Machine Learning (ML)

Chap 1 : Les concepts de base

David TCHOUTA

Académie Française du Numérique
www.frenchtechacademie.fr
Tél/Whatsapp : +33 (0)7 49 62 72 49

June 23, 2022

"Les gens qui réussissent font le premier pas, ils sont tournés vers l'action." Brian Tracy

Table de matières

Objectifs du chapitre

Dans ce premier chapitre, nous allons apprendre les concepts de base du Machine Learning (ML).

À la fin de ce chapitre, vous serez en mesure :

- de définir le Machine Learning
- de calculer les statistiques descriptives
- d'importer, de lire et de manipuler les données avec Pandas
- de faire des calculs avec Numpy
- de construire les graphiques avec Matplotlib

Definition

Definition

Le Machine Learning (ML) est la façon dont nous prenons les données et les transformons en information. Nous utilisons la puissance de calcul des ordinateurs pour analyser les données du passé et prédire les résultats des nouvelles données.

Le Machine Learning est de plus en plus répandu dans nos usages quotidiens. Par exemple, lorsque l'algorithme de Netflix récommande un film, il se base sur les films que les autres utilisateurs ont regardés (y compris nous-même) pour faire cette recommandation. De même que les prix pratiqués par Amazon se basent sur la manière dont les articles similaires ont été vendus par le passé. Il en va de même pour la surveillance des financements illicites.

Les prérequis

Dans ce chapitre, nous utilisons bien évidemmnet le langage de programmation Python, ce qui suppose que l'apprenant a un bon niveau en Python.

Les packages tels que Pandas (lecture des données), Numpy (calcul des données numériques), Matplotlib (construire les graphiques) et Scikit-learn (manipulation des modèles de ML) seront utilisés.

Attention !

Nous allons couvrir la théorie et la pratique des modèles de ML, en mettant l'accent sur la manière dont ils sont utilisés dans le monde réel.

Contenu du cours 1

En ML, on distingue l'apprentissage supervisé de l'apprentissage non supervisé.

On parle **d'apprentissage supervisé lorsque nous connaissons la valeur réelle de la variable cible** (par exemple, prédire à quel prix une maison sera vendu) tandis que **l'apprentissage non supervisé** est determiné par le fait que **la réponse passée n'est pas connu** (par exemple, déterminer les sujets de discussion dans une revue d'économie).

Contenu du cours 2

Nous allons nous concentrer sur les algorithmes d'apprentissage supervisé tels que la régression (prédire une valeur numérique, par exemple la croissance économique d'une région) et la classification (prédire à quelle classe appartient une donnée, par exemple prédire si une transaction est une fraude ou pas). Nous allons nous concentrer sur les problèmes de classification.

Nous allons prédire quel passager survivra au crash du Titaninc, déterminer un chiffre manuscrit à partir d'une image ou encore utiliser les données d'une biopsie pour déterminer si une cellule est cancéreuse ou pas.

Contenu du cours 3

Nous verrons en detail les algorithmes suivants :

1. la regréssion logistique (**Logistic Regression**)
2. les arbres de décision (**decision Trees**)
3. les forêts aléatoires (**Random Forests**)
4. les réseaux de neurones (**Neural Networks**)

Les statistiques de base

La moyenne (mean) est la somme des valeurs de la distribution divisée par le nombre d'elements de la distribution.

La médiane (median) est la valeur centrale d'une distribution.

Soit la distribution d'âge suivante : 15, 16, 18, 19, 22, 24, 29, 30, 34

moyenne = (15 + 16 + 18 + 19 + 22 + 24 + 29 + 30 + 34) / 9 = 207/9 = 23

La médiane est la cinquième valeur (22) puisqu'il y a 9 valeurs au total.

Exercice d'application 1

Moyenne et médiane

Calculer la moyenne est la médiane de la distribution suivante : 0,1,1,2,6

Les percentiles 1

la **médiane** correspond au **50e percentile (deuxième quartile)**, ce qui signifie que 50% des données sont inférieures à la médiane et 50% des données sont supérieures à la médiane. Elle nous renseigne sur l'endroit où se situe la valeur centrale. Nous nous intéressons également au **25e percentile (premier quartile)** et au **75e percentile (troixième quartile)**.

Le 25e percentile est la valeur qui se trouve à un quart des données. C'est la valeur pour laquelle 25% des données sont inférieures à cette valeur (et 75% des données sont supérieures à cette valeur)

Les percentiles 2

De même, le 75e percentile correspond aux trois quarts des données. Il s'agit de la valeur pour laquelle 75% des données sont inférieures à cette valeur (et 25% des données sont supérieures à cette valeur).

Reprenons notre distribution de l'âge : 15, 16, 18, 19, 22, 24, 29, 30, 34

Nous avons 9 valeurs, donc 25% des données correspondraient à la 2e valeur environ. La 3e valeur est supérieure à 25% des données, donc le 25e percentile est 18.

De même, 75% des données correspondent à la 6e valeur environ. La 7e valeur est donc supérieur à 75% des données. Ainsi, le 75e percentile est de 29 (la 7e valeur des données).

Les percentiles 3

Attention !

S'il y a un nombre pair de données, pour trouver la médiane (ou le 50e percentile), vous prenez la moyenne des deux valeurs du milieu.

Exercice d'application 2

Percentiles

Soit la distribution suivante : 0, 1, 1, 1, 1, 1, 2, 2, 2, 3, 6

1) le 25e percentile est : .....

2) le 50e percentile est : .....

3) le 75e percentile est : .....

Écart-type & Variance 1

Pour avoir une compréhension plus poussée de la distribution, nous pouvons calculer **l'écart-type et la variance**, qui répresentent **les mesures de dispersion des données**.
Nous mesurons de combien un point se situe par rapport à la moyenne.

Soit la distribution suivante : **15, 16, 18, 19, 22, 24, 29, 30, 34**
Sa moyenne est de 23. Si nous calculons la distance de chaque point par rapport à la moyenne, (23-15 = 8 pour la première valeur), on a le tableau suivant :

8, 7, 5, 4, 1, 1, 6, 7, 11

On élève ces valeurs au carré et on les additionne :
$8^2 + 7^2 + 5^2 + 4^2 + 1^2 + 1^2 + 6^2 + 7^2 + 11^2 = 362$

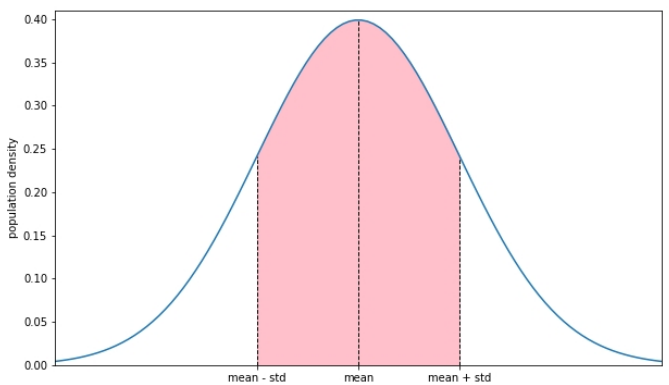
Écart-type & Variance 2

Si nous **divisons cette valeur par le nombre total de valeurs** et cela nous donne **la variance** : $362 / 9 = 40.22$.

En prenant **la racine carré de la variance**, nous obtenons **l'écart-type** : $\sqrt{40.22} = 6.34$

Écart-type & Variance 3

Si nos données sont normalement distribuées comme dans le graphique ci-dessous, 68% de la population se situe dans la fourchette d'un écart-type de la moyenne. Dans le graphique, nous avons mis en évidence la zone située à l'intérieur d'un écart-type de la moyenne. Vous pouvez voir que la zone rose représente environ deux tiers (plus précisément 68%) de la surface totale sous la courbe. Si nous supposons que nos données sont normalement distribuées, nous pouvons dire que 68% des données se situent dans un écart-type de la moyenne.



Écart-type & Variance 4

Dans notre exemple, bien que les âges ne soient probablement pas exactement distribués normalement, nous supposons qu'ils le sont et disons qu'environ 68% de la population a un âge compris dans un écart-type de la moyenne. Comme la moyenne est de 23 et l'écart-type de 6,34, nous pouvons dire qu'environ 68% des âges de notre population se situent entre 16,66 (23 - 6,34) et 29,34 (23 + 6,34).

Attention !

Même si les données ne présentent jamais une distribution normale parfaite, nous pouvons tout de même utiliser l'écart-type pour avoir une idée de la façon dont les données sont distribuées.

Exercice d'application 3

Écart-type & Variance

1) Qu'est-ce que la variance ?

2) Qu'est-ce que l'écart-type ?

3) Comment sont-ils calculés ?

4) Que traduit un écart-type élévé ou faible ?

Numpy : Calcul Statistique 1

Nous allons utiliser les **fonctions de Numpy** pour le calcul des statistique tel que la **moyenne (mean), la médiane (median), les percentiles, std, var**.

Importons Numpy et initialisons la variable data pour avoir la liste des âges :

```
import numpy as np
data = [15, 16, 18, 19, 22, 24, 29, 30, 34]
print("mean:", np.mean(data))
print("median:", np.median(data))
print("50th percentile (median):", np.percentile(data, 50))
print("25th percentile:", np.percentile(data, 25))
print("75th percentile:", np.percentile(data, 75))
print("standard deviation:", np.std(data))
print("variance:", np.var(data))
```

Exercice d'application 4

Calculez les percentiles

import numpy as np

print(np. ..... (data, 70))

Pandas : Lire et Manipuler les données 1

Pandas est un module Python qui nous aide à lire et à manipuler des données. Ce qu'il y a de bien avec Pandas, c'est que vous pouvez prendre des données et les afficher sous la forme d'un tableau lisible par l'homme, mais elles peuvent aussi être interprétées numériquement, ce qui vous permet d'effectuer de nombreux calculs.

Nous appelons le tableau de données un DataFrame.

Exercice d'application 5

L'objet Pandas

Comment appelle-t-on l'objet Pandas (Pandas data object) ?

Pandas : Lire et Manipuler les données 2

Pour importer Pandas : *import pandas as pd*

Nous allons travailler avec la base de données des passagers du Titanic. Pour chaque passager, nous aurons des données sur eux et nous saurons s'ils ont survécu ou non à l'accident.

Nos données sont stockées dans un fichier CSV (comma-separated values). Le fichier titanic.csv se trouve à `https://gist.github.com/fyyying/4aa5b471860321d7b47fd881898162b7`.

Pandas : Lire et Manipuler les données 3

La première ligne est l'en-tête, puis chaque ligne suivante est constituée des données d'un seul passager.

Chargons les données dans Pandas de manière à obtenir un Dataframe :

import pandas as pd
data = pd.read_csv(titanic.csv)
print(data.head())

```
In [7]: print(titanic.head())
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S
```

Exercice d'application 6

Lire les données

Écrire un programme qui permet d'afficher les 5 premières lignes et les 5 premières colonnes (de Passengerd jusqu'à Sex) de la base titanic.

Summary Statistics 1

Dans Pandas , **la méthode describe()** permet d'afficher **les statistiques descriptives**.
Pour forcer Python d'afficher un certain nombre de columns :
pd.options.display.max_columns = 4
print(data.describe())

```
In [76]: print(titanic.describe())
       PassengerId    Survived  ...       Parch        Fare
count   891.000000  891.000000  ...  891.000000  891.000000
mean    446.000000    0.383838  ...    0.381594   32.204208
std     257.353842    0.486592  ...    0.806057   49.693429
min       1.000000    0.000000  ...    0.000000    0.000000
25%     223.500000    0.000000  ...    0.000000    7.910400
50%     446.000000    0.000000  ...    0.000000   14.454200
75%     668.500000    1.000000  ...    0.000000   31.000000
max     891.000000    1.000000  ...    6.000000  512.329200
```

NB : Seules les Statistiques de colonnes numériques sont affichées.

Summary Statistics 2

Signification des indicateurs affichés :
count : C'est le nombre de lignes ayant une valeur. Dans notre cas, chaque passager a une valeur pour chaque colonne, donc la valeur est 891 (le nombre total de passagers).
mean : c'est la moyenne
std : c'est l'écart-type
min : c'est la plus petite valeur
25% : c'est le 25e percentile
50% : c'est le 50e percentile, ou encore la médiane
75% : c'est le 75e percentile
max : c'est la valeur la plus élévée
La méthode **describe()** permet d'avoir **quelques intuitions par rapport aux données (forme, dispersion, valeurs abérrantes, etc.)**.

Exercice d'application 7

Summary Statistics

Quelles sont les valeurs du maximum de Pclass et la médiane de age ?

Selection d'une seule colonne (Serie)

Pour selectionner une **seule colonne**, on utilise les square brackets([ ]) et le nom de la colonne qu'on veut selectionner.
col = data['age']
print(col)

Le **résultat** est ce qu'on appelle une **Pandas Series**. **Une Serie est comme un DataFrame, avec juste une seule colonne**.

Exercice d'application 8

Type

col = titanic['Survived']

Quel est le type de col ?

Selection de plusieurs colonnes

Pour selectionner plusieurs colonnes, nous mettons le nom des colonnes dans une liste :
['Survived', 'Age', 'Sex']. Puis nous insérons cette liste dans les brackets comme suit :

subset = data[['Survived', 'Age', 'Sex']]
print(subset.head())

Attention !

Lorsqu'on veut selectionner **une seule colonne**, on utilise les **single square brackets [ ]**. Lorsqu'on veut **récupérer plusieurs colonnes**, on utilise les **double square brackets [[ ]]**.

Exercice d'application 9

Selection des plusieurs colonnes

Écrire un programme permettant de selectionner les colonnes Fare, Age et SibSp.

Création d'une colonne

Nous allons créer une nouvelle colonne **genre** de type boolean (True pour les male et False pour les female : *data['Sex']=='male'*) en se basant sur la colonne Sex : *data['genre'] = data['Sex'] == 'male'*

```
In [103]: print(titanic.head())
   PassengerId  Survived  ...  Embarked  genre
0            1         0  ...         S   True
1            2         1  ...         C  False
2            3         1  ...         S  False
3            4         1  ...         S  False
4            5         0  ...         S   True
```

Attention !

Souvent, nos données ne sont pas dans le format idéal. Heureusement, Pandas nous permet de créer facilement de nouvelles colonnes basées sur nos données afin de les formater de manière appropriée.

Exercice d'application 10

Création d'une nouvelle colonne

Écrire un programme pour créer la colonne "First Class", qui est True si le passager est dans la Pclasse 1 et False sinon.

Numpy

Numpy est un module de Python qui permet de manipuler les listes (table data = numpy array) et les tableaux.

Bien qu'on peut faire certaines opérations sous Pandas, Il est préférable d'utiliser Numpy pour les calculs sur les tableaux. Pandas a en fait été construit en utilisant Numpy comme base.

Conversion de Pandas à Numpy

L'attribut values permet **de convertir une Pandas Serie** en **numpy array** :

data['Fare'].values
array([7.25 , 71.2833, 7.925, 53.1, 8.05, 8.4583, . . .

Le résultat est un tableau à une dimension. Vous pouvez le constater puisqu'il n'y a qu'un seul jeu de parenthèses et qu'il ne s'étend que sur la page (et non vers le bas).

Exercice d'application 11

Conversion d'une Serie en Numpy array

Écrivez le code pour obtenir un Numpy array des valeurs de la colonne 'Age' à partir du DataFrame titanic.

Conversion d'un Pandas DataFrame en Numpy array

Si nous avons **un Pandas DataFrame** (au lieu d'une Pandas Serie comme précédemment), nous pouvons toujours utiliser **l'attribut values**, mais **il renvoie un numpy array à 2 dimensions** :

data[['Pclass', 'Fare', 'Age']].values

```
array([[ 3.    ,  7.25  , 22.    ],
       [ 1.    , 71.2833, 38.    ],
       [ 3.    ,  7.925 , 26.    ],
       ...,
       [ 3.    , 23.45  ,     nan],
       [ 1.    , 30.    , 26.    ],
       [ 3.    ,  7.75  , 32.    ]])
```

C'est **un numpy** array à **deux dimensions**, car il y a **deux square brackets**.

Exercice d'application 12

Conversion d'un Pandas DataFrame en Numpy array

Quelle est la dimension du numpy array renvoyé par l'instruction suivante : *titanic[['Sex', 'Survived', 'Age']].values* ?

L'attribut shape

Nous utilisons **l'attribut shape** pour déterminer la **taille de notre numpy array**. **La taille** nous indique le **nombre de lignes et de colonnes de nos données**.

arr = data[['Pclass', 'Fare', 'Age']].values
print(arr.shape) $\Longrightarrow$ (891, 3)

Ce résultat signifie que nous avons 891 lignes et 3 colonnes.

Vous pouvez également utiliser **l'attribut shape** sur un **Pandas DataFrame** (titanic.shape).

Exercice d'application 13

Taille d'un Numpy array

Quel est l'output des instructions suivantes ?

arr = df[['Survived', 'Pclass']].values

print(arr.shape)

Selection des valeurs dans un Numpy Array

Soit le numpy array suivant :
arr = df[['Pclass', 'Fare', 'Age']].values
print(arr[0, 1])
Ce sera **la 2ème colonne de la 1ère ligne** (rappelez-vous que nous commençons à compter à 0). Ce sera donc le tarif du 1er passager, soit 7,25.

Nous pouvons également sélectionner **une seule ligne**, par exemple, la ligne entière du premier passager :
print(arr[0])

Pour sélectionner **une seule colonne** (la colonne Age par exemple) :
print(arr[:,2]) $\Longrightarrow$ array([3. , 7.25, 22.])

Exercice d'application 14

Selection

Soit le tabaleau suivant :

arr = data[['Pclass', 'Fare', 'Age']].values

Écrire un programme permettant d'afficher/selectionner les frais de tous les passagers.

Selection des données en fonction critère

Créons un sous-ensemble de données des passagers ayant moins de 18 ans (mineurs) :

mask = arr[:, 2] ¡ 18
print(arr[mask])

Une autre façon d'obtenir le même résultat :
print(arr[arr[:, 2] ¡ 18])

Attention !

Un masque (mask) est un tableau booléen (True/False) qui nous indique les valeurs du tableau qui nous intéressent.

Exercice d'application 15

Selection

Soit le tableau suivant :

arr = data[['Pclass', 'Fare', 'Age']].values

Complétez le programme suivant de manière à afficher/selectionner uniquement les passagers de la première classe (Pclass=1):

arr[..... == 1]

Somme des valeurs boolean

arr = df[['Pclass', 'Fare', 'Age']].values
mask = arr[:, 2] ¡ 18
Rappelez-vous que les **valeurs vraies sont interprétées comme 1** et que les **valeurs fausses sont interprétées comme 0**. Nous pouvons donc simplement additionner le tableau et cela équivaut à compter le nombre de valeurs vraies.
print(mask.sum())

Une autre façon d'obtenir le même résultat :
print((arr[:, 2] ¡ 18).sum())

Attention !

Sommer un tableau de valeurs booléennes donne le nombre de valeurs vraies.

Exercice d'application 16

Sommer un tableau de boolean

Soit le tableau suivant :

arr = data[['Pclass', 'Fare', 'Age']].values

Laquelle des propositions suivantes est correcte pour compter le nombre de passagers dans la classe 1 ?

1. (arr[0] == 1).sum()
2. (arr[:, 0] == 1).sum
3. (arr[:, 0] == 1).sum()
4. (arr[0] == 1).sum

Scatter plot 1

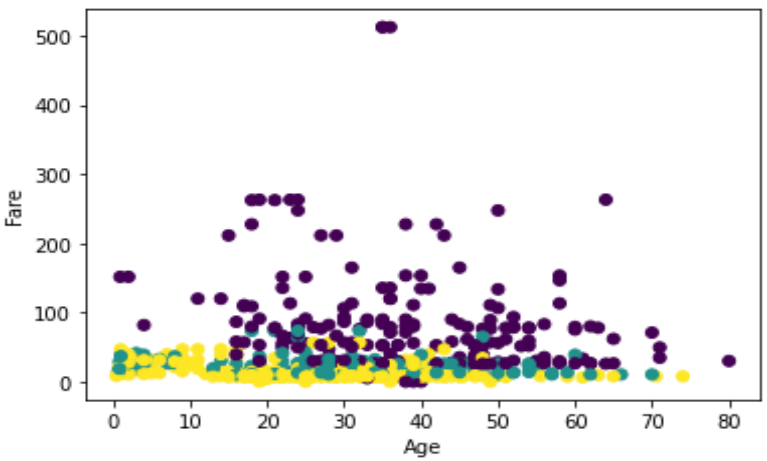
Nous pouvons utiliser la bibliothèque **matplotlib** pour tracer des graphiques. Le tracé des graphiques peut souvent nous aider à avoir un premier aperçu des données.

Nous utilisons la fonction **scatter()** pour tracer un **cross-plot**. Le premier argument est l'axe des x (direction horizontale) et le second argument est l'axe des y (direction verticale).

Nous pouvons également utiliser nos données **pour coder en couleur notre nuage de points**. Cela donnera à chacune des 3 classes une couleur différente. Nous ajoutons le **paramètre c** et lui donnons une Pandas Series. Dans ce cas, notre Pandas Series a 3 valeurs possibles (1ère, 2ème et 3ème classe), nous verrons donc nos points de données recevoir chacun une des trois couleurs.

Scatter plot 2

import matplotlib.pyplot as plt plt.scatter(titanic['Age'], titanic['Fare'], c=titanic['Pclass'])
plt.xlabel('Age')
plt.ylabel('Fare')
plt.show()



Les points violets sont de première classe, les points verts de deuxième classe et les points jaunes de troisième classe.

Exercice d'application 17

Construire un cross-plot

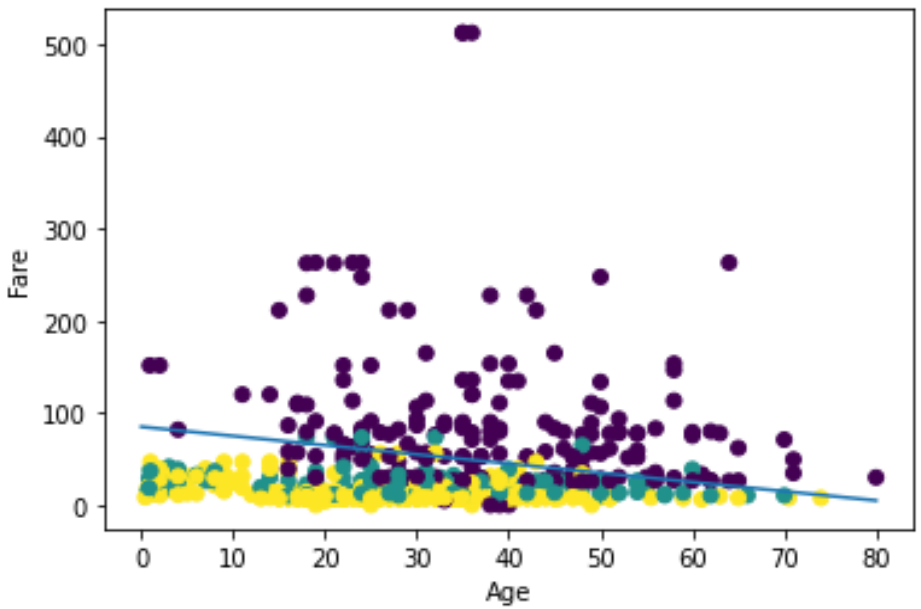
Écrivez un programme pour créer un cross-plot avec Pclass sur l'axe des y et Fare sur l'axe des x. Attribuez un code de couleur selon qu'ils ont survécu ou non. Ajoutez les étiquettes "Fare" et "Pclass" sur les axes x et y respectivement.

Tracé d'une droite 1

La fonction plot() permet de tracer **une droite à partir de ces coordonnées**.L'exemple suivant trace une ligne qui sépare approximativement la première classe de la deuxième et de la troisième classe. Á vue d'oeil, nous placerons la droite de (0, 85) à (80, 5). Notre syntaxe ci-dessous comporte une liste de valeurs x et une liste de valeurs y :

```
plt.scatter(titanic['Age'], titanic['Fare'], c=titanic['Pclass'])
plt.xlabel('Age')
plt.ylabel('Fare')
plt.plot([0, 80], [85, 5])
plt.show()
```

Tracé d'une droite 2



Vous pouvez voir que les points jaunes (3ème classe) et verts (2ème classe) sont principalement en dessous de la ligne et que les points violets (1ère classe) sont principalement au-dessus. Nous avons fait cela manuellement, mais dans le prochain module, nous apprendrons à le faire de manière algorithmique.

Exercice d'application 18

Construire une droite

Reprendre le graphique précédent en traçant cette fois-ci une droite qui va de (0, 15) à (100, 15).

Quiz 1

1) Quel module permet de manipuler et lire les données avec un main data object de DataFrame ?

1. Matplotlib
2. Pandas
3. Numpy

2) Quel module permet de réaliser des calculs et des analyses numériques avec un main data object d' array ?

1. Matplotlib
2. Pandas
3. Numpy

Quiz 2

3) Quelle est la médiane et la moyenne de la Serie suivante ?

5,10, 15, 20, 25

1. mean = 20 / median = 10
2. mean = 15 / median = 10
3. mean = 15 / median = 15

4) Quelle est la mesure de la dispersion des données ?

1. Mean
2. Median
3. Standard deviation
4. Variance

Quiz 3

5) Chargez et affichez les données

Soit un fichier csv appelé eleves.csv avec trois colonnes : nom, prenom et age.

Écrire un programme qui permet de charger le fichier csv comme un Pandas DataFrame et afficher uniquementla Serie appelée 'nom'.

6) Quelle est l'instruction qui permet de récupérer les colonnes age et prenom comme un Numpy array ?

1. data['prenom', 'age']
2. data['prenom', 'age'].values
3. data[['prenom', 'age']]
4. data[['prenom', 'age']].values

Quiz 4

7) Construction d'un cross-plot

Soit un fichier csv appelé eleves.csv avec trois colonnes : nom, taille et age.

Complétez le programme suivant permettant de construire un cross-plot avec la taille sur l'axe des abscisses et l'age sur l'axe des ordonnées :

import matplotlib as plt
..... (data['taille'], data['age'])