## Université of Science et Technologies HOUARI BOUMEDIENE Département d'informatique



APPRENTISSAGE AUTOMATIQUE

ET RESEAUX DE NEURONNES

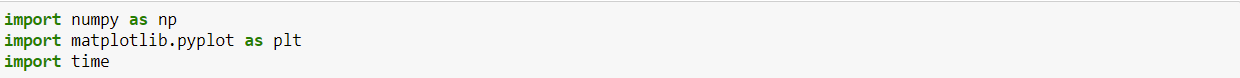
# **« RAPPORT TP 3»**

|  |  |  |  |
| --- | --- | --- | --- |
| *NOM* | *PRENOM* | *MATRICULLE* | *GROUPE* |
| ***BELIMI*** | ***Ibrahim Sabri*** | ***161631074255*** | ***A1 TP1*** |
| ***ARAB*** | ***MAHER*** | ***171731045353*** | ***A1 TP1*** |
| ***ZAIT*** | ***Fouad*** | ***181831072145*** | ***A1 TP4*** |
| ***ASMA*** | ***SRAOUIA*** | ***161631102642*** |  |

***OBJECTIFS***

* Dans ce TP, nous aimerions prédire l'admission d'un étudiant à une spécialité donnée selon ses notes dans deux matières (2 futures : X1 note de l’étudiant dans la matière 1, X2 note de l’étudiant dans le matière 2 ; y sera 1 ou 0 admis ou pas admis )
* Pour ce faire, nous étudierons un ensemble de données sur laquelle nous appliquerons la régression logistique .

**Importation des librairies nécessaires pour le travail**



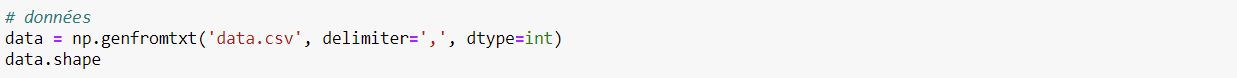
***numpy :*** pour la manipulation des tableaux (array)

***matplotlib :*** pour la manipulation des graphs (2d, 3d)

***time :*** pour calculer le temps d’exécution

**Lecture des fichiers de données pour les classifier**

**Code :**



**Résultat :**

===> 

**(colonne 1 X1: note dans le matière 1**

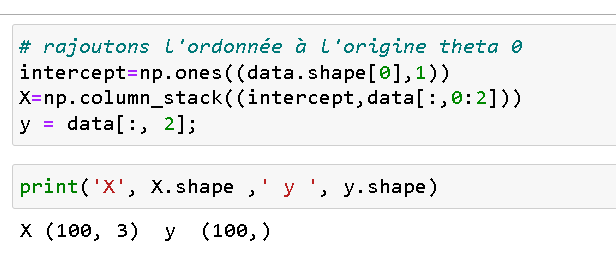
**colonne  2 X2: note dans le matière 2**

**Colonne 3 Y: admis 1 ou pas admis 0**

**)**

Dans ces données (data), la première colonne représente la première note, la deuxième colonne la deuxième note et la troisième colonne représente l'admission à la spécialité (1 admis 0 non admis).

Chaque ligne représente un exemple de notre ensemble de données.



On met dans X nos futures et dans y les valeurs prédites de X . on a 100 exemples (étudiants)

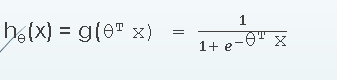
X aura 3 colonnes X0=1 , X1= note dans la matiére 1 , X2=note dans la matiére 2 .

**Fonction Sigmoid:**



Cette fonction nous peremettera de claculer la Sigmoid d’une valeur donnée , nous aurons besoin de la sigmoid pour calculer l’hypothése de la régression logistique on lui donnera en entrée quand on l’appelerra :

theta0 \*X0+theta1\*X1+theta2\*X2 pour avoir hθ(x).

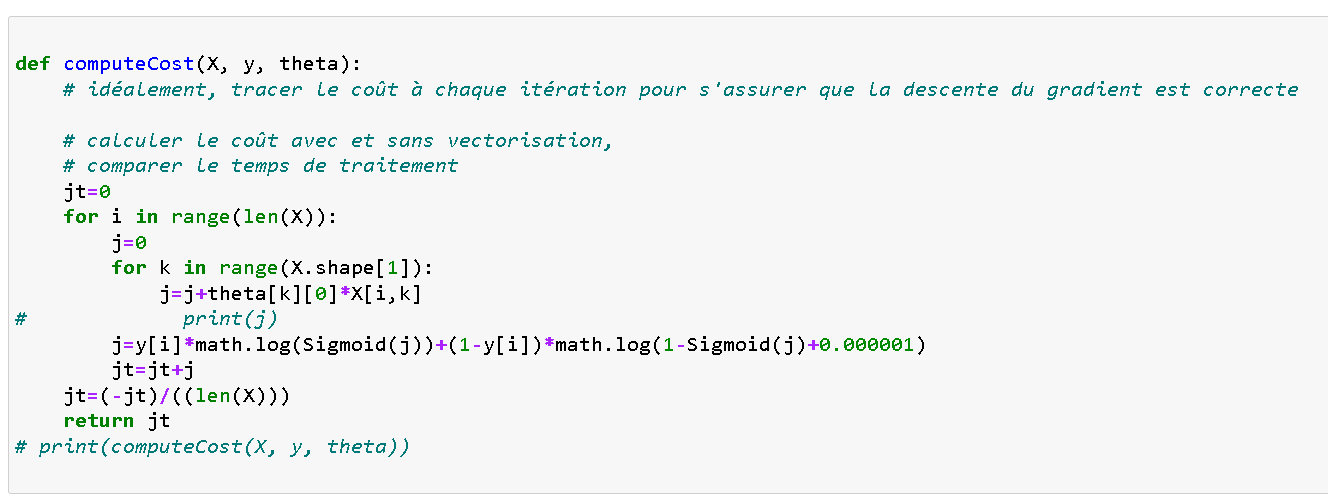


# **Descente du Gradient**

# **Préparation des fonctions:**

**1. La Fonction de Cout (sans vectorisation) :**

**Code:**



Cette fonction nous permettra de calculer le cout vu en cours de la régression logistique on a implémenté la fonction de cout ci-dessous :



Avec :

hθ(x)= θ0X0 +θ1X1 + θ2X2

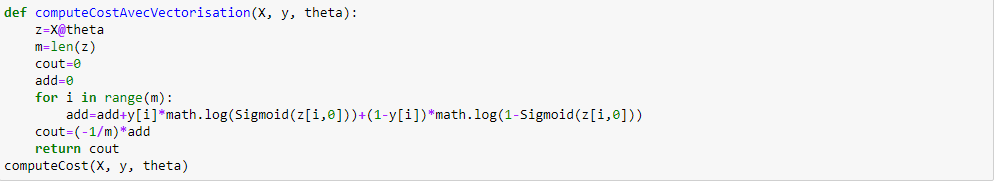
X0 =1

On a modéliser nos θ0 et θ1 et θ2 par un vecteur à 3 lignes et 1 colonne la 1ere ligne représente θ0 et la 2eme ligne représente θ1 **theta[1,0] = 1 ,**

**theta[0,0] = 0** et la 3eme ligne représente θ2 **theta[2,0] =** θ2.Nous allons d’abord parcourir tous les Xi et calculer le cout grâce à la formule et les sommer pour avoir le cout J(θ0 , θ1, θ2 ).

**2. La Fonction de Cout (Avec vectorisation) :**

**Code :**

****

Cette fonction nous permettra de calculer le cout vu en cours ,comme la précédente juste que cette fois pour raccourcir les calculs et gagner en temps d’exécution au lieu de faire une boucle pour comparer à chaque fois on calcul le cout en une itération grâce à numpy .

Comparaison des temps d’exécution:

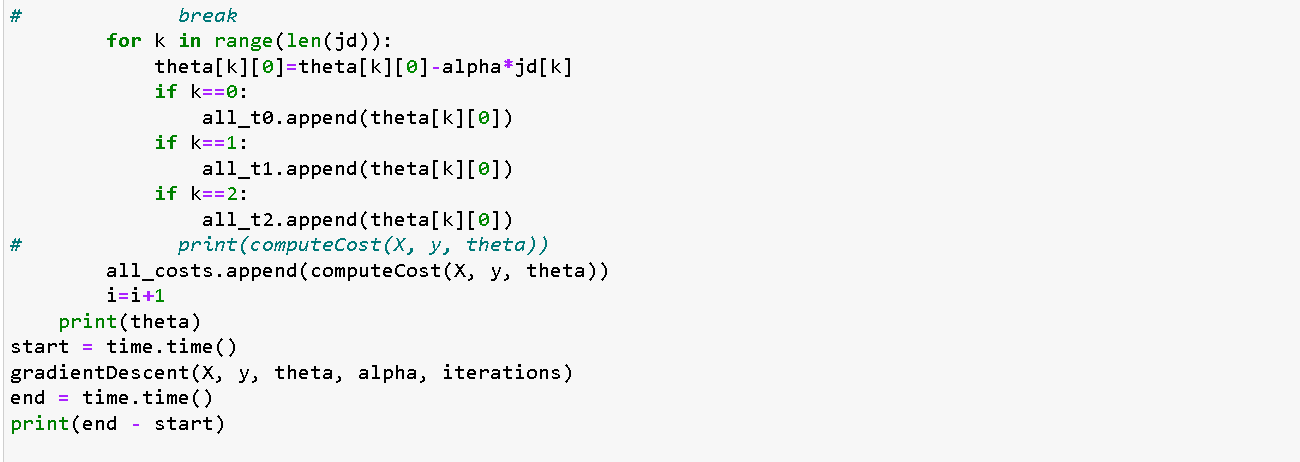
Pour theta=(0,0)

Sans vectorisation: 0.6931471805599458

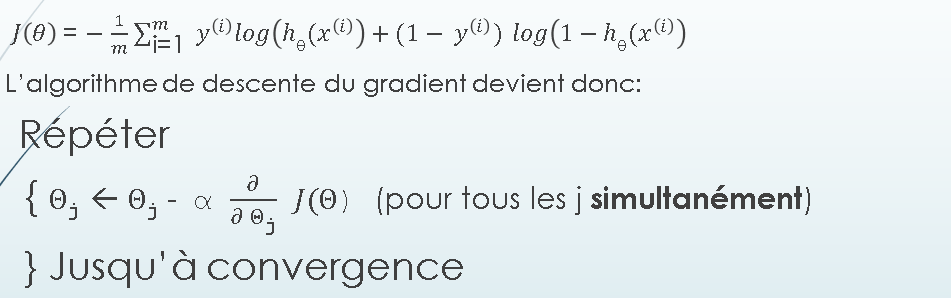
Avec vectorisation: 0.693147180559946

**3. Fonction de la descente du gradient :**





Dans cette fonction nous avons implémenter la descente du gradient ci-dessous vu en cours



Nous lui donnerons en entrée X ,y , alpha (taux d’apprentissage),et itérations qui est le nombre d’itérations qu’on fixe pour la convergence ,tant qu’on n’a pas atteint le nombre d’itérations fixer ou la dérivée est assez petite on mettra à jour à chaque fois θ0 et θ1 et θ2:

**theta[0,0] = theta[0,0] - alpha \* jd[0]**

**theta[1,0] = theta[1,0] -alpha \* jd[1]**

**theta[2,0] = theta[2,0] -alpha \* jd[2]**

**Dans notre algorithme nous avons généralisé pour theta[n,0] = theta[n,0] -alpha \* jd[n]**

Si on l’applique à notre exemple : jd est un tableau à 3 cases qui contiendra les dérivés de :jd[0] (dérivé de J(theta0,theta 1,theta2) par rapport à theta 0 qu’on calcul grâce à un appelle à la fonction j ) et : jd[1] (dérivé de J(theta0,theta 1,theta2) par rapport à theta 1 qu’on calcul grâce à un appelle à la fonction j et : jd[2] (dérivé de J(theta0,theta 1,theta2) par rapport à theta 2 qu’on calcul grâce à un appelle à la fonction j) .

on l’applique jusqu’à divergence ( atteindre le minimum local ) .

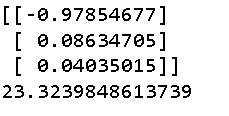
**Initialisation de θ0 et θ1 et θ2 ,calcul du cout initial et appel de la fonction de calcul du gradient :**



Theta est un vecteur de 3 lignes et une colonne qui contiendra nos θ0 θ1 θ2, initialCost pour calculer le cout initial , on fixe le nombre d’itérations à 1500 et le taux d’apprentissage alpha à 0.01 ensuite on appelle la fonction de calcul du gradient qu’on a défini précédemment et regardons le résultat :

On observe à chaque fois le changement des valeurs de nos θ0 et θ1 θ2 jusqu’à convergence .

Le vecteur des theta finaux est :



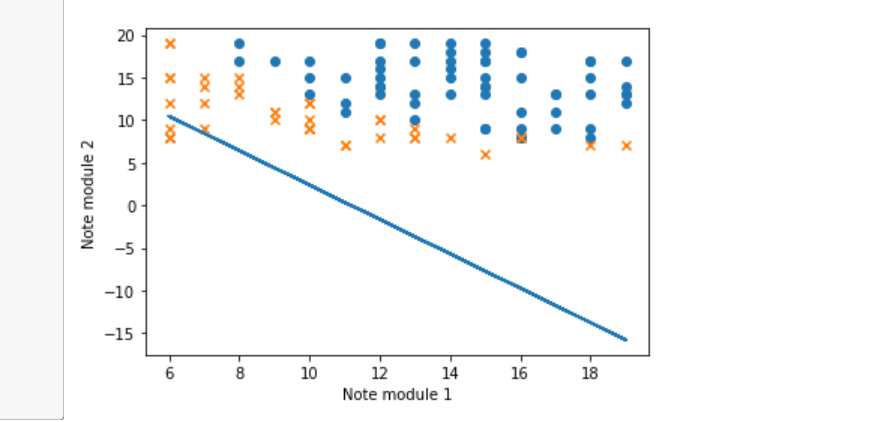
**Dessin de la limite de decision (Desicion Boundary) :**

Cette fonction nous permettera de classifier nos données au dessus de la ligne y=1 l’etudiant est admis en dessous de la ligne y=0 l’etudiant n’est pas admis**.**

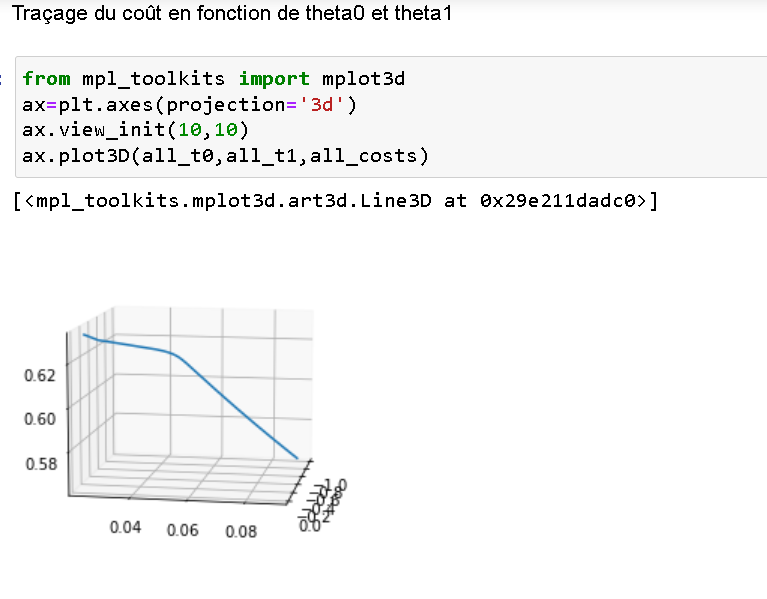
**On obtient la fonction de la ligne de decision :**

hθ(x)= θ0X0 +θ1X1 + θ2X2 =0

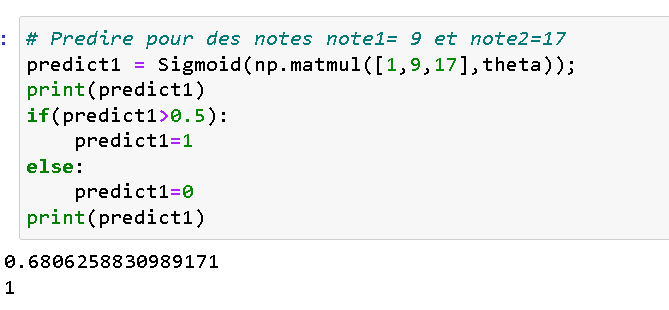
X2 =( - θ0X0 - θ1X1) / θ2



**Traçage de la fonction du coût en fonction de theta0 et theta1 :**



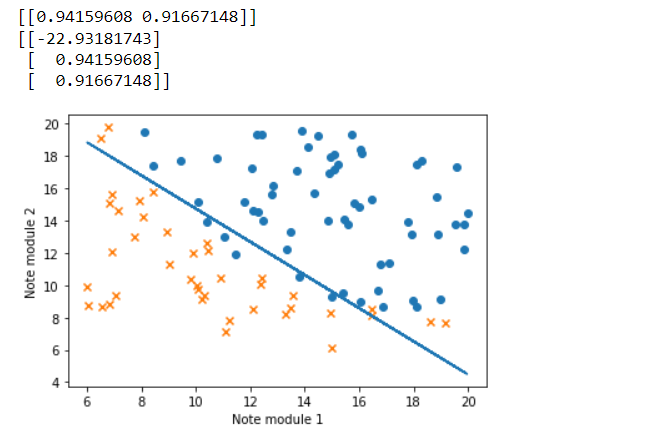
**Prédire les valeurs de y :**



Comparaison avec scikitlearn:

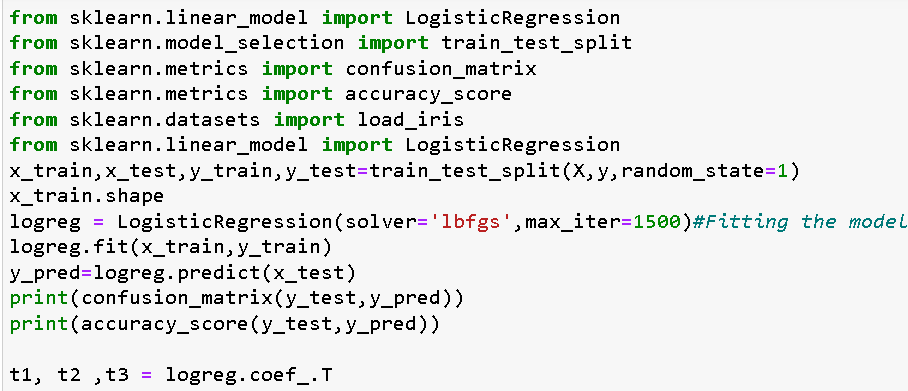
Cette algorithme nous a permis d’analyser la decision boundary donné par scikitlearn

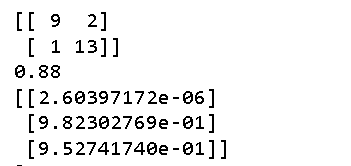




On obtient la meme ligne de décision que celui de notre algorithme.

un autre algorithme nous a permis de voir les metriques de performences la matrice de confusion,la précision et les thetas donnés par scikitlearn:





**Renforcement d’apprentissage:**

pour renforcer notre apprentissage nous avons tester avec un nombre d'iterations=15000 , les resultats ont été meilleur .

