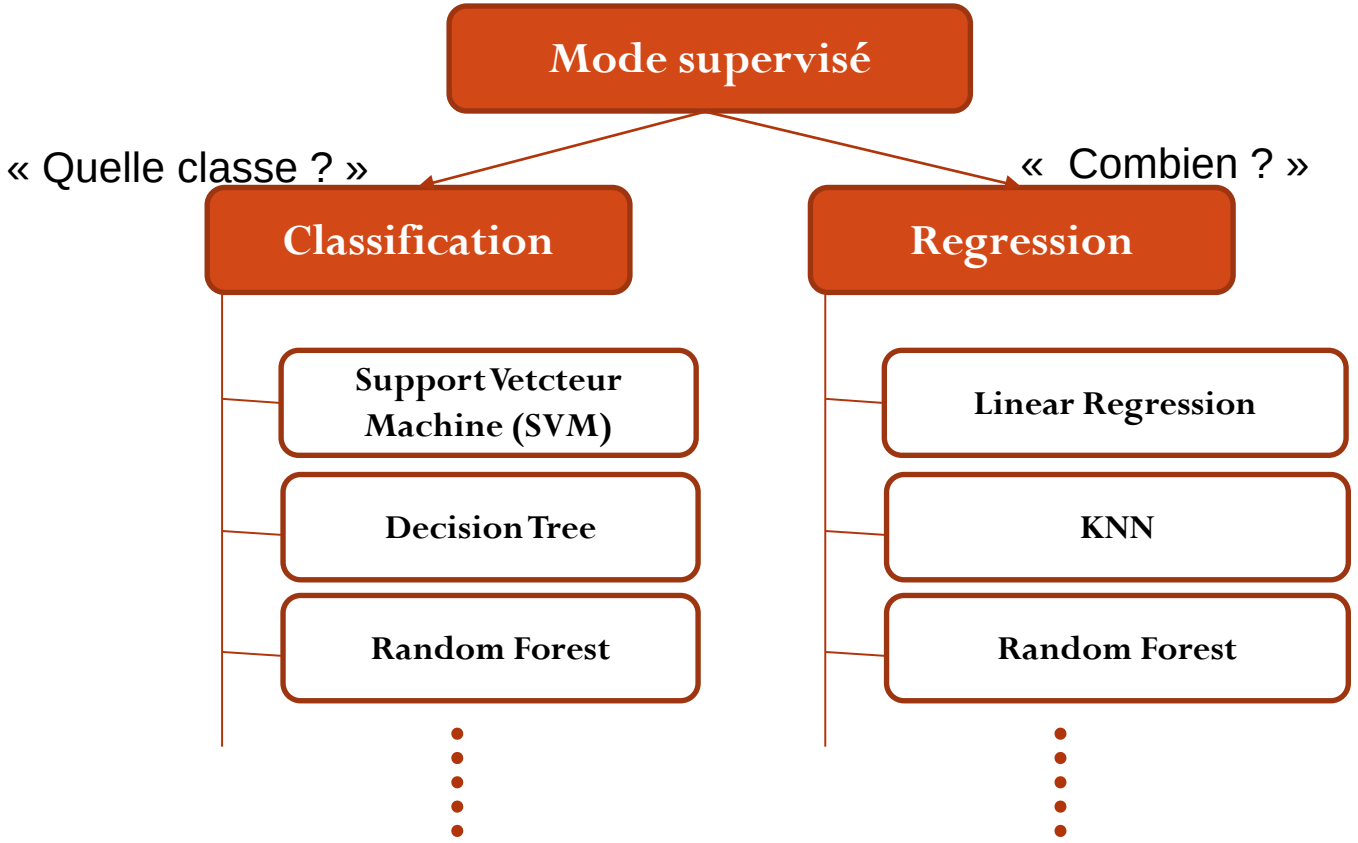
Naïma Halouani
Hounaïda Moalla
ISET Sfax

Plan

- Introduction
- La classification
- La modélisation prédictive
  - KNN
  - Decision Tree
  - Random Forest
  - SVM
  - Logistic Regression

Introduction

Types d'algorithmes



La classification

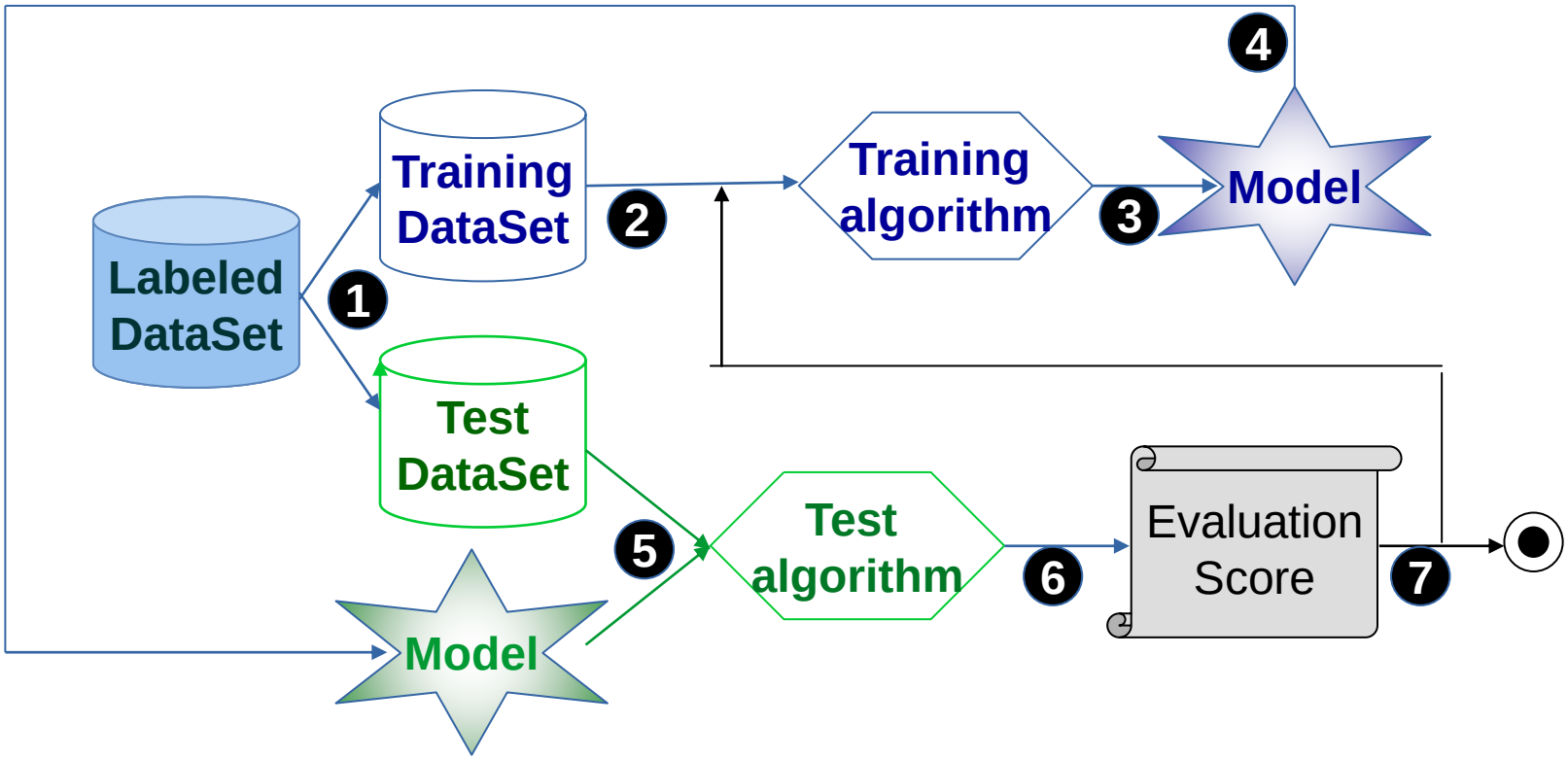
Processus d'apprentissage supervisé

Un algorithme d'apprentissage reçoit un ensemble de données étiquetées sur lequel il va pouvoir s'entraîner et définir un modèle de prédiction.

Ce modèle pourra par la suite être utilisé sur de nouvelles données afin de prédire leurs valeurs de sorties correspondantes.

La classification

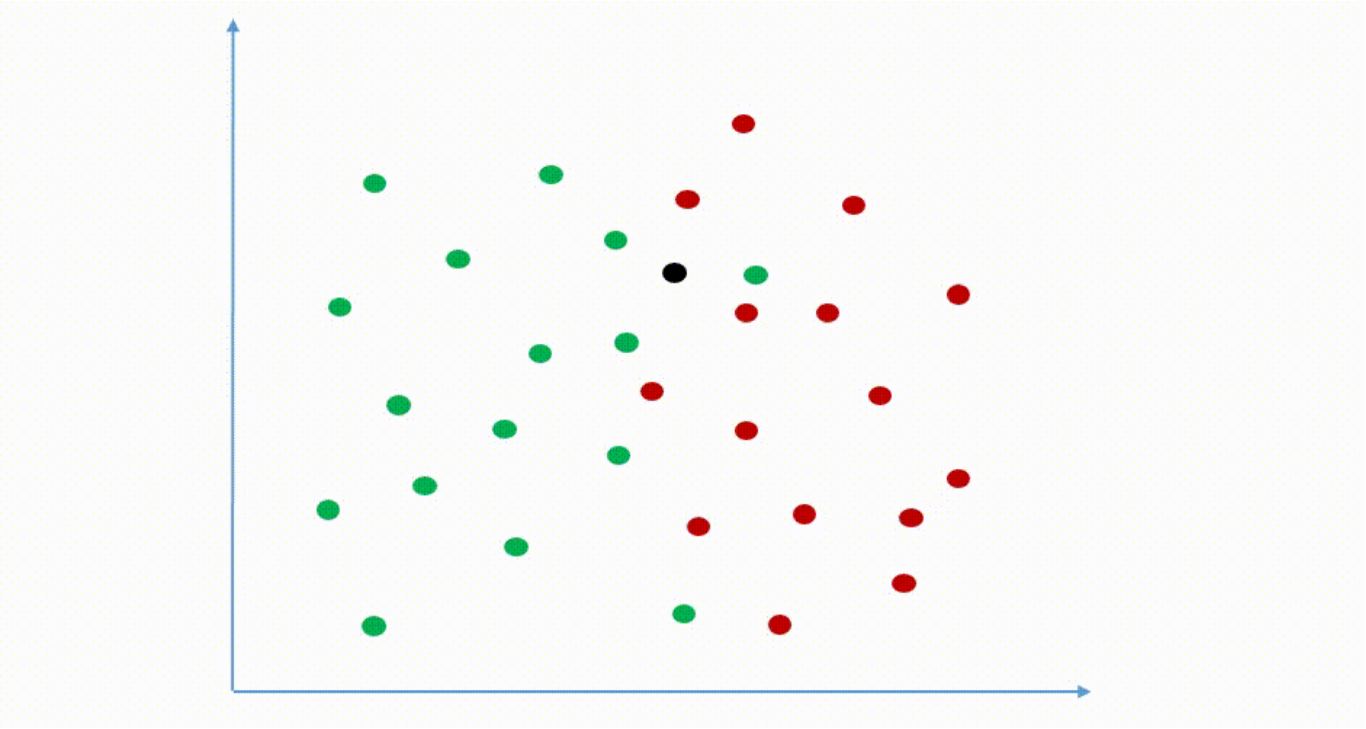
Processus d'apprentissage supervisé



Modélisation Prédictive

K-Nearest Neighbors (KNN)

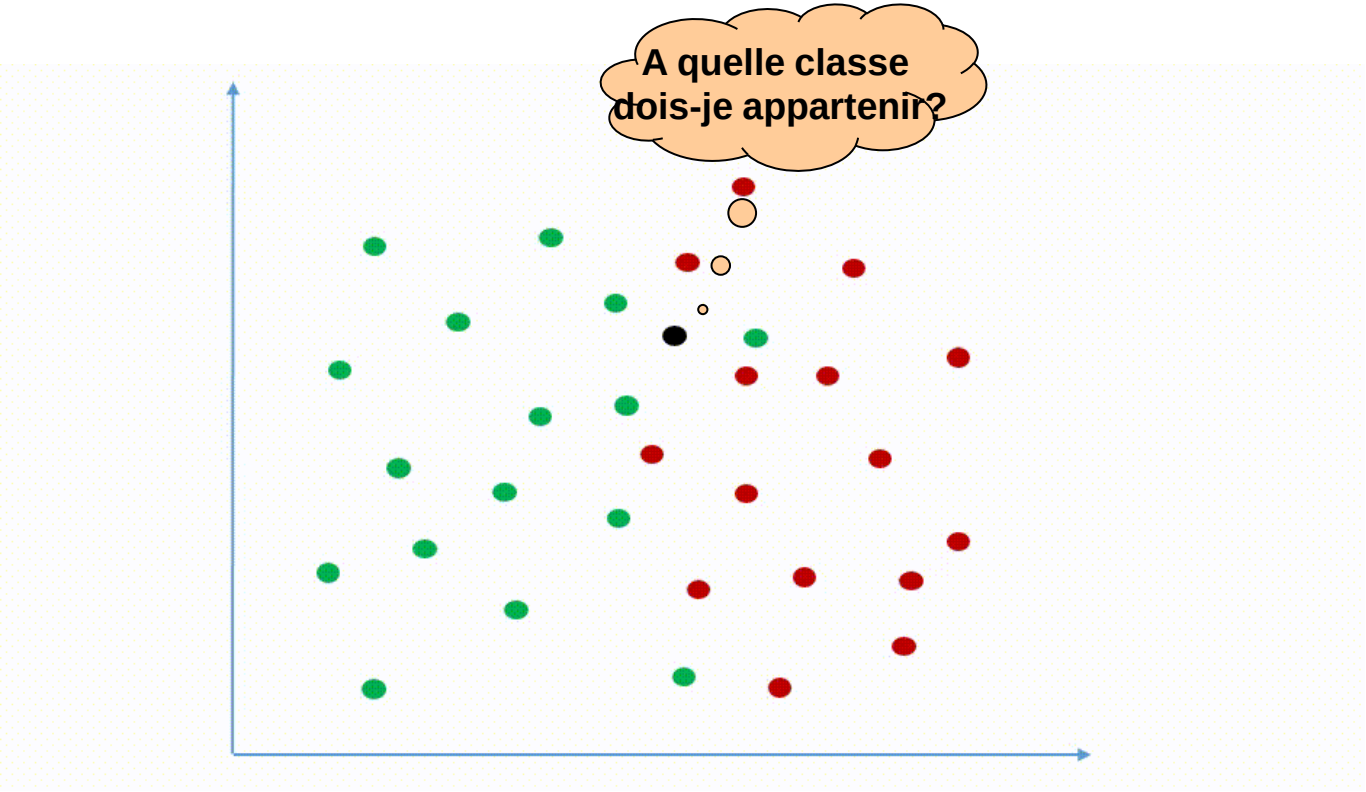
Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Modélisation Prédictive

K-Nearest Neighbors (KNN)

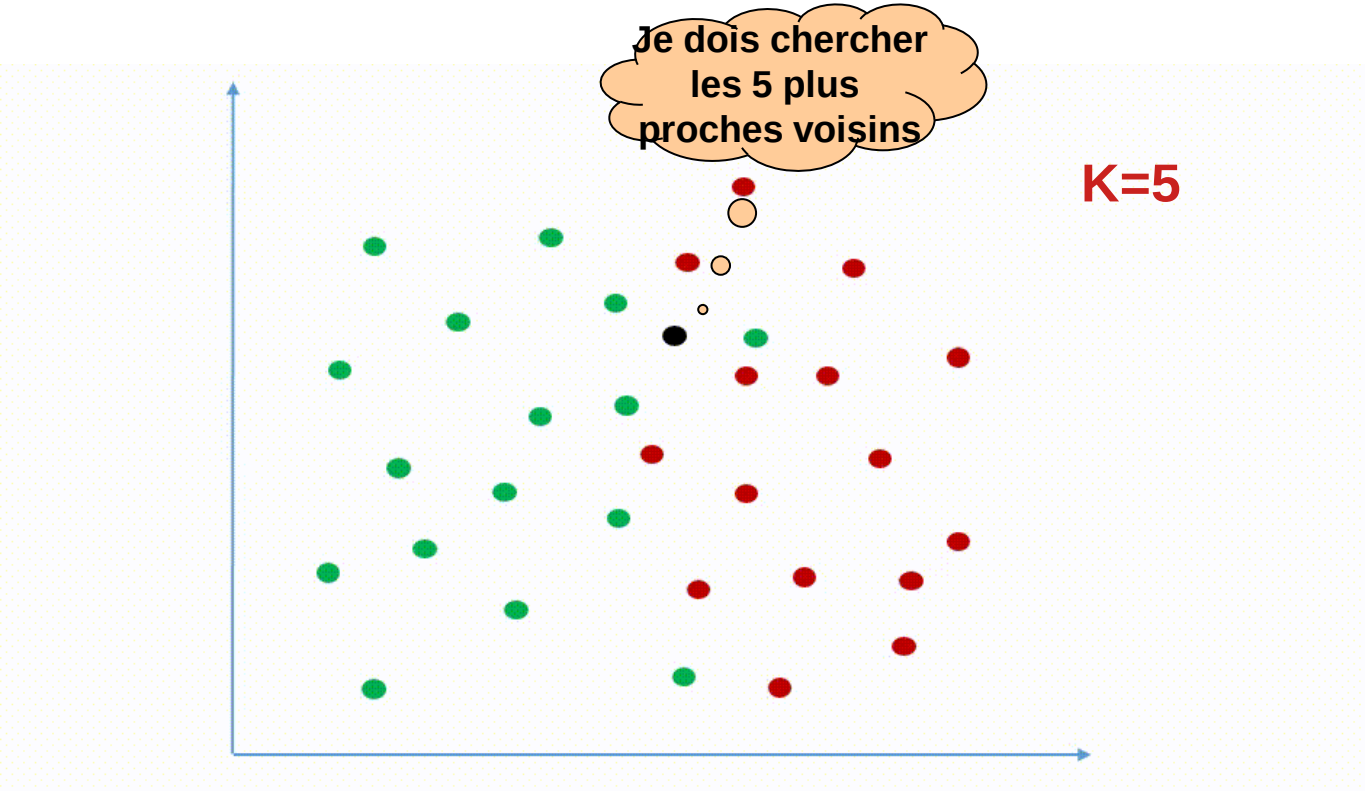
Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Modélisation Prédictive

K-Nearest Neighbors (KNN)

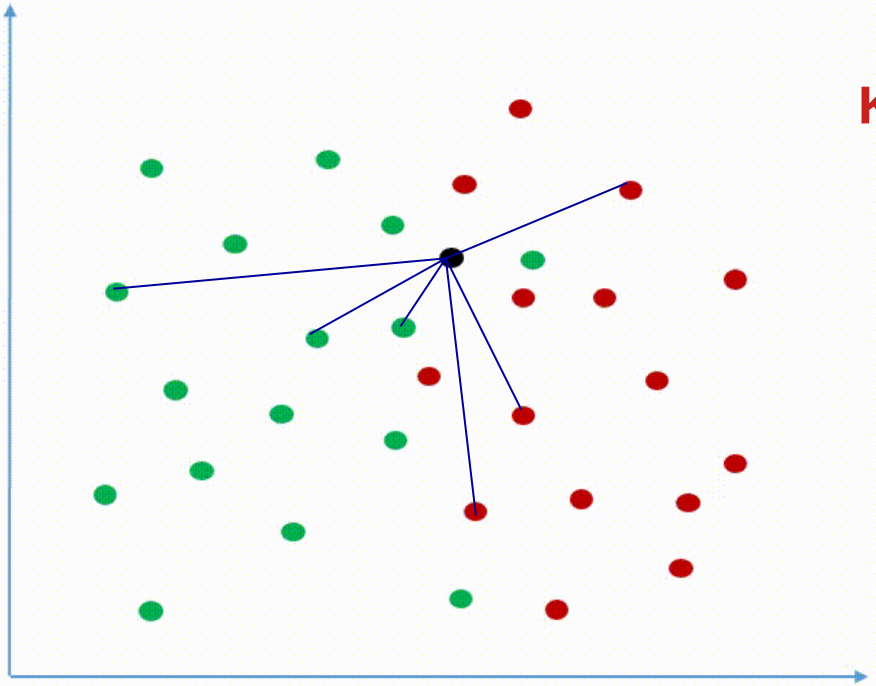
Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Modélisation Prédictive

K-Nearest Neighbors (KNN)

Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Distance Euclidienne :

$$\sum_{i=1}^{n} |x_i - y_i|$$

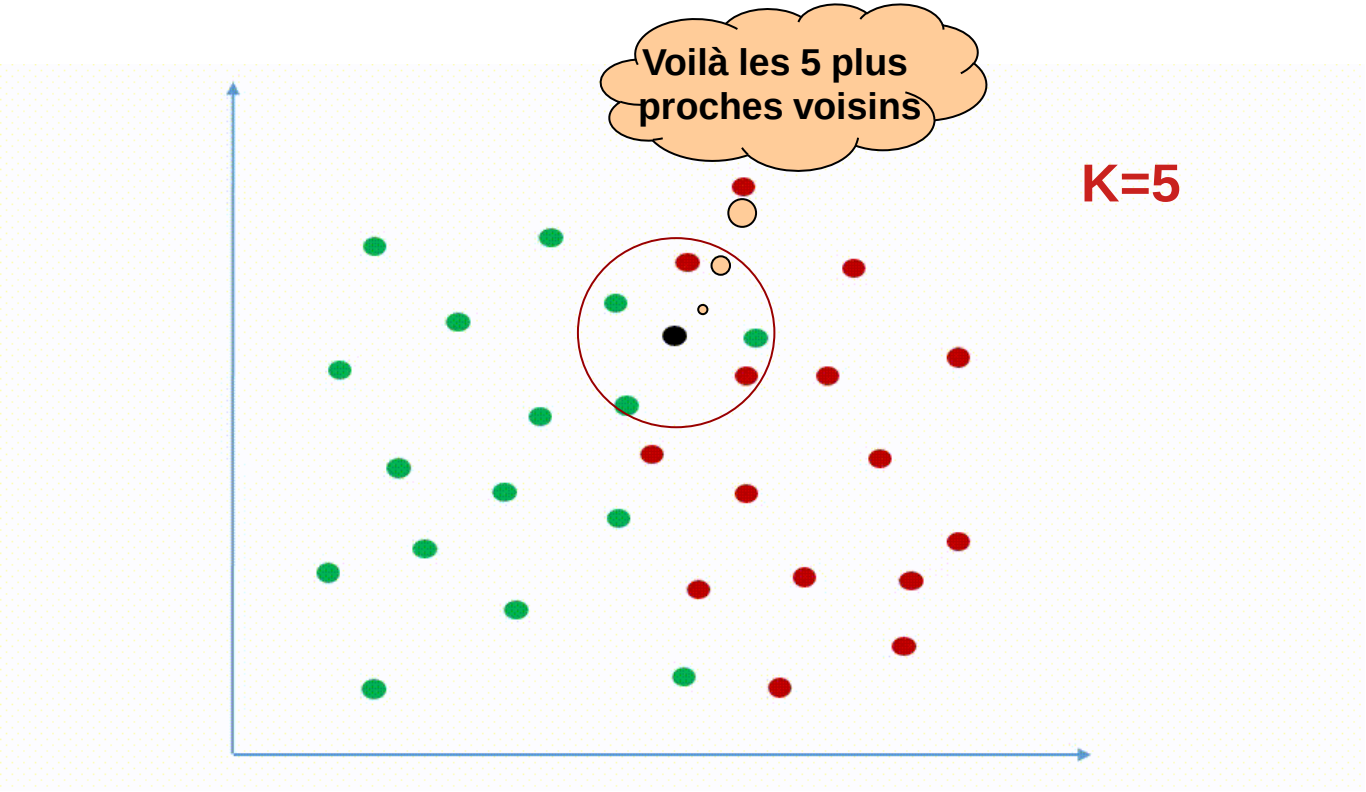
Distance Manhattan :

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Modélisation Prédictive

K-Nearest Neighbors (KNN)

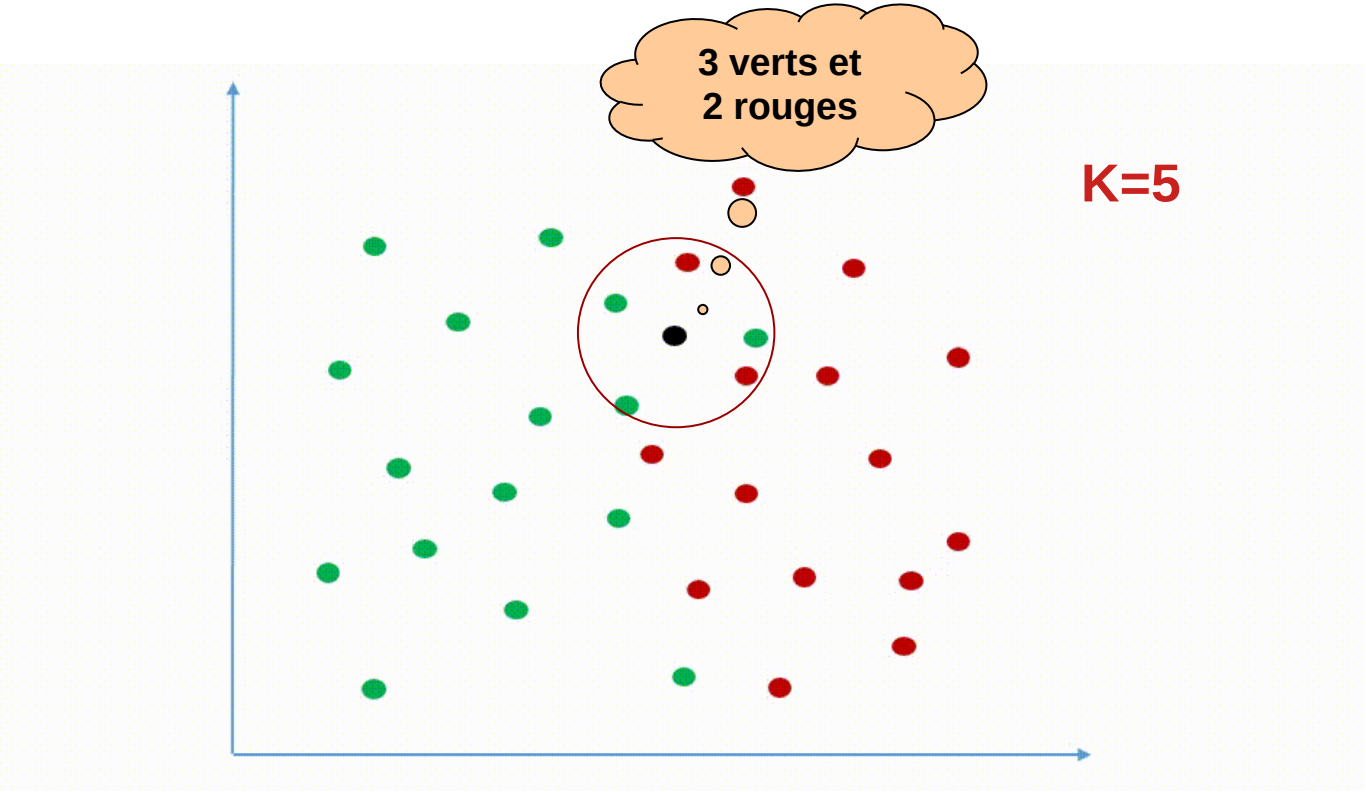
Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Modélisation Prédictive

K-Nearest Neighbors (KNN)

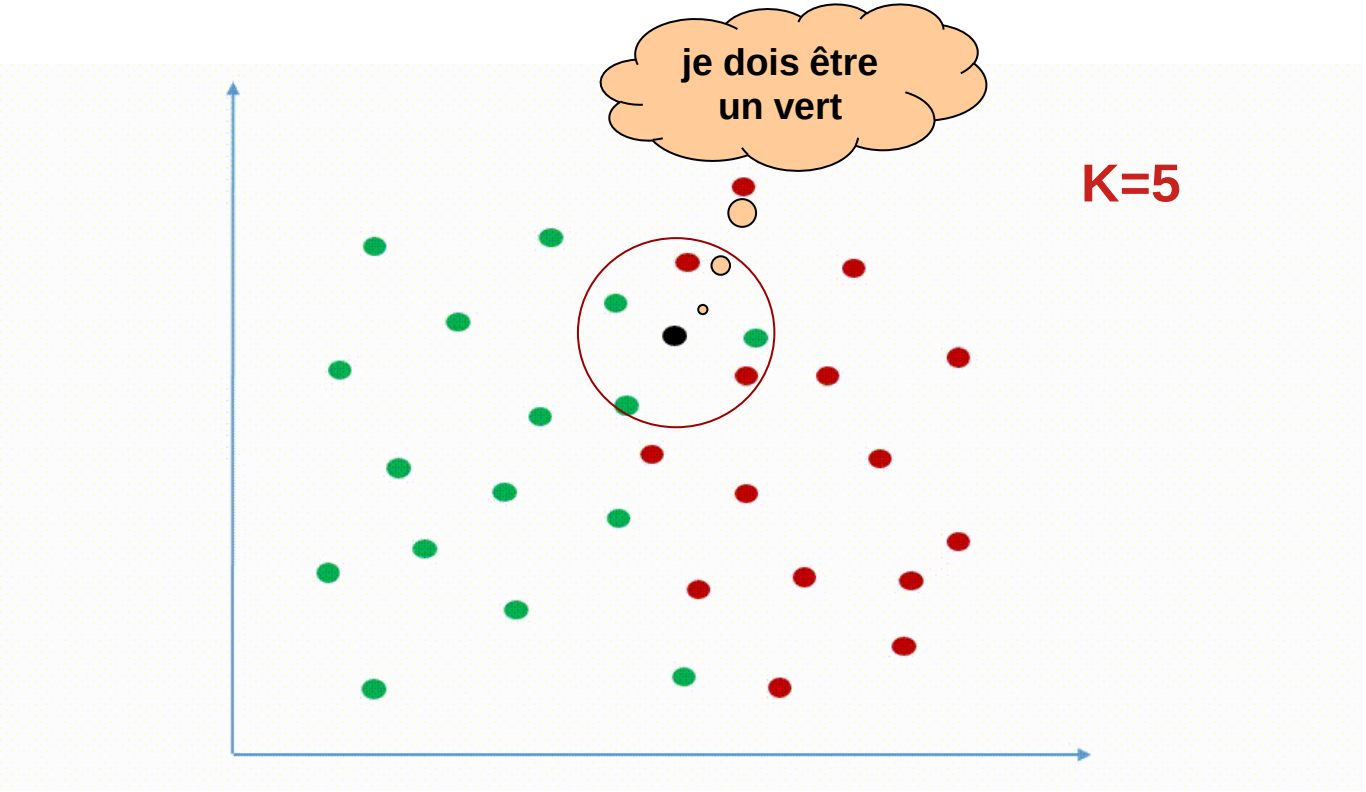
Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Modélisation Prédictive

K-Nearest Neighbors (KNN)

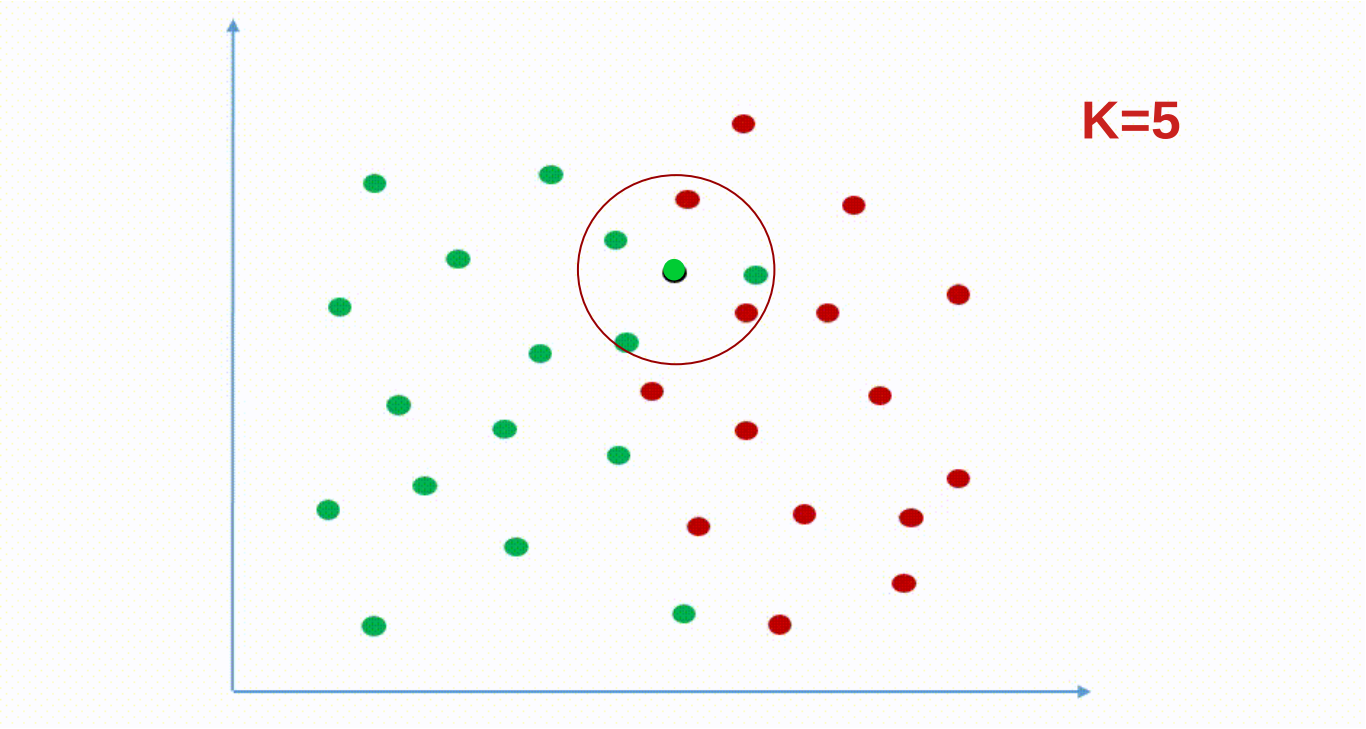
Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



Modélisation Prédictive

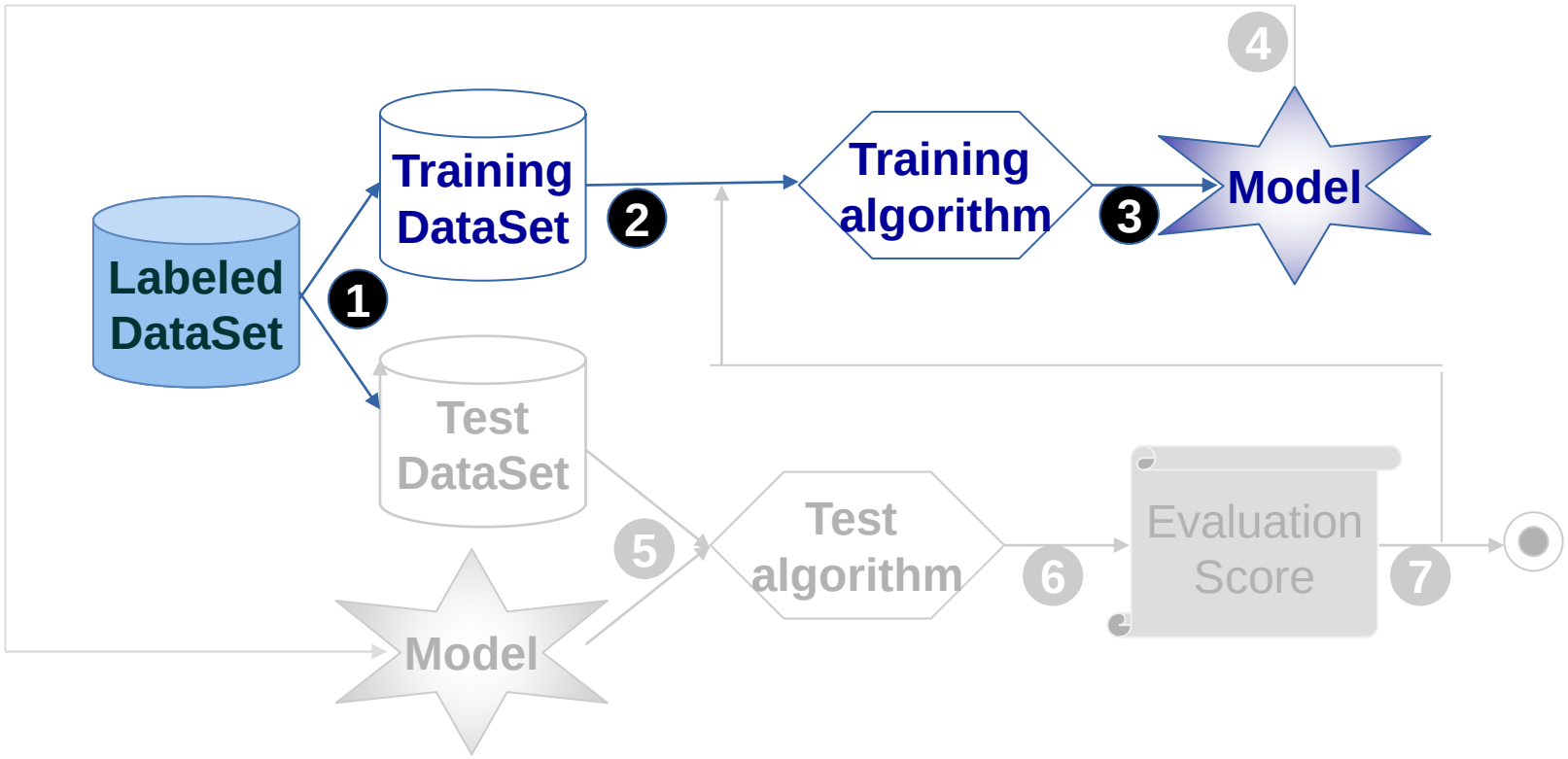
K-Nearest Neighbors (KNN)

Exemple : Soit la distribution de deux classes de données - Vert et Rouge. Pour une nouvelle donnée (marquée de couleur noire) où nous ne savons pas à quelle classe elle appartient.



La classification

Etape 1 : Apprentissage



Modélisation Prédictive

K-Nearest Neighbors (KNN)

Etapes :

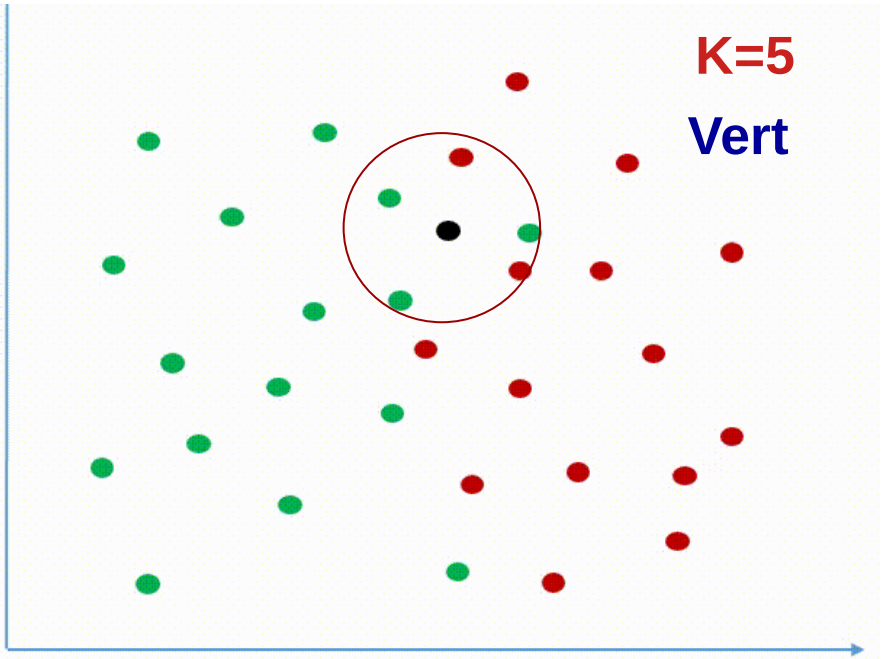
- choisir K ;
- calculer la distance entre le point cible et les autres points ;
- choisir les K premiers voisins dont la distance est la plus proche du point cible ;
- le point cible appartient à la classe majoritaire.

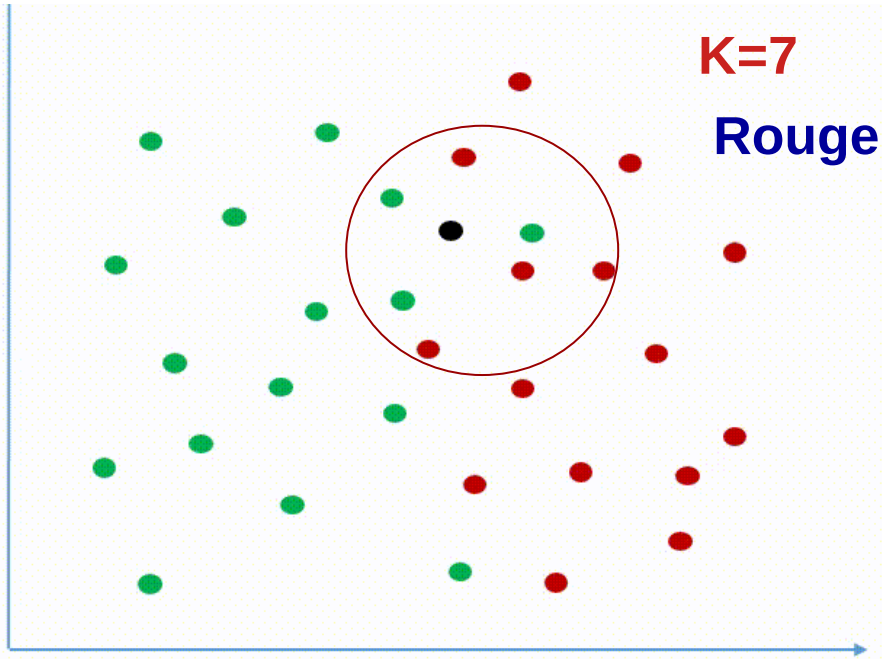
Modélisation Prédictive

K-Nearest Neighbors (KNN)

Choix de K :

Différents choix de K sur les mêmes données peuvent produire des résultats différents.





Modélisation Prédictive

Choix de K :

Il n'y a pas de règle ou de formule pour fixer la valeur de K, mais voici quelques lignes directrices :

- Il est conseillé de choisir une valeur impaire de K pour éviter tout lien entre les classes voisines les plus fréquentes.

- exécuter l'algorithme sur un ensemble de test et évaluer la prédiction
- → augmenter et diminuer K jusqu'à ne plus augmenter la précision de la prédiction.

Modélisation Prédictive

Remarques :

- une très grande valeur de K va à l'encontre de l'objectif de l'algorithme K-NN où vous pourriez finir par explorer des données en dehors du voisinage des données considérées.

- une petite valeur de K est efficace en termes de calcul et, comme prévu, une grande valeur de K peut devenir coûteuse en termes de calcul.

Modélisation Prédictive

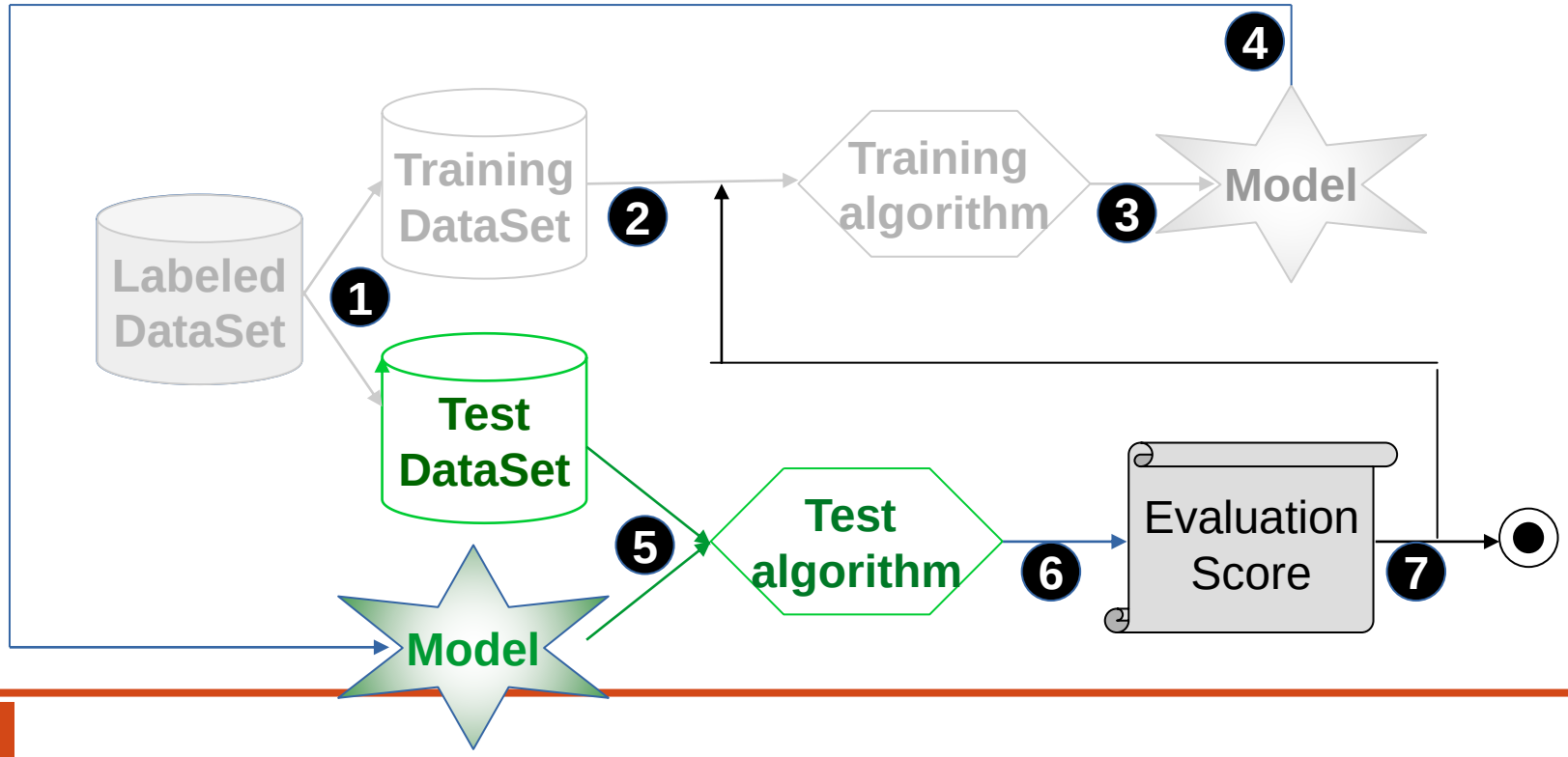
Traduction en Python :

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train,y_train)

y_pred_knn = knn.predict(X_test)

for i in range( len(y_test) ) :
    print(y_test[i], y_pred_knn[i])
```

Modélisation Prédictive

Etape 2 : Test

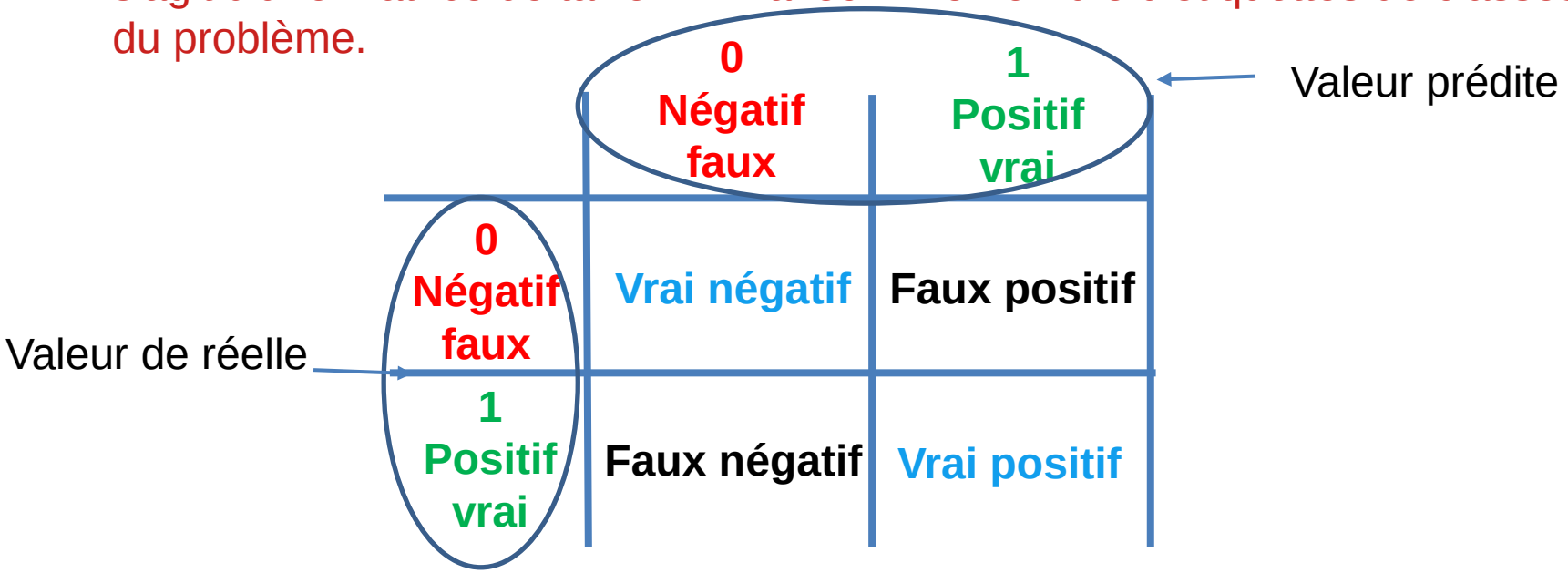
Un bon modèle est un modèle qui généralise : la généralisation, c'est la capacité d'un modèle à faire des prédictions sur de nouvelles données : c'est la phase de test.



Modélisation Prédictive

Il existe plusieurs métriques à l'aide desquelles nous pourrons évaluer le modèle.

1 **La matrice de confusion** aide à évaluer les performances du modèle. Il s'agit d'une matrice de taille n x n avec n : le nombre d'étiquettes de classes du problème.



Modélisation Prédictive

1 L'accuracy :

L'accuracy indique le pourcentage de valeurs correctement prédites sur toutes les observations de données.

```
from sklearn.metrics import accuracy_score, confusion_matrix

print('Confusion matrix knn \n', confusion_matrix(y_test,y_pred_knn))
print('Accuracy knn', accuracy_score(y_test,y_pred_knn))
```

VP

FN

```
Confusion matrix knn
 [[1486  109]
 [ 237  168]]
Accuracy knn 0.827
```

$$Accuracy = \frac{\text{Vrai positif} + \text{Vrai négatif}}{\text{Vrai positif} + \text{Faux positif} + \text{Faux négatif} + \text{Vrai négatif}}$$

Modélisation Prédictive

Parfois, l'accuracy n'est pas toujours la meilleure mesure

2 TPR (True Positive Rate) or Recall:

$$\text{Recall} = \frac{\text{Vrai positif}}{\text{Vrai positif} + \text{Faux négatif}}$$

Indique, parmi toutes les observations de données positives, combien ont été réellement identifiées comme positives par le modèle.
→ depuis la matrice de confusion, c'est TP divisé par les valeurs de la ligne dans laquelle TP est présent.

3 La Précision

$$\text{Precision} = \frac{\text{Vrai positif}}{\text{Vrai positif} + \text{Faux positif}}$$

→Sur toutes les observations qui ont été identifiées comme positives par le modèle, combien sont en fait vrai.

Modélisation Prédictive

K-Nearest Neighbors (KNN)

```
from sklearn.metrics import precision_score, recall_score

print('Recall knn : ', recall_score(y_test,y_pred_knn))
print('Precision knn : ', precision_score(y_test,y_pred_knn))
```

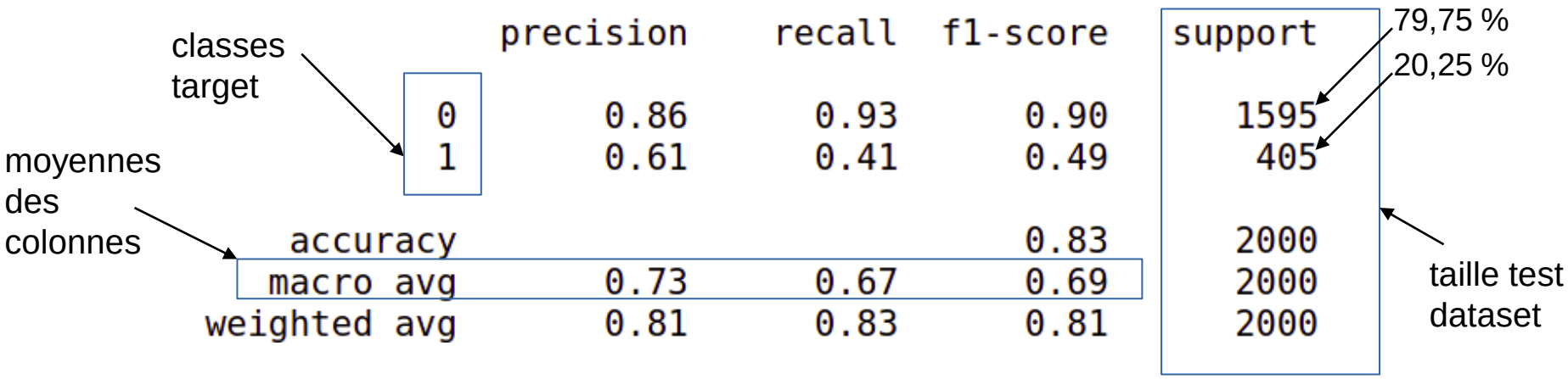
```
Recall knn : 0.4148148148148148
Precision knn : 0.6064981949458483
```

Nous pouvons construire un rapport textuel indiquant les principales métriques de classification :

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_knn))
```

Modélisation Prédictive

K-Nearest Neighbors (KNN)



	precision	recall	f1-score	support
0	0.86	0.93	0.90	1595
1	0.61	0.41	0.49	405
accuracy			0.83	2000
macro avg	0.73	0.67	0.69	2000
weighted avg	0.81	0.83	0.81	2000

macro avg = $(\text{precision}_{\text{class0}} + \text{precision}_{\text{class1}}) / 2 = (0.86 + 0.61) / 2$

weighted avg = $(TP_{\text{class0}} + TP_{\text{class1}})/(nb_{\text{class0}} + nb_{\text{class1}})$ ou bien

$= 0.7975 * 0.86 + 0.2025 * 0.61$

Modélisation Prédictive

K-Nearest Neighbors (KNN)

f1-score = moyenne harmonique pondérée de la précision et du recall.
→ un f1-score atteint sa meilleure valeur à 1 et son pire score à 0.
→ utilisée lorsqu'on veut déterminer s'ils existent des liens de proportionnalité inverse entre Precision et Recall.

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

Conclusion :

Sur la base des mesures de performance ci-dessus, nous pouvons nous baser sur la Precision, le Recall ou l'Accuracy globale.

Si les données sont équilibrées, ce qui signifie une répartition entre 50/50 échantillons vrais et négatifs, on peux choisir la précision.

Modélisation Prédictive

GridSearch : Une GridSearch est utilisée pour exécuter plusieurs modèles avec plusieurs valeurs de K pour déduire la plus optimale.

Execution :

Différentes valeurs de K (impaires)

Critère d'évaluation

Cross validation

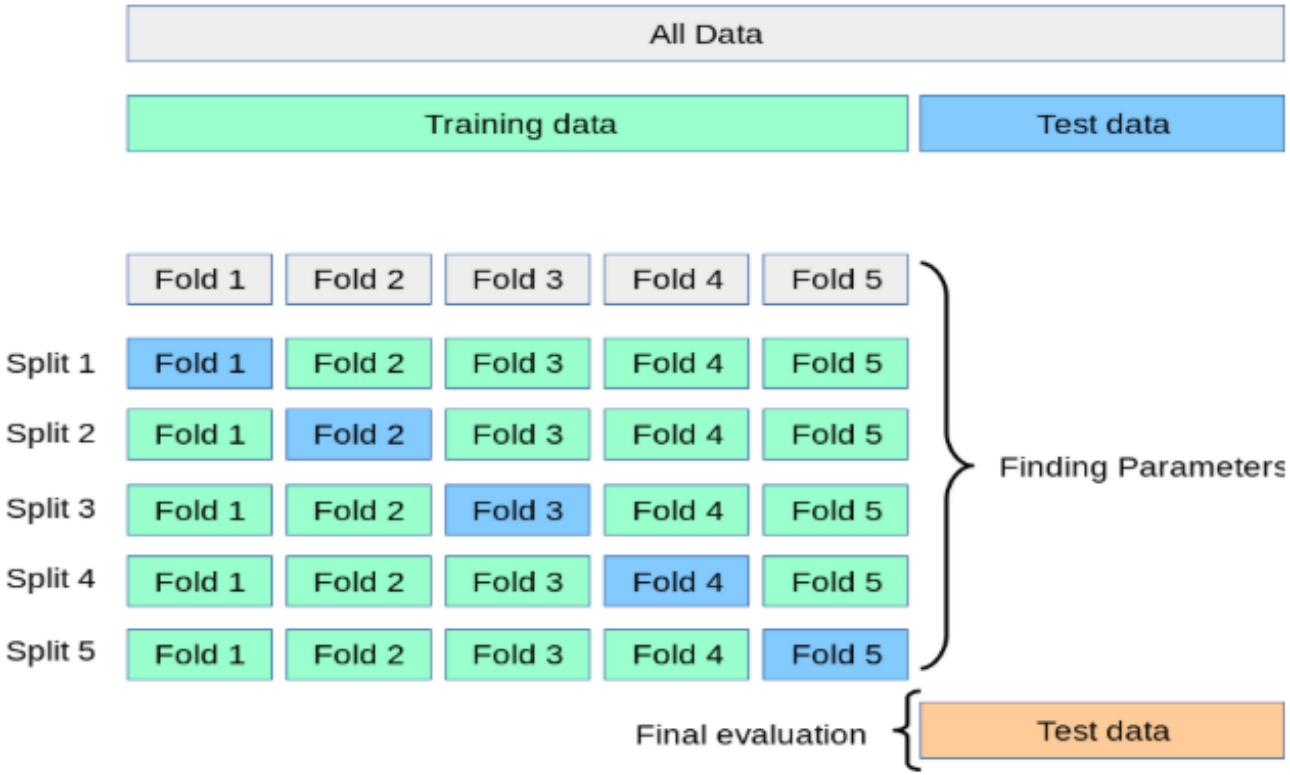
```
from sklearn.model_selection import GridSearchCV
parameters = {'n_neighbors' : [1,3,5,7,9,11,13]}
model = KNeighborsClassifier()

clf = GridSearchCV(model, parameters, scoring='accuracy', cv=5)
grille = clf.fit(X_train_sc,y_train)

print(grille.best_params_)
print(grille.best_score_)
```

```
{'n_neighbors': 9}
0.8303750000000001
```

Modélisation Prédictive

K-Nearest Neighbors (KNN)

Cross validation :



Modélisation Prédictive

K-Nearest Neighbors (KNN)

Evaluation :

```
y_pred_knn_o = grille.predict(X_test_sc)

print('Confusion matrix knn op \n', confusion_matrix(y_test,y_pred_knn_o))
print('Accuracy knn op', accuracy_score(y_test,y_pred_knn_o))
print('Recall knn op', recall_score(y_test,y_pred_knn_o))
print('Precision knn op', precision_score(y_test,y_pred_knn_o))
```

```
Confusion matrix knn op
 [[1516   79]
 [ 252  153]]
Accuracy knn op 0.8345
Recall knn op 0.37777777777777777
Precision knn op 0.6594827586206896
```