Mise à jour: 20/01/2025

ALIASE
IA - SERVICE & FORMATION

Formation Machine Learning

Atelier Pratique AP-ML2

ALIASE

copyright © 2021 aliase-formation.com

Atelier Pratique AP-ML2 : Exercice ML2.5

ALIASE
IA - SERVICE & FORMATION

Objectif : Calculer la chance de survivre sur le Titanic

1) Charger les colonnes ***pclass, survived, sex, age*** du fichier ***Titanic.xls*** dans un Dataframe df ,

puis afficher le début du Dataframe

2) ENCODAGE : Remplacer les mots ***"male" et "female"*** par des valeurs numériques ***0 et 1***

syntaxe : df['sex'].replace(['male', 'female'], [0,1], inplace=True)

3) Afficher les statistiques de base : ***df.describe()***

4) Supprimer les lignes où il y a des données manquantes

5) Séparer le dataset en 2 parties : ***Y (target) et X (features),*** où la target est la colonne ***survived***

Atelier Pratique AP-ML2 : Exercice ML2.5

ALIASE
IA - SERVICE & FORMATION

6) Créer et entrainer le modèle

7) Définir une fonction ***predire (model, pclass, sex, age)***

Entrée : les information sur un passager (pclass, sex, age)

Sortie : l'information que le passager ***survivrait (1) ou pas (0)***

8) Afficher le ***% de chance de survivre d'un passager*** (la probabilité que predict retourne 1)

en utilisant la fonction de Sklearn : ***model.predict_proba(passager)***

Solution → Voir MLExo2.5.py

Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

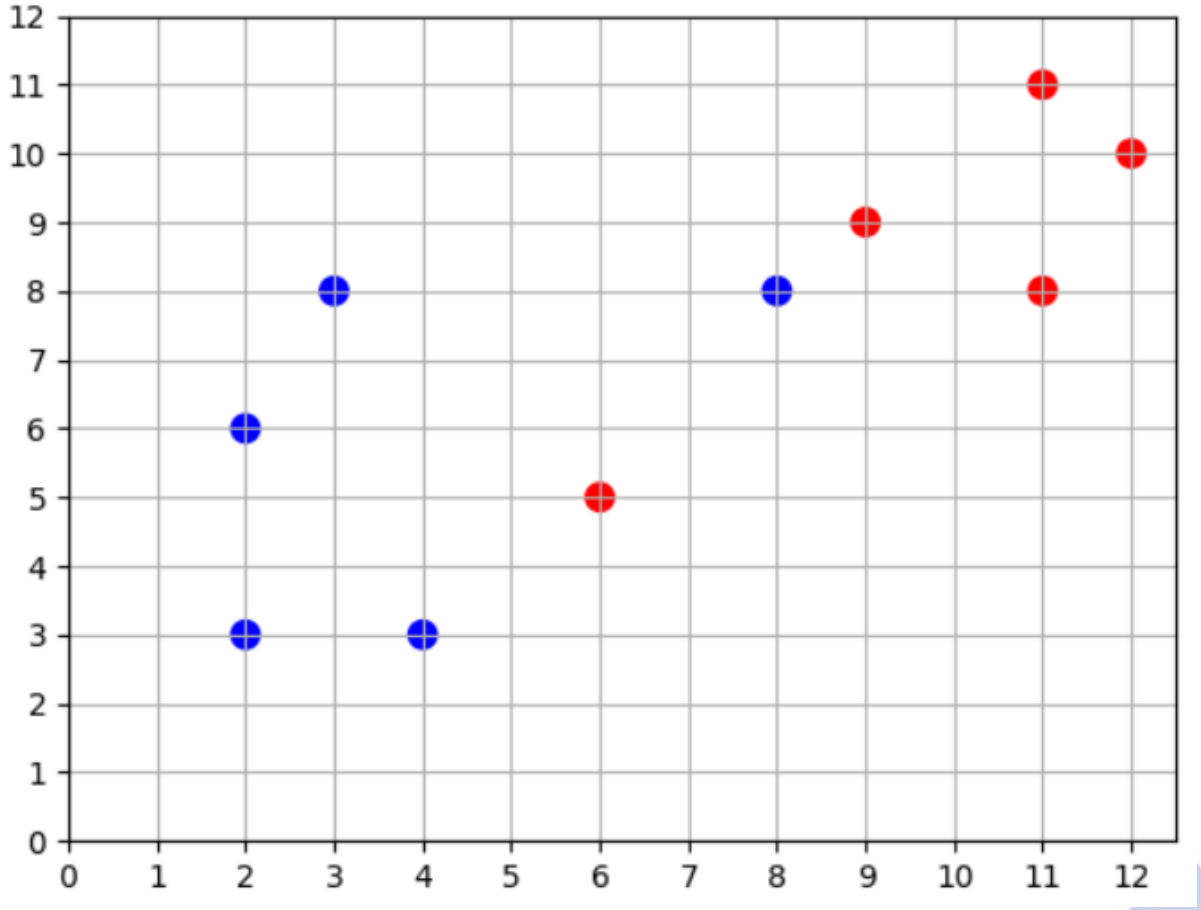
Objectif : Programmer l'algorithme KNN

1) Charger le dataset à partir du fichier ***dataset-KNN.csv*** dans un dataframe ***df*** puis afficher ***df***

x1	x2	label
2	3	0
2	6	0
4	3	0
3	8	0
8	8	0
6	5	1
9	9	1
11	8	1
11	11	1
12	10	1

Atelier Pratique AP-ML2 : Exercice ML2.6

2) Afficher les données sur un plan en utilisant la fonction ***scatter .*** La figure suivante devrait s'afficher :

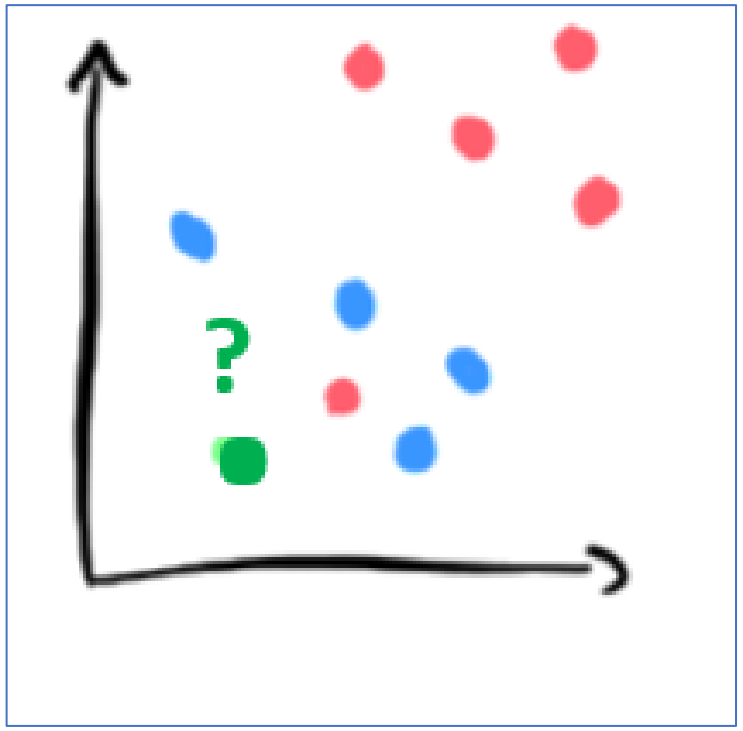


Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

3) Ajouter une colonne '***distance***':

x1	x2	label	distance
2	3	0	
2	6	0	
4	3	0	
3	8	0	
8	8	0	
6	5	1	
9	9	1	
11	8	1	
11	11	1	
12	10	1	



Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

4) isoler les features dans **X** , et les colonnes ***label + distance*** dans **Y**

X

x1	x2
2	3
2	6
4	3
3	8
8	8
6	5
9	9
11	8
11	11
12	10

Y

label	distance
0	
0	
0	
0	
0	
1	
1	
1	
1	
1	

Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

5) Définir une fonction ***distance (pt1, pt2)*** qui calcule la distance Euclidienne entre deux points pt1 et pt2.

pt1 et pt2 sont des **tuples** :

- **pt1 (a1, a2, . . . , an)**
- **pt2 (b1, b2, . . ., bn)**

$$d(A,B) = \sqrt{\sum_{i=1}^{n} (ai - bi)^2}$$

Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

6) Définir une fonction ***myKNN (newpt, K)*** qui retourne le label qu'elle va attribuer au nouveau point ***newpt*** :

Démarche :

a. Pour chaque point ***pt*** dans la dataset :

- calculer la distance entre ***newpt*** (nouveau point passé en paramètre) et ce point ***pt***
- stocker la ***distance*** calculée dans la colonne ***distance*** de ***Y***

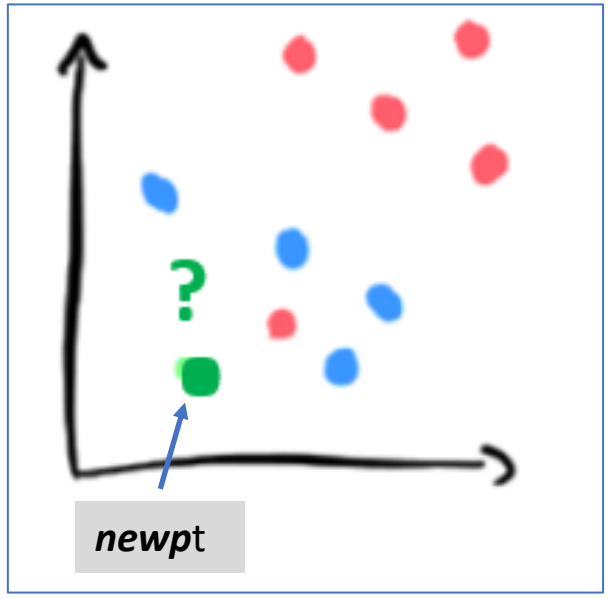
X

pt

x1	x2
2	3
2	6
4	3
3	8
8	8
6	5
9	9
11	8
11	11
12	10

Y

label	distance
1	8.60
0	2.41
1	9.67
0	4.32
0	5.39
1	1.01
1	7.53
1	8.60
1	10.42
0	3.25



Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

6) Définir une fonction ***myKNN (newpt, K)*** qui retourne le label qu'elle va attribuer au nouveau point ***pt*** :

Démarche (cont):

b. Copier Y dans ***Y_temp***, puis trier ***Y_temp*** sur la colonne ***distance***, puis afficher le *Y_temp*

Y_temp

label	distance
1	1.01
0	2.41
0	3.25
0	4.32
0	5.39
0	6.46
1	7.53
1	8.60
1	9.67
1	10.74

Tri

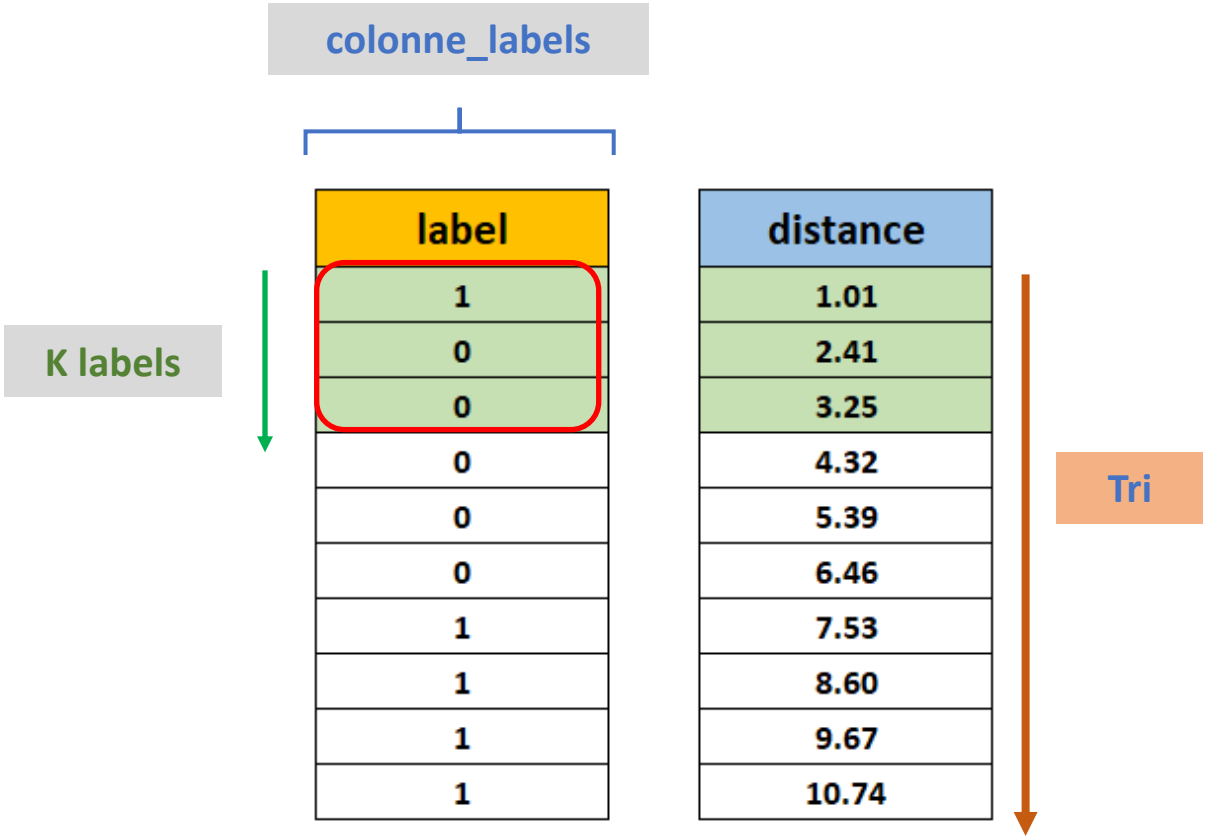
Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

6) Définir une fonction ***myKNN (pt, K)*** qui retourne le label qu'elle va attribuer au nouveau point ***pt*** :

Démarche (cont):

c. Isoler la colonne **'label'** avec <u>uniquement</u> les **K premiers labels** (en créant une série ***colonne_labels***)



label	distance
1	1.01
0	2.41
0	3.25
0	4.32
0	5.39
0	6.46
1	7.53
1	8.60
1	9.67
1	10.74

Atelier Pratique AP-ML2 : Exercice ML2.6

ALIASE
IA - SERVICE & FORMATION

Démarche : (cont)

d. Sur ces K premiers labels, calculer le **nombre d'occurrence** de chaque label (***value_counts()***)

Le résultat du ***value_counts()*** est une série qu'on appellera ***lab_count***

e. Le label recherché correspond à l'index du plus grand des nombres d'occurrences.

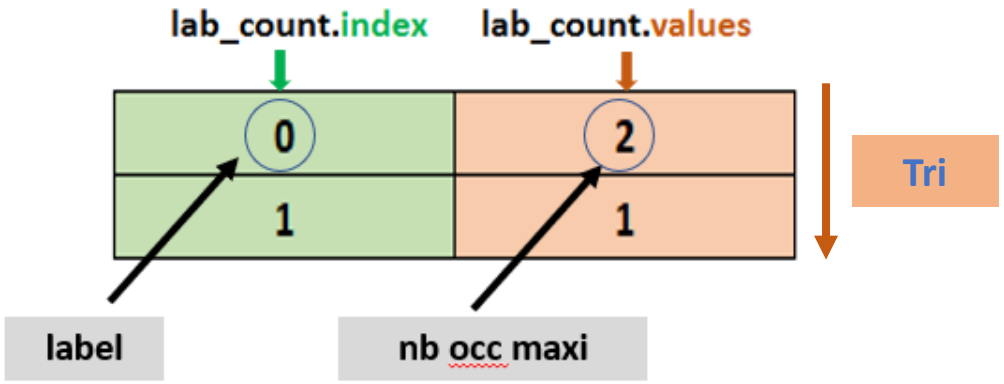
Pour y accéder :

. Méthode-1:

- utiliser argmax()

. Méthode-2:

- trier ***lab_count*** dans l'ordre décroissant
- récupérer le label , c'est ***l'index*** de la ligne 0



copyright © 2021 aliase-formation.com

Atelier Pratique AP-ML2 : Exercice ML2.6

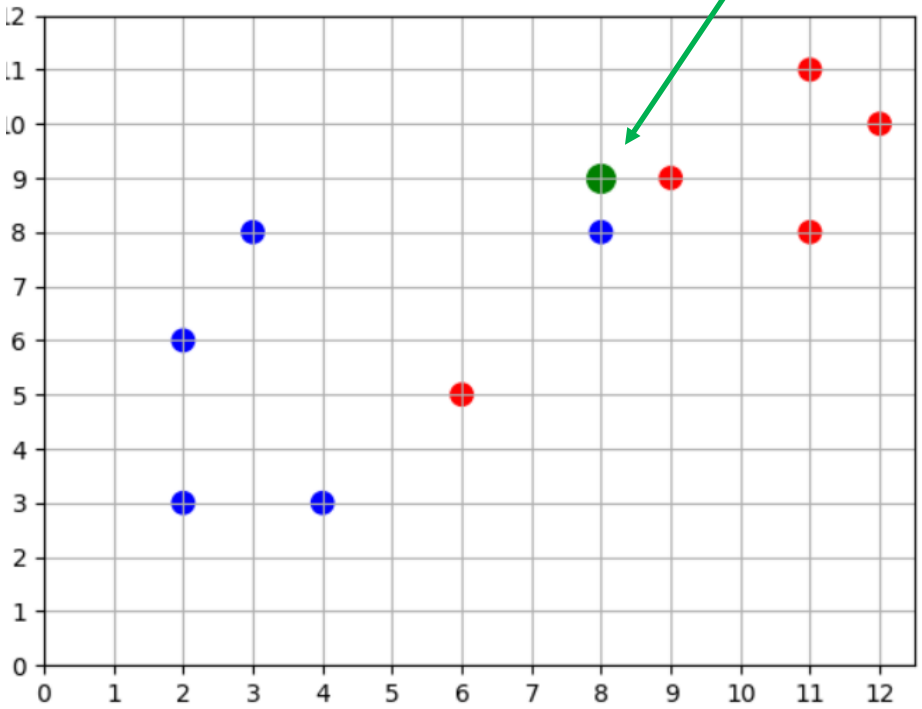
ALIASE
IA - SERVICE & FORMATION

Démarche : (cont)

7) Faire des prédictions en entrant des points (pt) avec des coordonnées x1 et x2

- donner les features (x1,x2) d'un nouveau point pt
- afficher le label attribué au nouveau point pt

8) Afficher le Dataset + le nouveau point en couleur = 'green'



Solution → Voir MLExo2.6.py

Atelier Pratique AP-ML2 : Exercice ML2.7

ALIASE
IA - SERVICE & FORMATION

Objectif : Tester l'algorithme *MyKNN* avec le dataset des images (chiffres manuscrits)

Classe 0
Classe 1
Classe 2
Classe 3
Classe 4
Classe 5
Classe 6
Classe 7
Classe 8
Classe 9

Solution → Voir MLExo2.7.py

Atelier Pratique AP-ML2 : Exercice ML2.8

ALIASE
IA - SERVICE & FORMATION

Objectif : Classification des images (digits) avec le modèle SVC (Support Vector Classification)

- **SVC** est un modèle de classification dont l'implémentation est basée sur la librairie **SVM**

- **SVM** (Support Vector Machine) sont une famille d'algorithmes d'apprentissage automatique de type supervisé et qui peuvent être utilisées pour des problèmes de :
  - discrimination (à quelle classe appartient un échantillon)
  - régression (SVR)
  - détection d'anomalie

```
from sklearn import svm

# Créer le modèle

model = svm.SVC()
```

Solution → Voir MLExo2.8.py

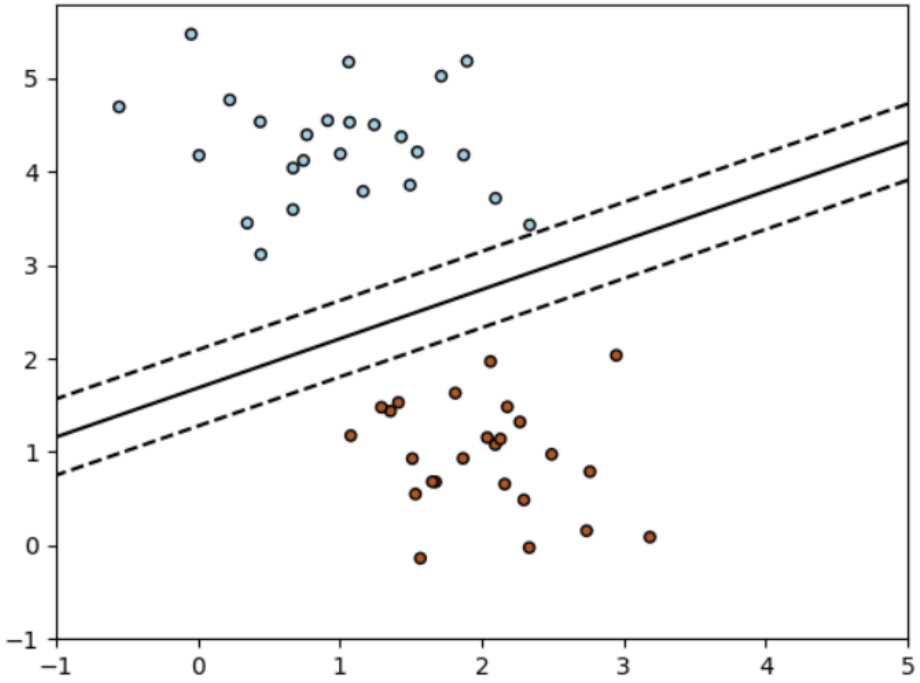
Atelier Pratique AP-ML2 : Exercice ML2.9

ALIASE
IA - SERVICE & FORMATION

Objectif : Classification des images (digits) avec le modèle SGDClassifier

- **SGDClassifier** est un Classificateur **linéaire** avec entraînement **SGD** (Stochastic Gradient Descent)

- **Il** implémente une simple routine d'apprentissage de descente de gradient stochastique qui prend en charge différentes fonctions de perte et pénalités pour la classification.




```
from sklearn.linear_model import SGDClassifier

# Créer le modèle
model = SGDClassifier(max_iter=1000, eta0=0.001)
```

Solution → Voir MLExo2.9.py

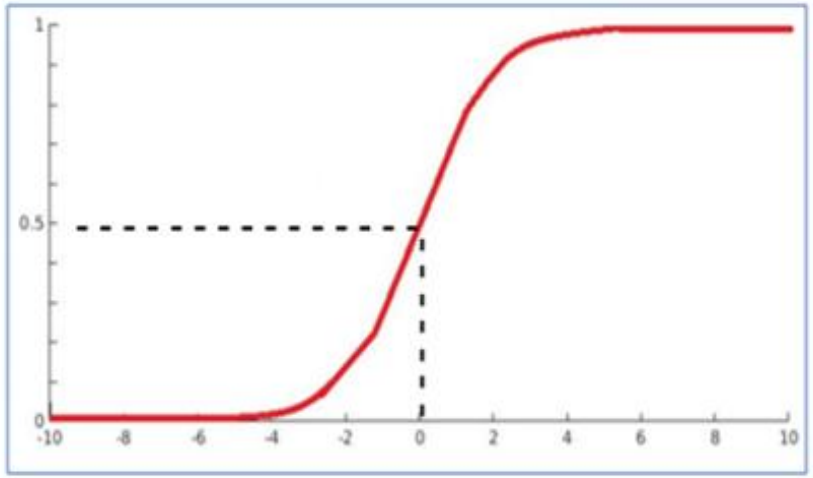
Atelier Pratique AP-ML2 : Exercice ML2.10

ALIASE
IA - SERVICE & FORMATION

Objectif : prévoir si un patient est **diabétique ou non avec le modèle LogisticRegression**

- **LogisticRegression** est un Classificateur **binaire**
- La cible (y) n'a que 2 valeurs, Ex "Spam" ou "pas Spam.
- Il utilise la fonction Sigmoïde pour les prédictions

Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$ (Classification binaire)




```
from sklearn.linear_model import LogisticRegression

# Créer le modèle
model = LogisticRegression()
```

Si proche de 0, réponse A
Si proche de 1, réponse B

Solution → Voir MLExo2.10.py

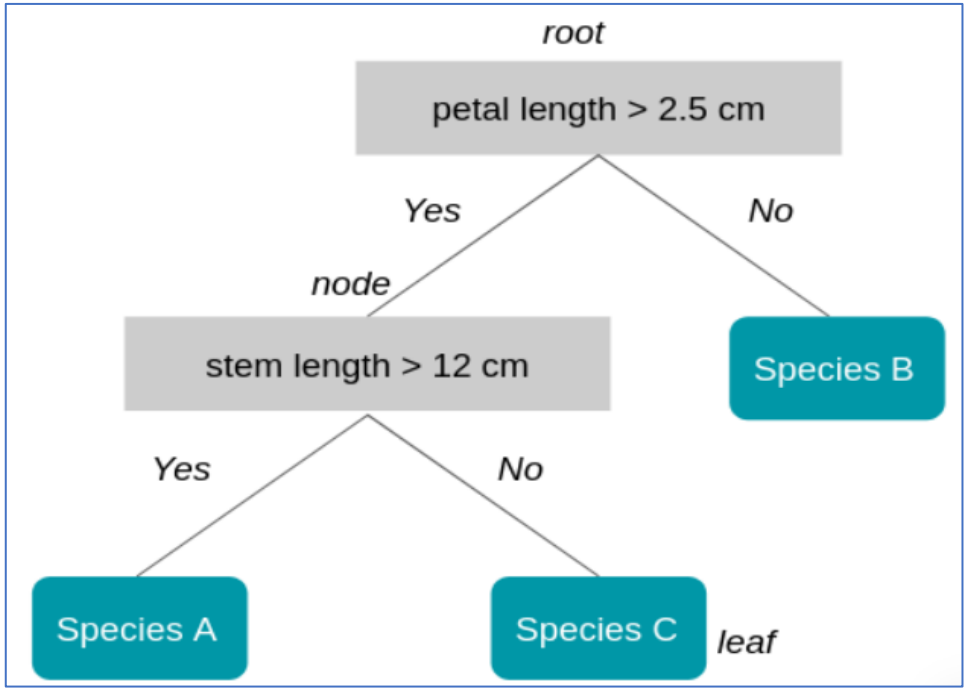
Atelier Pratique AP-ML2 : Exercice ML2.11

ALIASE
IA - SERVICE & FORMATION

Objectif : Classification des **fleurs Iris** avec le modèle **DecisionTreeClassifier**

Algorithme de classification basé sur l'utilisation d'un arbre de décision comme modèle prédictif.

On l'utilise notamment en fouille de données.



. **Racine :** contient toutes les observations

. **Nœud :** représente une règle

. **Parcourir** l'arbre consiste à vérifier un certain nombre de règles

```
from sklearn.tree import DecisionTreeClassifier

# Créer le modèle
model = DecisionTreeClassifier()
```

Solution → Voir MLExo2.11.py

copyright © 2021 aliase-formation.com

Atelier Pratique AP-ML2 : Exercice ML2.12

ALIASE
IA - SERVICE & FORMATION

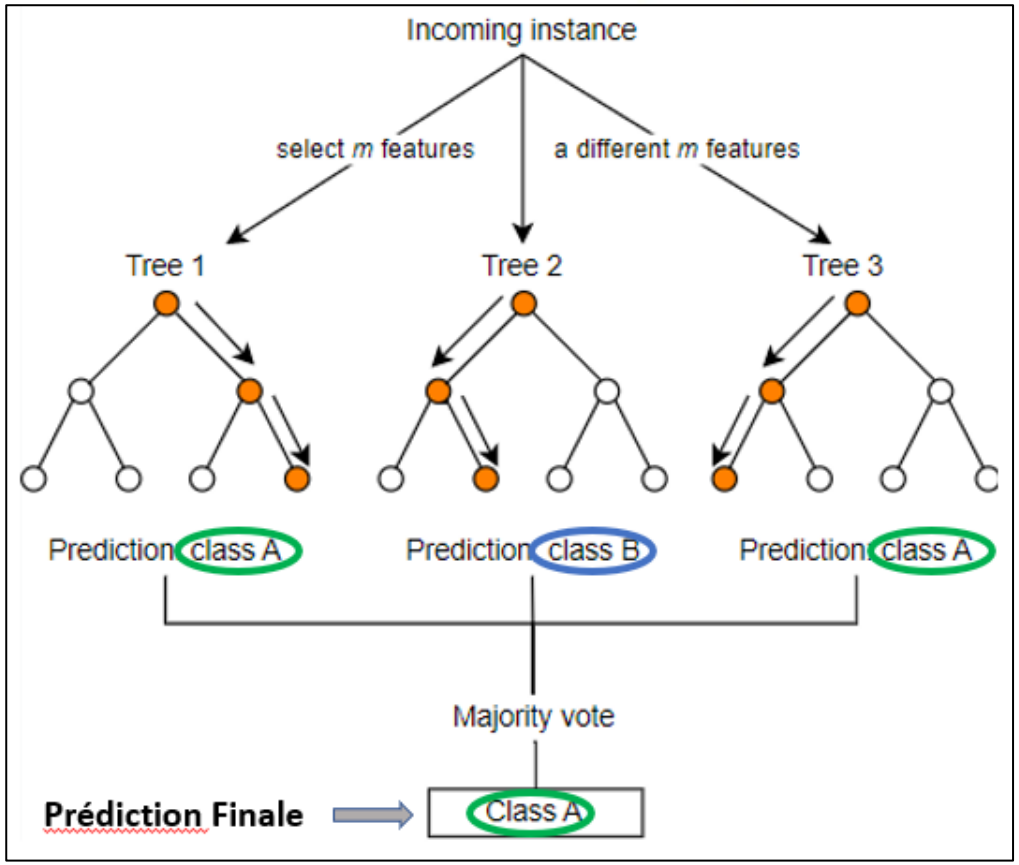
Objectif : Classification des **fleurs Iris** avec le modèle **RandomForestClassifier**

```
from sklearn.ensemble import RandomForestClassifier


# Créer le modèle
model = RandomForestClassifier()
```

Principe : Le modèle

1. construit plusieurs **arbres de décision** à partir de sous-échantillons aléatoires des données d'entraînement.
2. Combiner les **prédictions des modèles (Tree1, Tree2, Tree3)** pour améliorer la performance de la prédiction globale
3. La **prédiction finale (Classe A)** est déterminée par un vote majoritaire (dans le cas d'une classification) ou une moyenne (dans le cas d'une régression) des prédictions de tous les arbres.



Solution → Voir MLExo2.12.py