M1 Informatique AMU – TP3 ISD — 2023/24 — Rapport. **Date : 07 novembre 2023.**
Nom, prénom [+éventuel binôme], groupe TP# 1

RÉGRESSION LINÉAIRE PAR MOINDRES CARRÉS & SKLEARN

Partie : Etude d'un régresseur simple

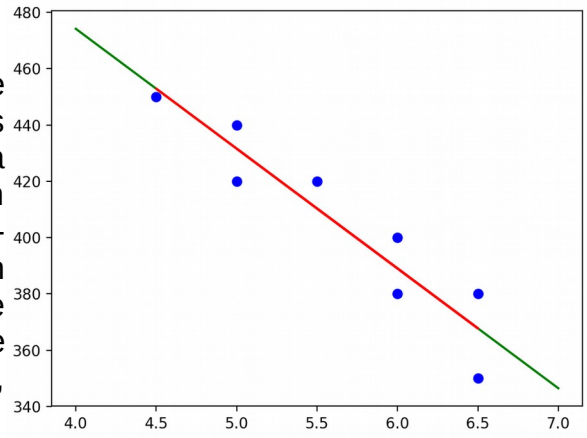
Code. Tout le code de cette partie est disponible dans le fichier partie1_TP3.py.

On crée le jeu de données sous forme de tableaux : un tableau de descriptions (X) à une seule dimension, et un tableau de cibles (y) forcément à une seule dimension aussi, les deux sur 8 exemples. On affiche le score (qui est le coefficient de détermination de Pearson R^2 dans le cas de la régression linéaire), et la MSE (Mean Square Error) en utilisant les fonctions associées à la classe LinearRegression de SkLearn, avec comme jeu de données celui d'apprentissage. On affiche aussi les paramètres calculés lors de l'apprentissage sur ce petit jeu de données, à savoir coef_ (les coefficients de la droite de régression) et intercept_ (le biais – écart à l'origine). On obtient l'affichage :

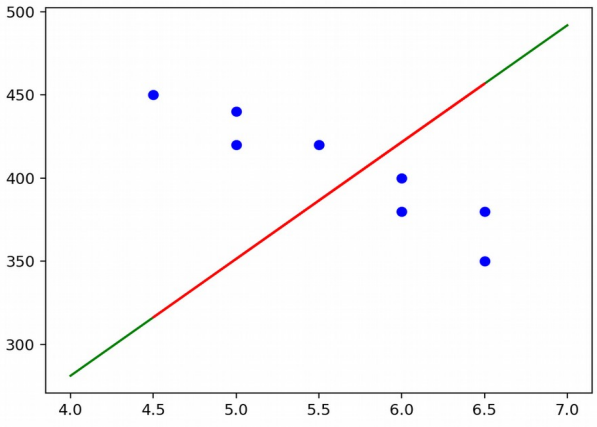
```
MSE = 121.7741935483871
Score = 0.8782258064516129
Paramètres = [-42.58064516] 644.5161290322579
```

Ainsi, la droite apprise sur ces données a pour équation (arrondie) *y = -42.6x + 644.5*
Le score (Pearson) est proche de 1, donc plutôt très bon.

L'affichage du nuage de points, de la droite reconstituée en rouge par utilisation explicite des paramètres appris (pour calculer deux points de la droite et donc afficher la droite), et de la droite en vert telle que stockée par SkLearn, est indiqué ci-contre. Les droites rouge et verte coïncident, et on constate que la solution semble bonne, ce que confirme le bon score. Au regard de l'échelle de grandeur des données, la MSE semble petite (1/4), donc bonne.



Avec fit_intercept_ à False comme hyper-paramètre du régresseur par moindre carrés, on force la solution à passer par l'origine. De ce fait, les scores deviennent mauvais (score de détermination négatif et grande MSE au regard de l'échelle des données : *90). Visuellement, on constate qu'effectivement en imposant intercept_ à être nul, la solution apprise perd en pertinence.

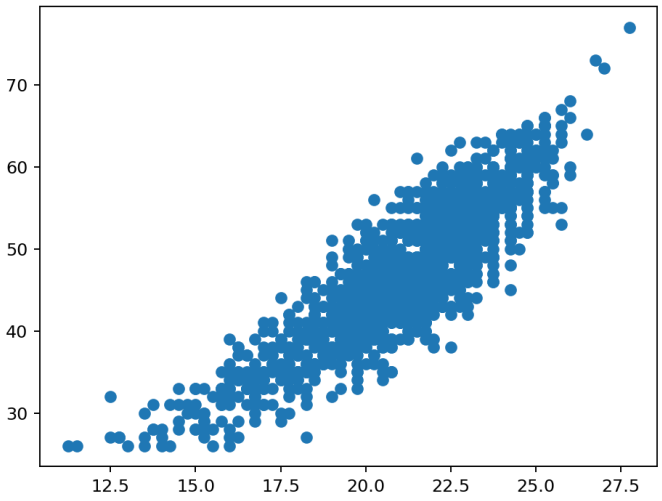



```
MSE = 6385.116731517507
Score = -5.385116731517507
```

Partie : Régression et jeu de données Eucalyptus

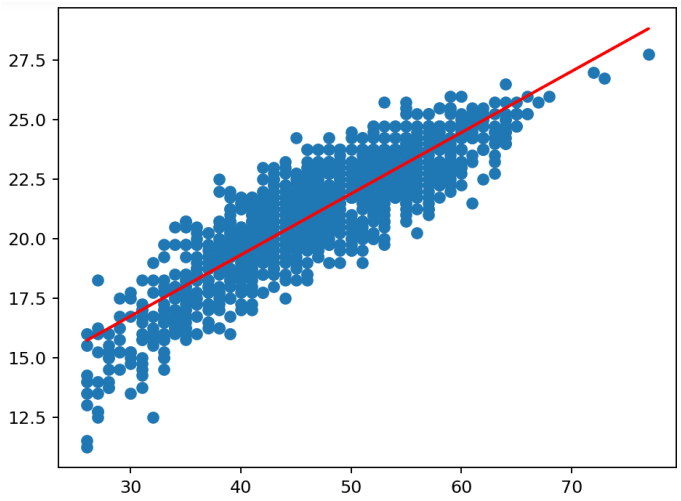
Code. Tout le code de cette partie est disponible dans le fichier partie2_TP3.py.

Le nuage de points est donné ci-contre, avec en abscisse la variable explicative (circonférence) et en ordonnée la cible (hauteur), donc en changeant l'ordre des colonnes telles que stockées dans le jeu de données (*NB : j'aurais amélioré mon rendu en affichant les légendes*). On constate visuellement qu'il semble y avoir une corrélation linéaire positive, même si le nuage est « épais » le long d'une droite, et même si le nuage semble un peu courbe.



En apprenant la droite de régression via la classe SkLearn LinearRegression, on obtient le modèle en rouge ci-contre (sur un des apprentissages), et le score de cette méthode – coefficient de détermination – est de 0.70 tel qu'estimé par validation croisée (fonction cross_val_score) 10 folds.

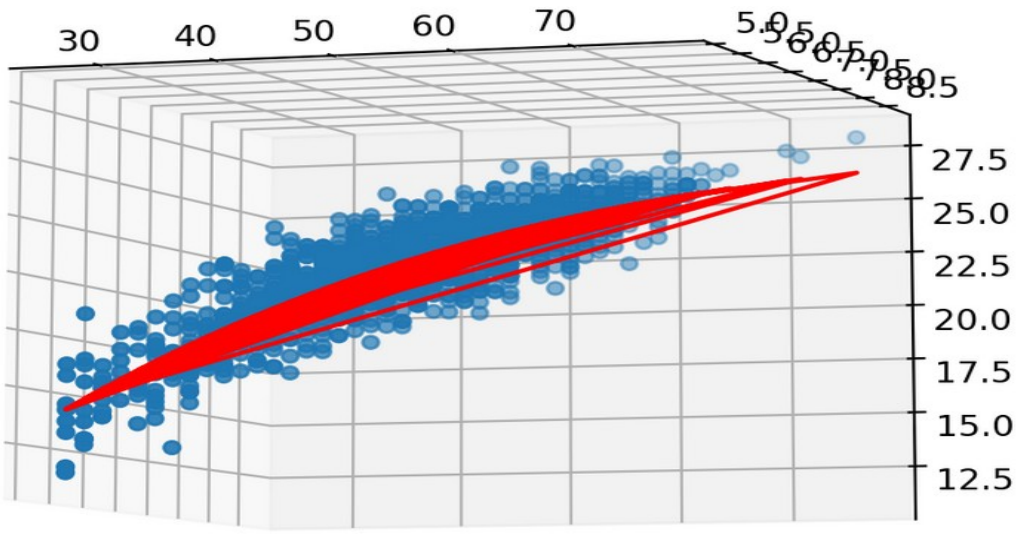
Pour une circonférence de 22.8, on prédit – avec le modèle visualisé – une hauteur prédite de 14.9 ce qui semble visuellement peu cohérent.



Parce que nous pensons trouver un meilleur régresseur (par intégration de cette courbure apparente sur le nuage de points), nous ajoutons une variable supplémentaire calculée à partir de la circonférence, qui porte une version non linéaire de la circonférence.

On choisit d'introduire une racine carré, car la courbe du nuage de point « ressemble » à la courbe de la racine carrée (tout au moins sur les points du dessus du nuage).

Nous obtenons les paramètres suivant (description en 2D maintenant) : Paramètres = [-0.48294547 9.98688814] -24.35200327424318, et un score (coeff de détermination Pearson) légèrement meilleur, à 0.73 (vs 0.70 avec 1D). On visualise (mal) le plan séparateur en rouge (en profondeur, la dimension ajoutée).



Forts de cette amélioration, nous tentons d'autres introductions de non-linéarité via plus de variables supplémentaires (donc nous perdons la visualisation). Par exemple, nous ajoutons une troisième variable explicative égale au carré de la circonférence. Nous améliorons encore un peu le score (0.75). Nous avons testé aussi des logarithmes, des polynômes, etc., et c'est le carré qui a remporté la meilleure amélioration. Lorsque nous aurons le temps, nous introduirons une quatrième variable, mais ne pouvons pas en ajouter trop au regard du fait que le nombre d'exemples reste fixe alors que la complexité du modèle augmente à chaque ajout de variable.

Conclusion. Ce TP a permis de comprendre en pratique la régression linéaire sur des jeux de données, que l'on peut visualiser. Nous avons vu que le biais (intercept_) peut fortement influencer le modèle (ici une droite) appris.
Sur la fin, nous avons introduit de nouvelles variables qui permettent d'introduire des informations calculées sur celles disponibles, introduisant de la non-linéarité : nous avons gagné en performance, mais les modèles ne sont plus visualisables : seuls les scores d'évaluation des modèles permettent de mesurer les améliorations.