SIMPLON.CO

Projet N°6
Janvier 2022

RAPPORT
Premier Modèle IA

(Régression linéaire simple, multiple et polynomiale)

Développer un modèle de Régression linéaire simple, multiple et polynomiale sans utiliser les modèles prédéfinis de Python.

Auteurs : **Maïna LE DEM**
Sylvia PENFEUNTEUN

Indice : A
Date d'émission : 31/01/2022
Rédacteurs : M. LE DEM et S. PENFEUNTEUN
Vérificateur : M. LE DEM et S. PENFEUNTEUN

SOMMAIRE

1. Rappel sur la régression linéaire simple, multiple et polynomiale 2

1.1 - Modèle de régression linéaire simple 2
1.2 - Modèle de régression multiple 2
1.3 - Modèle de régression polynomiale 2

2. Explication de chaque fonction en présentant l'équation matricielle qui lui correspond (par la méthode normale) 3

2.1 - Régression linéaire simple 3
2.2 - Régression multiple 3
2.3 - Régression polynomiale 4

3. Présentation des résultats des modèles sous forme de graphiques 5

3.1 - Régression linéaire simple 5
3.2 - Régression multiple 5
3.3 - Régression polynomiale 6

4. Evaluation des modèles 7

4.1 - Régression linéaire simple 7
4.2 - Régression multiple 7
4.3 - Régression polynomiale 8

5. Présentation des résultats avec le module Scikit-Learn 9

5.1 - Régression linéaire simple 9
5.2 - Régression multiple 9
5.3 - Régression polynomiale 10

6. Comparaison avec la méthode normale 11

6.1 - Régression linéaire simple 11
6.2 - Régression multiple 11
6.3 - Régression polynomiale 11

7. Conclusion 12

1. Rappel sur la régression linéaire simple, multiple et polynomiale

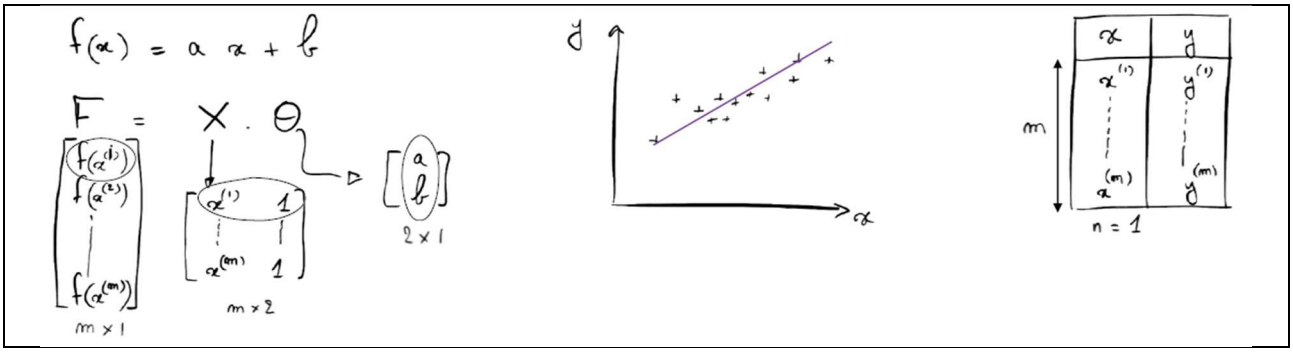
Le Machine Learning, c'est donner à une machine la capacité d'apprendre sans la programmer de façon explicite. C'est de l'apprentissage supervisé.

On calcul un algorithme, où on cherche à trouver F, telle que y = F(x)

Selon les données du dataset, il existe 3 types d'algorithmes : la régression linéaire simple, la régression linéaire multiple et la régression polynomiale.

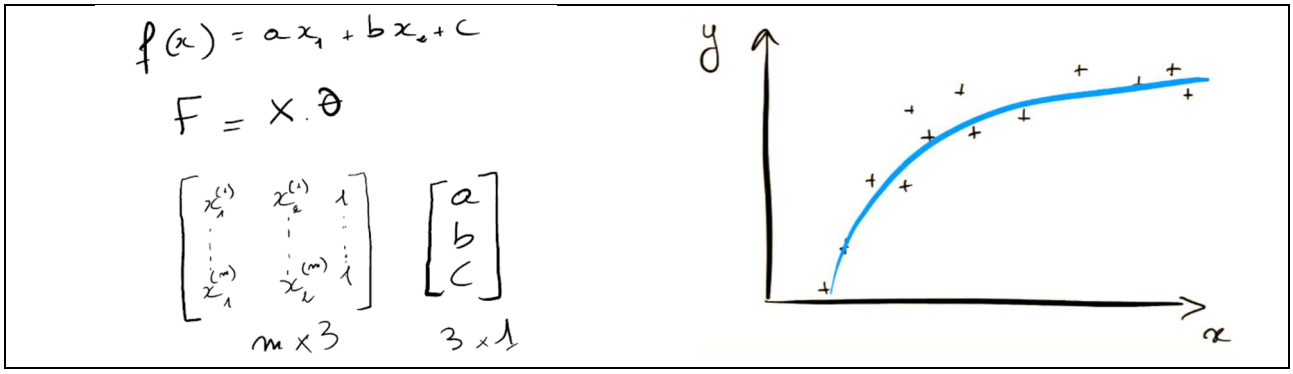
1.1 - *Modèle de régression linéaire simple*

Lorsque le dataset compte 1 feature x, le modèle est linéaire : Fonction affine



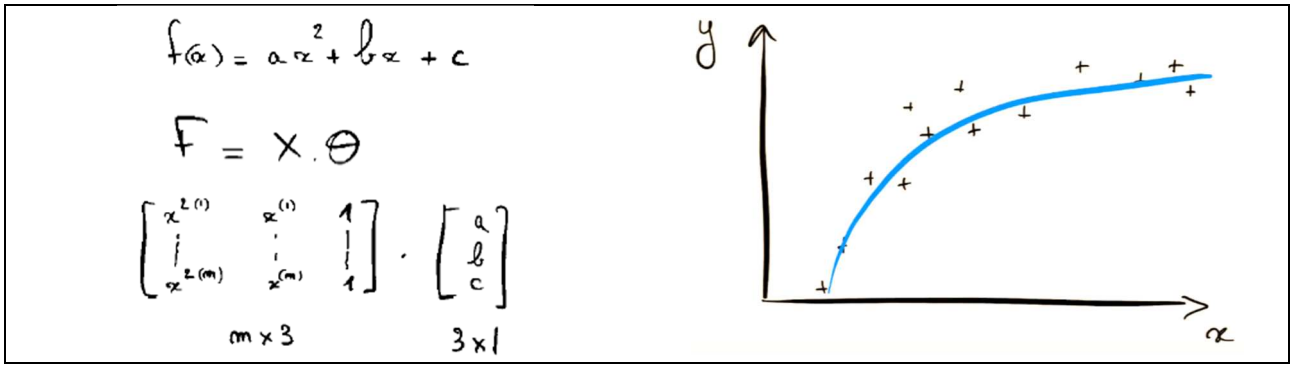
1.2 - *Modèle de régression multiple*

Lorsque le dataset compte n features x, le modèle est linéaire multiple : Fonction sigmoïde



1.3 - *Modèle de régression polynomiale*

Lorsque le dataset compte 1 feature x, et la courbe x/y est de forme sigmoïde, on dévolppe modèle est polynomiale : Fonction sigmoïde de n degrés



2. Explication de chaque fonction en présentant l'équation matricielle qui lui correspond (par la méthode normale)

2.1 - *Régression linéaire simple*

Les 4 étapes du calcul :

1) **Le dataset :** sa dimension, le nombre de lignes, et le nombre de features (x, y) avec m exemples

Matrices X et y :

$$X = \begin{bmatrix} x^{(1)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

$m \times (n+1)$ $\quad$ $m \times 1$

Theta :

$$\theta = \begin{bmatrix} a \\ b \end{bmatrix}$$

$(n+1) \times 1$

2) **Le modèle** qu'on veut développer :

F(x) = ax + b , soit : $F = X \cdot \Theta$

3) **La fonction coût :** erreur quadratique moyen

$$J(\theta) = \frac{1}{2m} \sum (X.\theta - Y)^2$$

4) **Algorithme de minimisation**

Calcul du gradient :

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T.(X.\theta - Y)$$

Descente de gradient :

$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

5) **Coefficient de détermination :** C'est le coefficient r² (Moindre carré), pour montrer la réelle performance de notre modèle.
Plus il est proche de 1 plus notre modèle rentre dans notre nuage de point

$$R^2 = 1 - \frac{\sum (y - f(x))^2}{\sum (y - \bar{y})^2}$$

2.2 - *Régression multiple*

Le calcul identique à l'algorithme de régression linéaire, à

1) **Le dataset :** sa dimension, le nombre de lignes, et le nombre de features (x, y) avec m exemples

Matrices X et y :

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & 1 \\ \dots & \dots & \dots \\ x_1^{(m)} & x_2^{(m)} & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

Theta :

$$\theta = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

2) **Le modèle qu'on veut développer :**

F(x) = $ax_1 + bx_2 + c$, soit : $F = X . \Theta$

3) **La fonction coût :** erreur quadratique moyen

$$J(\theta) = \frac{1}{2m}\sum(X.\theta - Y)^2$$

4) **Algorithme de minimisation**

Calcul du gradient :

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}X^T.(X.\theta - Y)$$

Descente de gradient :

$$\theta := \theta - \alpha\frac{\partial J(\theta)}{\partial \theta}$$

5) **Coefficient de détermination :** C'est le coefficient r^2 (Moindre carré), pour montrer la réelle performance de notre modèle.
Plus il est proche de 1 plus notre modèle rentre dans notre nuage de point

$$R^2 = 1 - \frac{\sum(y - f(x))^2}{\sum(y - \bar{y})^2}$$

2.3 - *Régression polynomiale*

Le calcul identique à l'algorithme de régression linéaire, à

1) **Le dataset :** sa dimension, le nombre de lignes, et le nombre de features
(x, y) avec m exemples

Matrice X et y :

$$X = \begin{bmatrix} x^{2(1)} & x^{(1)} & 1 \\ \dots & \dots & \dots \\ x^{2(m)} & x^{(m)} & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

$m \times 1$

Theta :

$$\theta = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

2) **Le modèle qu'on veut développer :**

F(x) = $ax^2 + bx + c$, soit : $F = X . \Theta$

3) **La fonction coût :** erreur quadratique moyen

$$J(\theta) = \frac{1}{2m}\sum(X.\theta - Y)^2$$

4) **Algorithme de minimisation**

Calcul du gradient :

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}X^T.(X.\theta - Y)$$

Descente de gradient :

$$\theta := \theta - \alpha\frac{\partial J(\theta)}{\partial \theta}$$

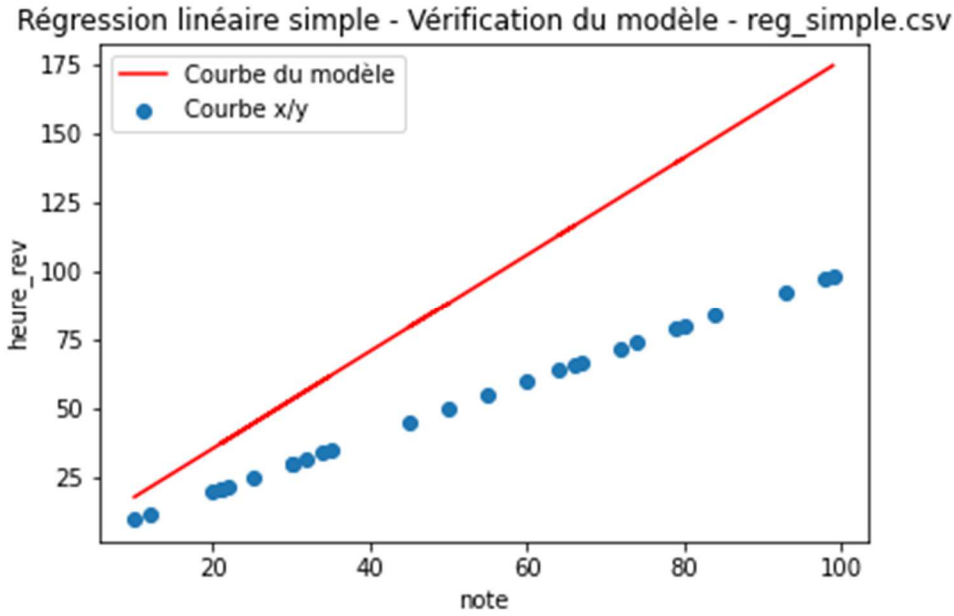
5) **Coefficient de détermination :** C'est le coefficient r² (Moindre carré), pour montrer la réelle performance de notre modèle.
Plus il est proche de 1 plus notre modèle rentre dans notre nuage de point

$$R^2 = 1 - \frac{\sum(y - f(x))^2}{\sum(y - \bar{y})^2}$$

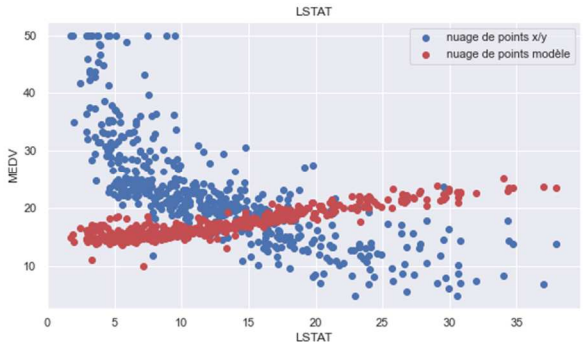
3. Présentation des résultats des modèles sous forme de graphiques

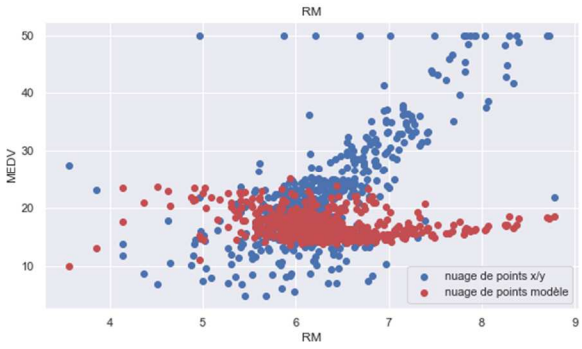
Présentation des différents modèles avant le calcul du gradient, de la descente de gradient, et de l'évaluation du modèle.

3.1 - *Régression linéaire simple*



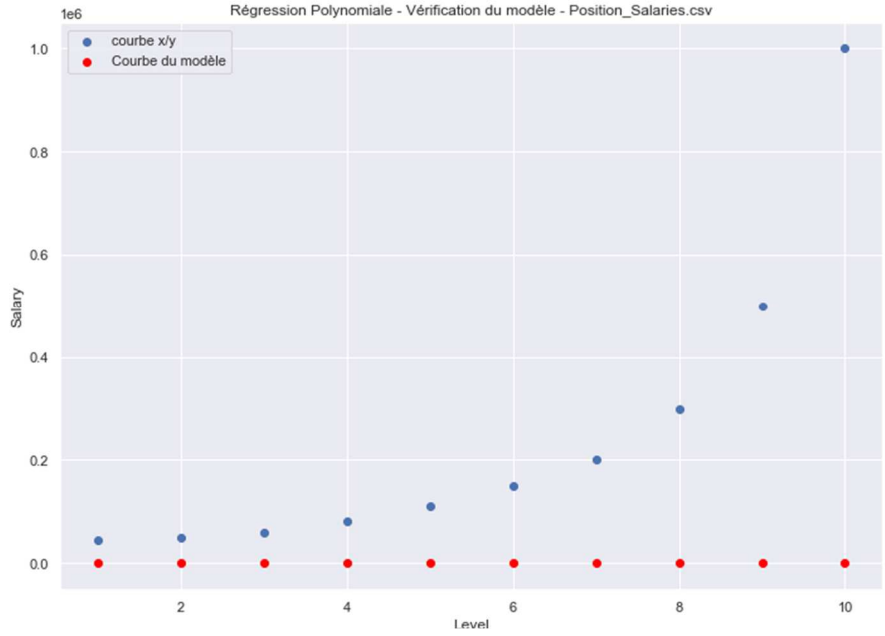
3.2 - *Régression multiple*



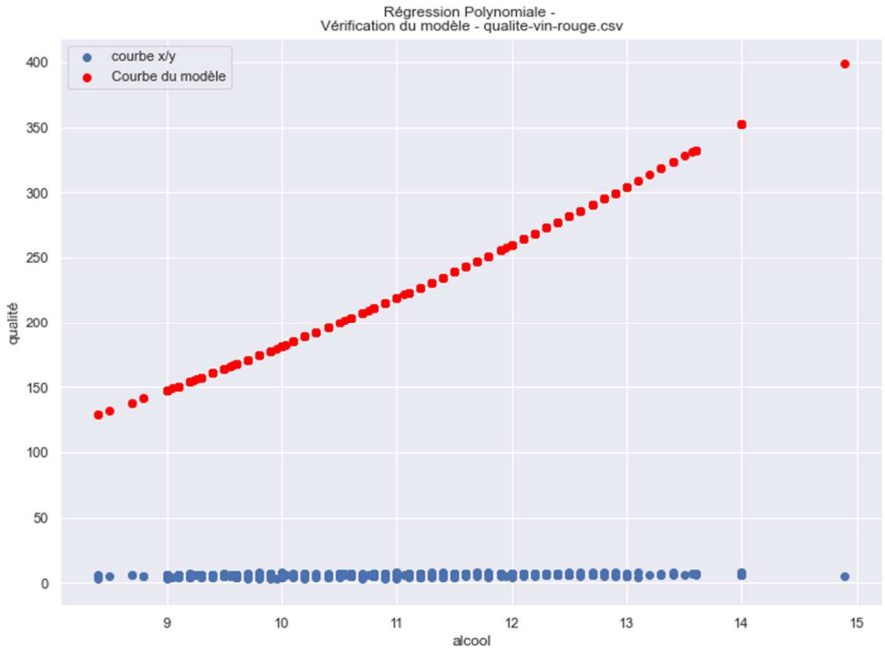


3.3 - *Régression polynomiale*

3.3.a. Position salaries



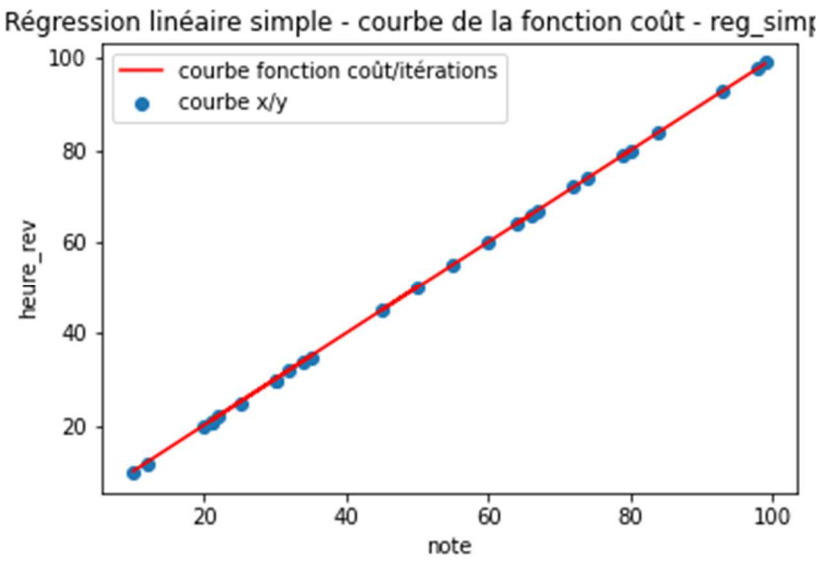
3.3.b. Qualité vin rouge



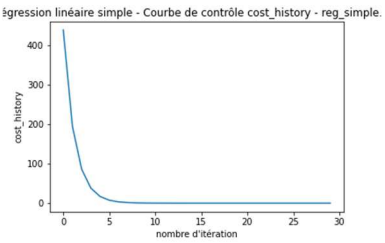
4. Evaluation des modèles

4.1 - *Régression linéaire simple*

Traçage de la courbe de la fonction du coût selon les itérations

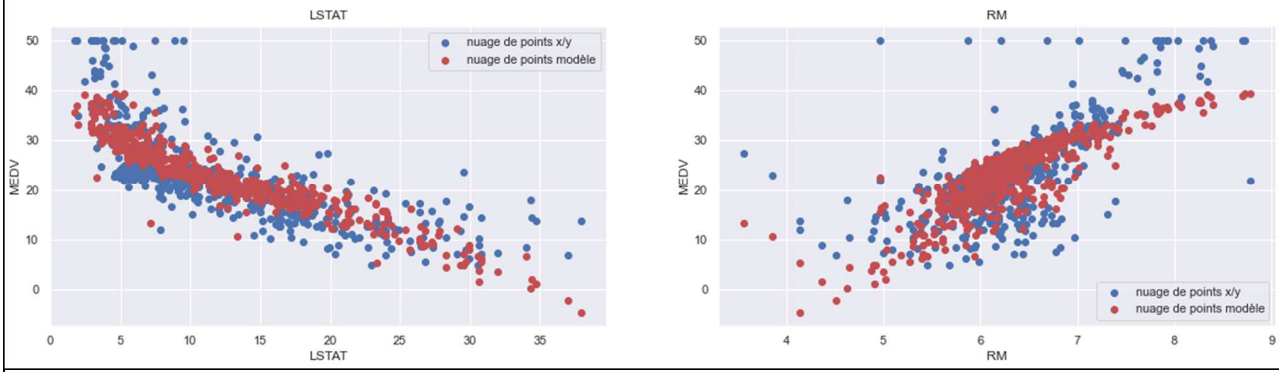


Courbe de contrôle cost_history

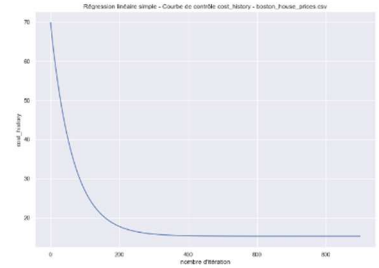


4.2 - *Régression multiple*

Traçage de la courbe de la fonction du coût selon les itérations

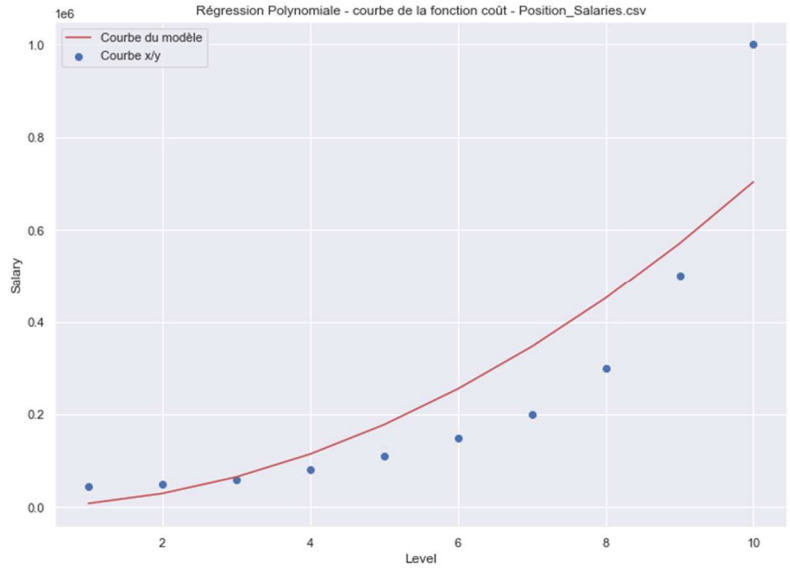
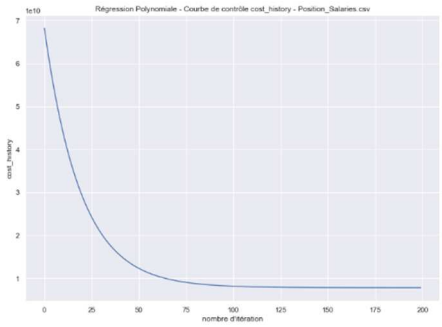


Courbe de contrôle cost_history

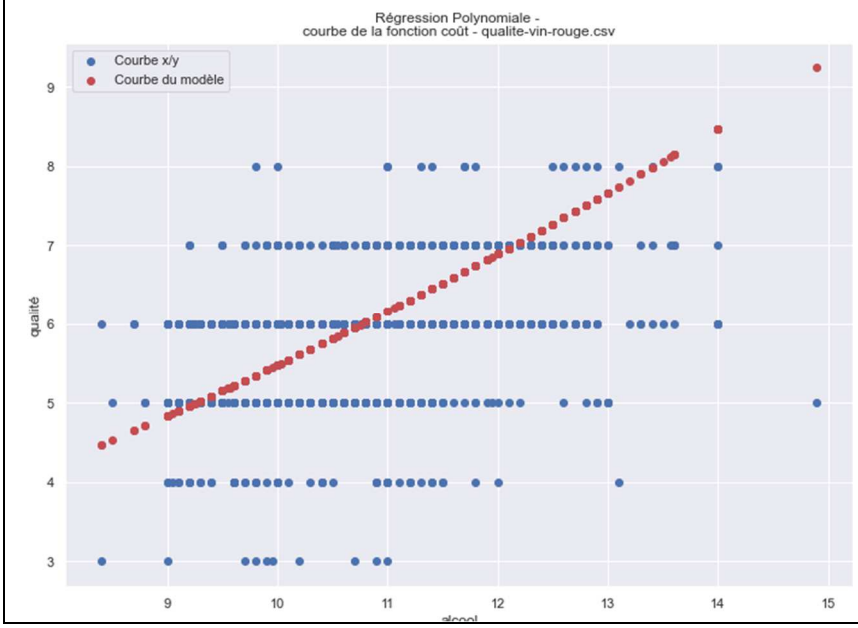
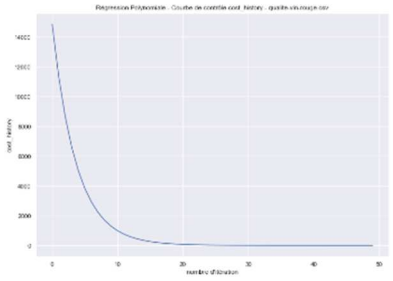


4.3 - *Régression polynomiale*

4.3.a. Position salaries

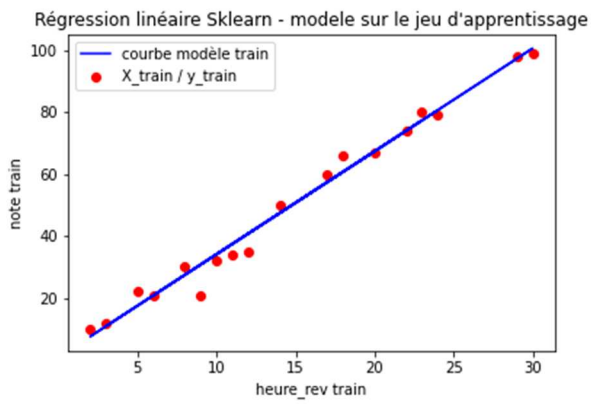
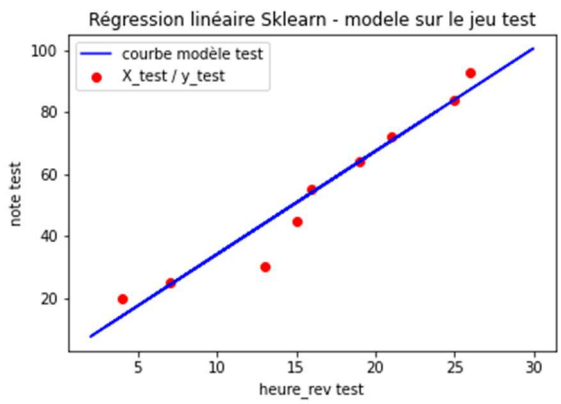
Traçage de la courbe de la fonction du coût selon les itérations	Courbe de contrôle cost_history
	

4.3.b. Qualité vin rouge

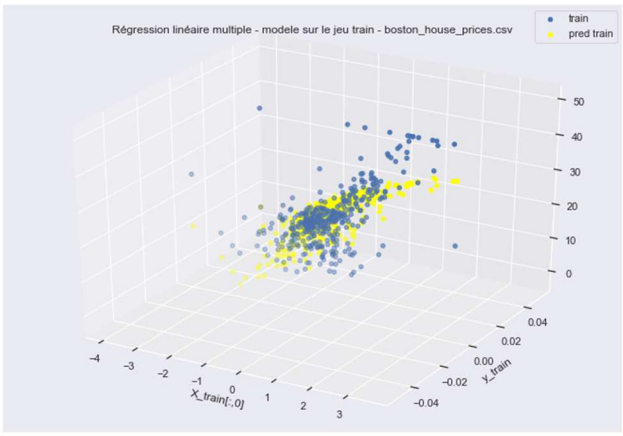
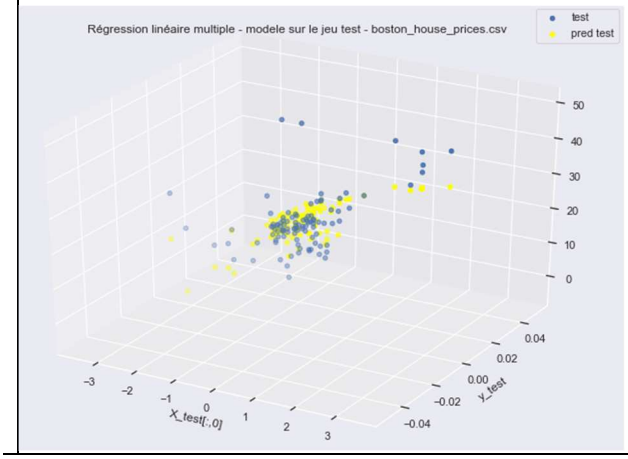
Traçage de la courbe de la fonction du coût selon les itérations	Courbe de contrôle cost_history
	

5. Présentation des résultats avec le module Scikit-Learn

5.1 - *Régression linéaire simple*

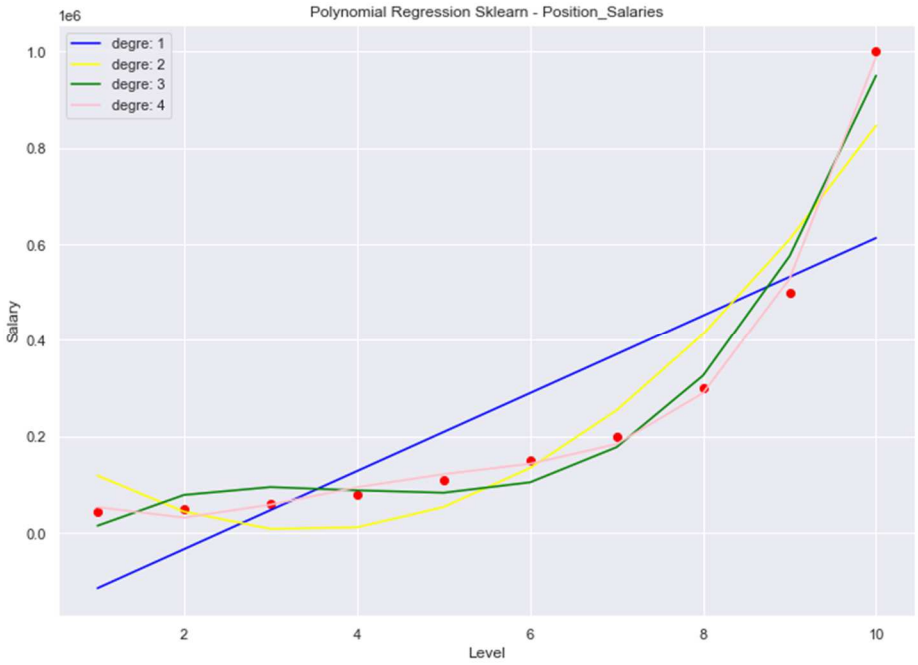
Modèle sur le jeu d'apprentissage	Contrôle sur le jeu test
	

5.2 - *Régression multiple*

Modèle sur le jeu d'apprentissage	Contrôle sur le jeu test
	

5.3 - *Régression polynomiale*

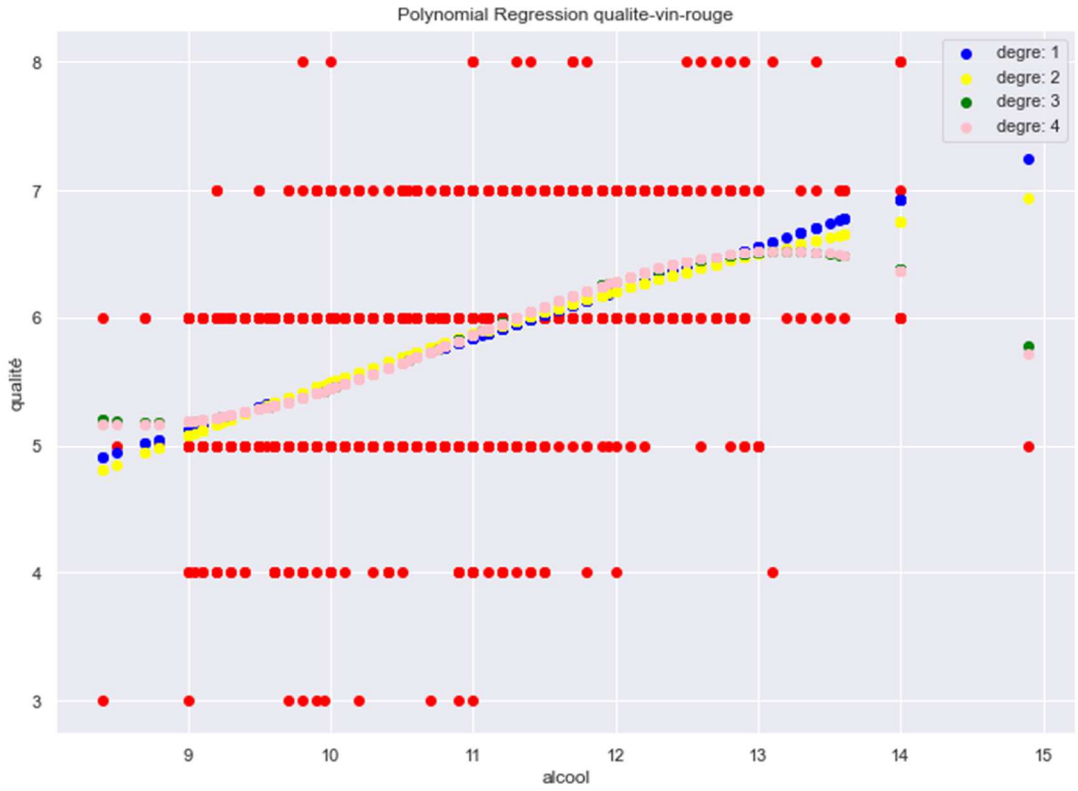
5.3.a. Position salaries



Recherche du meilleur modèle suivant le degré de la fonction.

On peut observer que la fonction de degré 4, s'approche au mieux du jeu de données

5.3.b. Qualité vin rouge



Recherche du meilleur modèle suivant le degré de la fonction.

On observe qu'aucun modèle ne correspond à notre dataset.

6. Comparaison avec la méthode normale

Comparaison de la méthode normale avec la méthode Scikit Learn, en observant les résultats des mean_squared_error (mse) et le coefficient R^2 (Moindre carré), pour montrer la réelle performance de notre modèle

6.1 - *Régression linéaire simple*

	Méthode normale *(y, y_pred)*	**Scikit Learn** *(y_test, y_pred)*
mse *(y, pred)*	141.364	33.4692
R^2 *(y, pred)*	0.80928	0.94522

On observe un meilleur résultat avec Scikit Learn

6.2 - *Régression multiple*

	Méthode normale *(y, y_pred)*	**Scikit Learn**	
		(y_train, y_pred_train)	*(y_test, y_pred)*
mse *(y, pred)*	30.5573	28.79027	37.38310
R^2 *(y, pred)*	0.63803	0.6618	0.54090

On observe un résultat légèrement meilleur avec Scikit Learn

6.3 - *Régression polynomiale*

6.3.a. Position salaries

	Méthode normale *(y, pred)*	**Scikit Learn** *(y, pred)*			
		degré 1	degré 2	degré 3	degré 4
mse *(y, pred)*	15560871021.77	26695878787.87	6758833333.33	1515662004.66	210343822.84
R^2 *(y, pred)*	0.540908	0.66904	0.9162082	0.9812097	0.9973922

Nous obtenons un meilleur résultat avec la méthode de Scikit Learn

En essayant plusieurs degrés à la fonction, on arrive à un meilleur résultat lorsqu'on applique une fonction de degré 4.

6.3.b. Qualité vin rouge

	Méthode normale *(y, pred)*	**Scikit Learn** *(y, pred)*			
		degré 1	degré 2	degré 3	degré 4
mse *(y, pred)*	0.660775	0.5039840	0.5031885	0.4991603	0.499151
R^2 *(y, pred)*	0.54090	0.226734	0.2279548	0.2341354	0.2341495

On observe un résultat encore moins bon avec Scikit Learn, cependant la "qualité" du vin est une valeur appréciative et non quantitative.

Pour ce type de cas, il serait judicieux de faire une classification ou une régression logistique.

7. Conclusion

Nous avons appris :

- les bases du Machine Learning
- à calculer une régression linéaire, multiple et polynomiale de façon manuelle, grâce aux vidéos de Machine Learnia de Guillaume Saint-Cirgue
- appliquer Scikit Learn pour réaliser ces mêmes régressions.

Les difficultés :

- Comprendre ce qu'il était attendu d'obtenir pour ma part (Sylvia)
- Comprendre à interpréter certains résultats. Par exemple : le mean_squared_error et de son utilité
- Faire les régressions avec Scikit Learn (multiples et polynomiales), sans le corrigé du précédent TP

Après ce projet, je me sens :

- Contente d'avoir enfin fini le notebook et le rapport