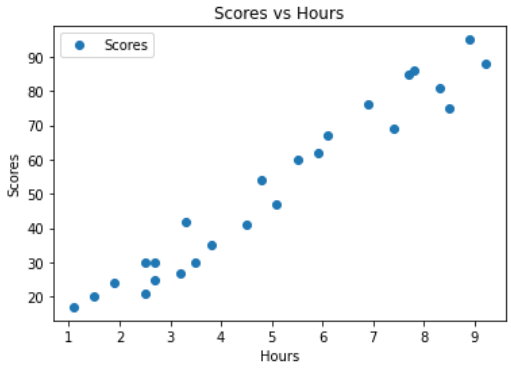
Travaux Pratique 2

Exercice 1 : Régression linéaire simple

Entraînez une régression linéaire et prédisez le score des élèves en utilisant une seule caractéristique. Dans cet exercice, vous utilisez le jeu de données *« student_scores »* afin de prédire le score d'un étudiant en fonction du nombre d'heures étudiés.
Le graphe ci-dessous montre qu'il y a une relation linéaire positive entre le score et le nombre d'heures. Donc, une régression linéaire devrait être capable de saisir cette relation.



Votre travail consiste à entrainer une régression linéaire, puis de visualiser graphiquement la ligne de régression.

1. Charger le fichier « student_scores.csv » (voir Moodle) dans un DataFrame appelé « score » à l'aide de la fonction ***read_csv()***
2. Explorer le DataFrame « score » à l'aide des méthodes pandas : ***info(), describe()*** et ***isnull()***
3. Entrainer un modèle de régression linaire sur l'ensemble de données.
4. La relation linéaire entre la variable « Scores » et l'attribut « Hours » est exprimée par l'équation suivante : $Scores = a * Hours + b$. Calculer la valeur de b et a.
5. Tracer le nuage de points de données « Hours » sur l'axe x et « Scores » sur l'axe des y en utilisant la fonction **plt.scatter()**
6. Superposez sur le tracé du nuage de points, la ligne de régression linéaire en utilisant la fonction **plt.plot()**.

Exercice 2 : Régression linéaire multiple

Dans cet exercice, vous allez travailler avec les données de la base « gas consumptions » afin de prédire les consommations d'essence en fonction de différents attributs : taxes sur l'essence, le revenu par habitat, des autoroutes pavées et la proportion de population possédant un permis de conduire. Cette base est disponible dans le fichier csv « petrol_consumption.csv » (voir Moodle).
Votre travail consiste à entrainer une régression linéaire multiple, puis à prédire le score et d'évaluer les performances du modèle.

1. Charger le fichier « petrol_consumption.csv » dans un DataFrame appelé « df » à l'aide de la fonction **read_csv()**
2. Explorer le DataFrame « score » à l'aide des méthodes pandas : **info(), describe()** et **isnull()**
7. Créer des ensembles de données d'entrainement et de test de sorte que 20% soient utilisés pour le test et 80 % pour l'entrainement. Utiliser un état aléatoire **random_state** égal à 42.
8. Entrainer un modèle de régression linéaire avec l'ensemble d'entrainement.
9. Prédire les valeurs de l'ensemble de test.
10. Calculer et afficher le score en utilisant la méthode « score » du modèle sur l'ensemble de test.
11. Calculer et afficher l'erreur RMSE. Pour le faire, commencez par calculer l'erreur quadratique moyenne à l'aide de la fonction **mean_squared_error()**, puis calculez sa racine carrée à l'aide **np.sqrt().**

Exercice 3 : Régression linéaire multiple avec la validation croisée

La validation croisée est une étape essentielle dans l'évaluation d'un modèle. Elle maximise la quantité de données utilisées pour entraîner le modèle. En effet, cela permet, au cours de l'apprentissage, d'entraîner le modèle et de le tester sur toutes les données disponibles.
Dans cet exercice, vous allez effectuer une validation croisée sur les données « score students ». Par défaut, la fonction **cross_val_score()** de *scikit-learn* utilise **R^2** comme métrique pour la régression (**R^2** est le coefficient de détermination, c'est l'accuracy utilisée pour une régression). Puisque vous effectuez une validation croisée 3 fois (3 folds), alors la fonction renverra 3 scores. Votre travail consiste à calculer ces 3 scores, ensuite de calculer leur moyenne.

1. Créez un modèle de régression linéaire appelé « reg ».
2. Utilisez la fonction **cross_val_score()** pour effectuer une validation croisée 3 folds sur « X » et « y ».
3. Calculez et afficher le score moyen de la validation croisée. Vous pouvez utiliser la fonction **mean()** de *NumPy* pour calculer la moyenne.

Exercice 4 : Régression polynomiale

La régression polynomiale est une approche statistique qui est employée pour modéliser une relation de forme non-linéaire entre la variable cible (y) et la ou les variables explicatives (x). L'idée de la régression polynomiale sera d'écrire cette relation comme suit :
$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ *(où n est le degré du polynôme)*

Dans cet exercice, vous allez entrainer une régression polynomiale sur les données « WeatherData » afin de prédire l'humidité en fonction de la pression. Les données sont disponibles dans le fichier « WeatherData.csv » (voir Moodle).

1. Charger le fichier « WeatherData.csv » dans un DataFrame appelé « weather » à l'aide de la fonction **read_csv().**
2. Explorer le DataFrame « weather » à l'aide des méthodes pandas : **info(), describe()** et **isnull()**

3. Créer le tableau X pour l'attribut « Pressure » et le tableau « y » pour la variable cible « Humidity ».
4. Entrainer un modèle de régression polynomiale avec l'ensemble de données X et y. En effet, il n'existe pas dans la librairie d'algorithmes d'apprentissage dédiés à la régression polynomiale, mais uniquement ceux de régression linéaire. Scikit propose cependant un préprocesseur permettant de calculer toutes les variables de degré supérieur. De cette façon, une fois le calcul est fait, les algorithmes de régression linéaire sont directement utilisables.
   Le préprocesseur est **PolynomialFeatures**. Son principal paramètre est **degree**, indiquant le degré maximal pour les calculs.
   Donc, pour le faire, il faut créer une instance de **PolynomialFeatures** avec **degree = 2**, puis transformer les données X à l'aide de la fonction **fit_transform().** Enfin, créer un modèle de régression linéaire appelé « poly_reg » à l'aide de la méthode **LinearRegression()**.
5. Prédire la valeur d'humidité quand la pression égale à 1007.
6. La fonction de régression polynomiale pour cet exemple est écrite comme suit :

$$y = a_0 + a_1 x + a_2 x^2$$

   Calculer la valeur de a_0, $a_1$et a_2

12. Tracer le nuage de points de données « Pressure » sur l'axe x et « Humidity » sur l'axe des y en utilisant la fonction **plt.scatter().**
13. Superposez sur le tracé du nuage de points, la ligne de régression (en rouge) en utilisant la fonction **plt.plot()**.
14. Calculer et afficher l'erreur RMSE. Pour le faire, commencez par calculer l'erreur quadratique moyenne à l'aide de la fonction **mean_squared_error()**, puis calculez sa racine carrée à l'aide **np.sqrt().**