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



APPRENTISSAGE AUTOMATIQUE

ET RESEAUX DE NEURONNES

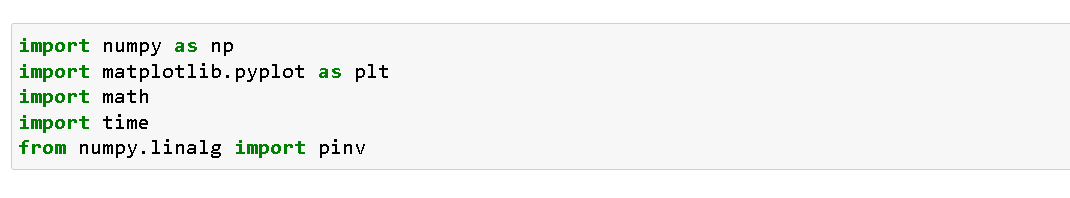
# **« RAPPORT TP 4 »**

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

***OBJECTIFS***

* Dans ce TP, nous aimerions faire une classification binaire en utilisant la régression.
* Pour ce faire, nous étudierons un ensemble de données avec la variable (y) représentant la commercialisation d'un produit et les caractéristiques (X) représentant les résultat des tests de qualité test 1 et test 2 du produit.
* La prédiction se fera avec l'algorithme de descente du gradient avec régularisation.

**Importation des librairies nécessaires pour le travail**



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

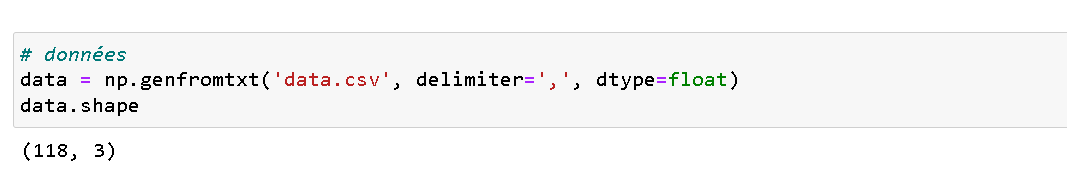
***matplotlib :*** pour la manipulation des graphs (2d, 3d)

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

***Math :*** pour der aux fonctions mathématiques (expo, log)

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

**Code :**



**Résultat :**

***118 :*** signifie le nombre de données (ou exemples) dans le fichier (***data.csv***) c.à.d. il traite ***118*** *données****.***

***3 :*** représente 3 colonnes (2 caractéristiques et y valeur prédite 1 ou 0) dans le fichier

**(1 ère colonne :** Qualité test 1

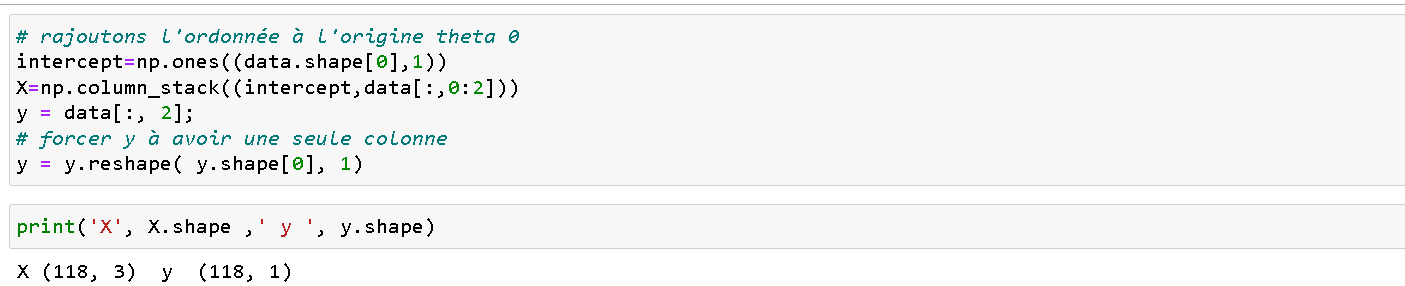
**et**

**2 -ème colonne :** Qualité test 2

**Et 3eme colonne :** commercialisation du produit(1 commercialisé / 0 non commercialisé)

***Mettons ces données dans leurs vecteurs correspondants :***

**Code :**



**X :** est une matrice sous forme de 3 colonnes X0 =1, X1=Qualité test 1,X2=Qualité test 2 qui contient 118 données (lignes)

**Y :** représente un vecteur de 118 données (lignes) composé d’une seul colonne qui représente la représentation du produits. Y=0 ou 1

# **Descente du Gradient**

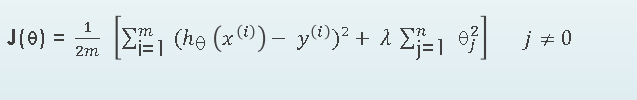
# **Préparation des fonctions:**

**1. La Fonction de Cout :**

**Code:**



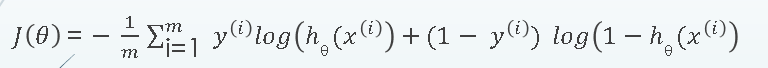
Dans cette fonction on a calculer le cout avec régularisation pour la regression logistique:



On a commencer par parcourir tous les X ligne par ligne et pour chaque ligne on parcours le vecteur theta donnée en entrée à la fonction cette étape(j=j+theta[k][0]\*X[i,k] nous permet d’avoir θ0X0+ θ1X1+ θ2X2+ θ3X3....... ensuite une fois qu’on a j on va calculer pour chaque ligne de X dans l’etape j=y[i]\*math.log(Sigmoid(j)+0.00001)+(1-y[i])\*math.log(1-Sigmoid(j)+0.00001) 

Ensuite jt=jt+j pour avoir la somme à chaque fois 

Et jt=(-jt)/((len(X))) pour avoir on divise la valeur trouvé précédemment par m la taille de X :



Dans la derniere partie nous avons appliquer la régularisation :

On a parcouru le vecteur theta pour valeur initiale I=1 on fais la somme des that [i] 2  :

**for i in range(len(theta)):**

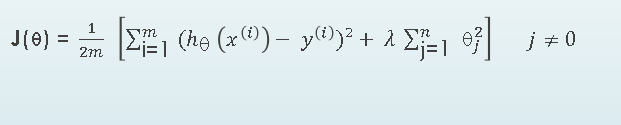
**if(i!=0):**

**t=t+theta[i][0]\*\*2**

Cette etape nous a permis d’avoir :

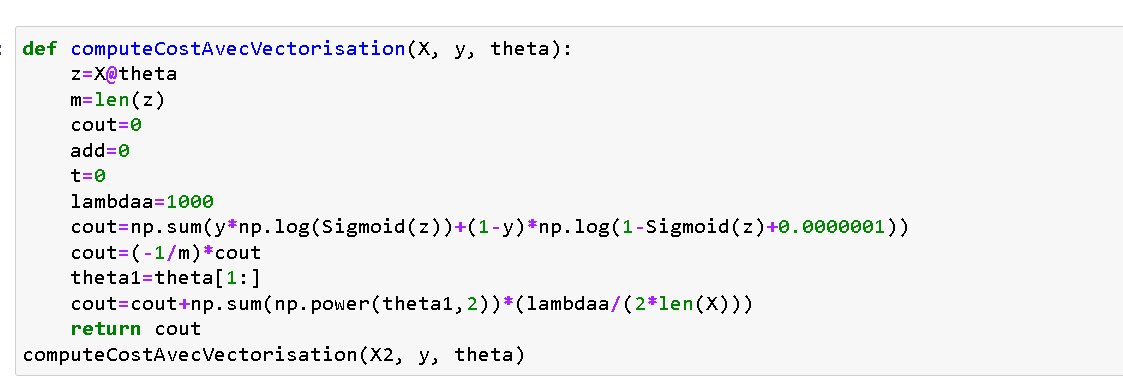


Une fois cette etape faite on sort de la boucle et on fais t=t/(2\*len(X2)) (on divise par 2\*m) et jt=jt+t pour l’aditionner à jt pour avoir :



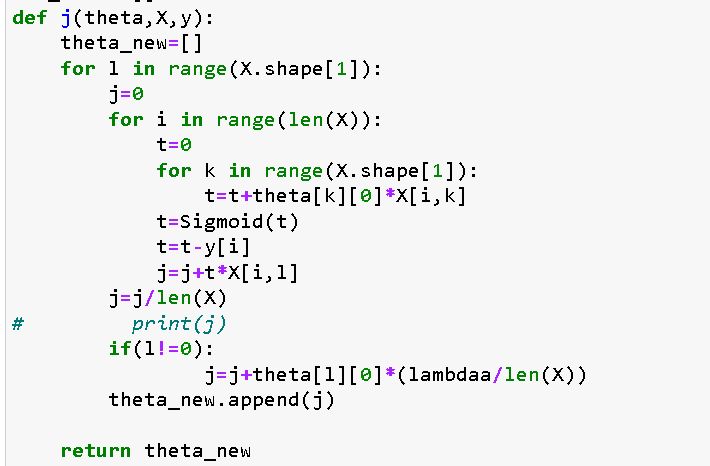
Avec lambda =1

**-Fonction de cout avec vectorisation :**



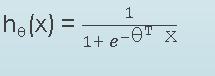
**3. La Fonction de Décente du Gradient:**

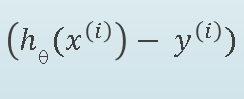
Pour la descente du gradient on a utiliser une fonction j qui nous permettera d’avoir la dérivé à chaque appelle de j .

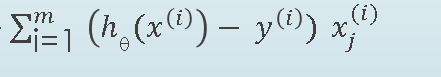


for k in range(X.shape[1]):

t=t+theta[k][0]\*X[i,k]

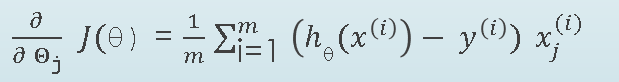
Cette étape nous permet d’avoir θ0X0+ θ1X1+ θ2X2+ θ3X3....... ensuite une fois qu’on obtient le resultat on appelle sigmoid(t) pour avoir:

Ensuite l’etape t=t-y[I] nous permet de calculer: 

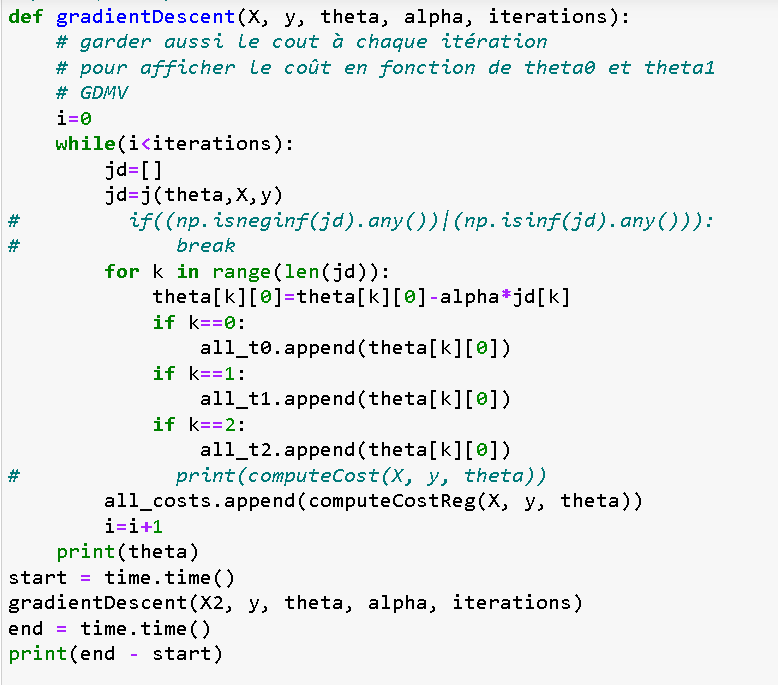
Ensuite l’étape : j=j+t\*X[i,l] nous permet d’avoir: 

Une fois qu’on a ce resultat on fais :

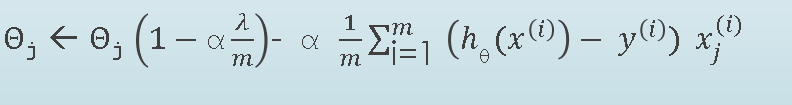
j=j/len(X) pour avoir la dérivée de j par rapport à theta :



**Fonction de la descente du gradient:**



Dans cette fonction on a applique à chaque fois la descente de gradient vu en cours :



# **Appel au fonctions:**

**1. Initialisation de theta**  **:**

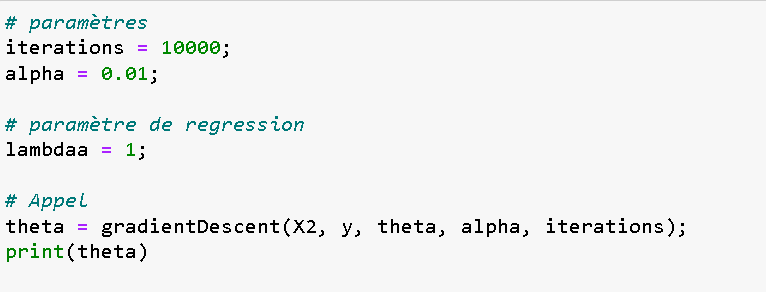
**Code:**



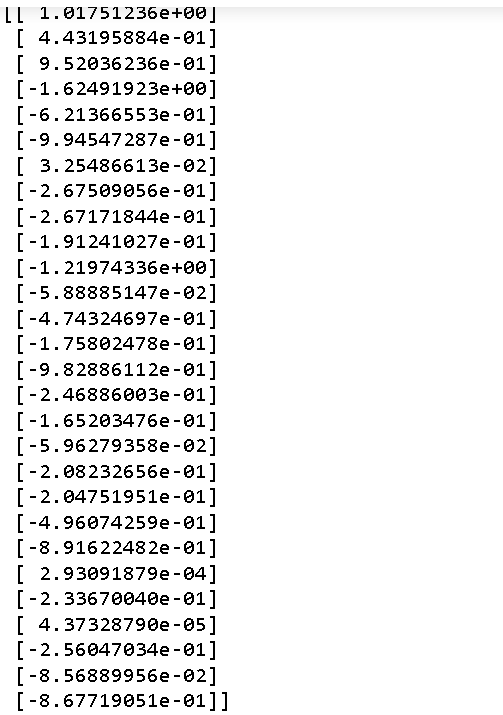
**2. Calcul du cout initial:**



**3-Appel à la fonction de la descente du gradient:**

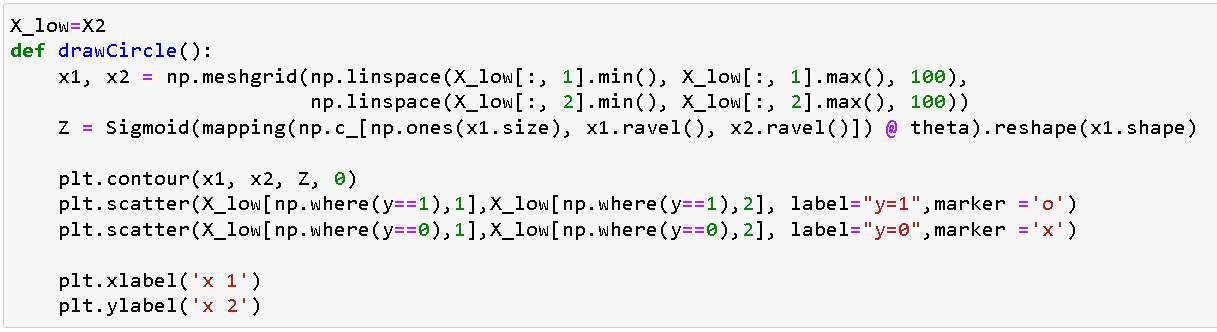


**Resultat:**

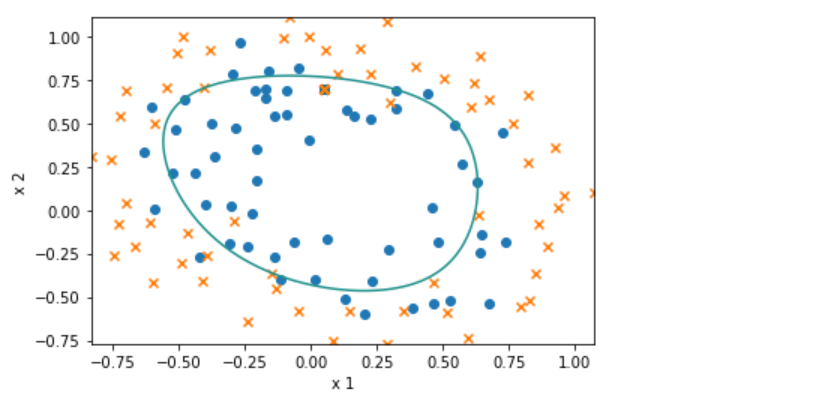


**4-Desicion Boundary:**

**Code:**

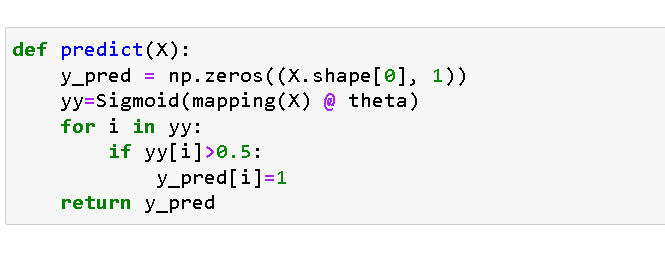


**Resultat:**

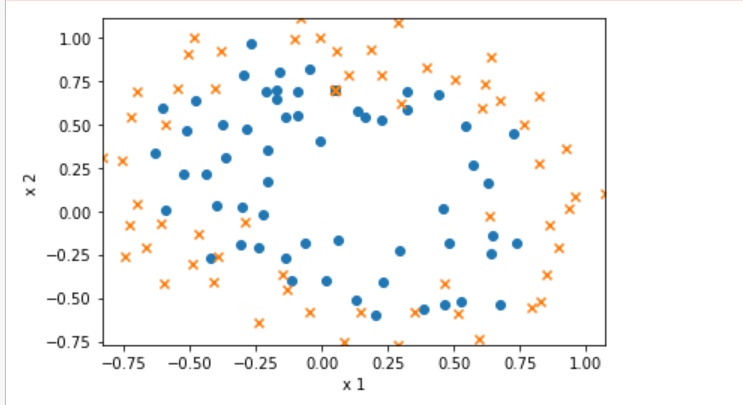


**Classification(prediction):**

**Code :**

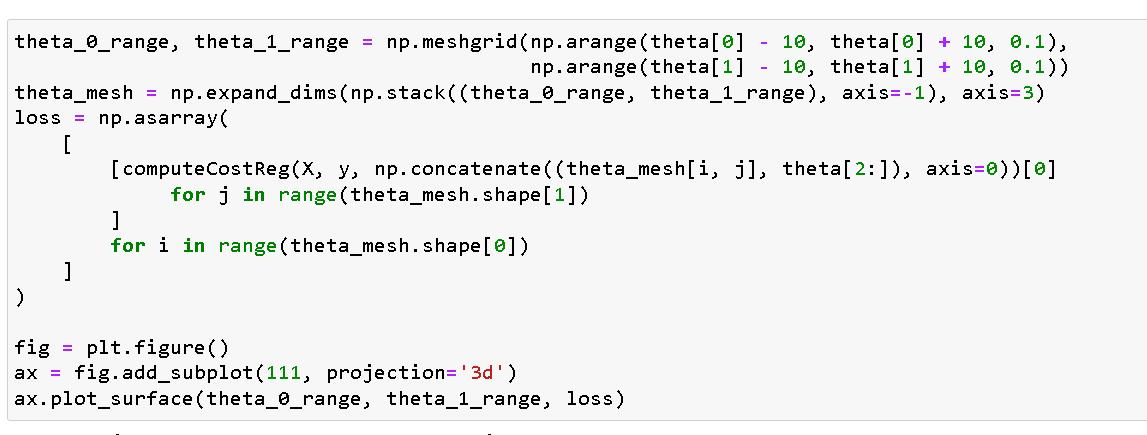


**Affichage :Graphe representant les acceptations selon les carracteristiques** :

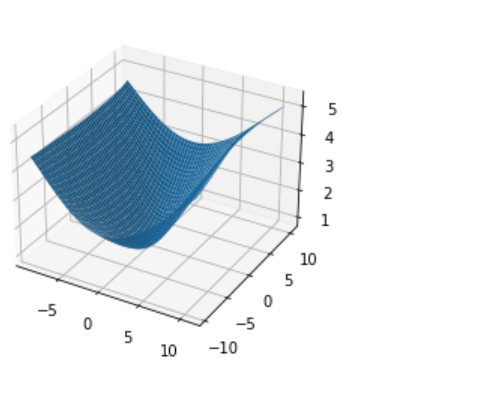


**Tracage du cout en fonction des theta:**

**Code:**



**Resultat:**



**Qualité du classifieur :**

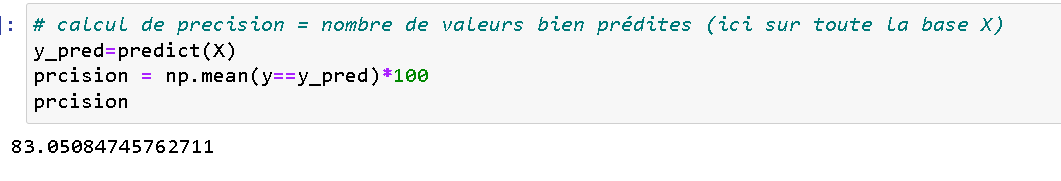
Prédire des valeurs de y

Ici il serait interessant de calculer la précision de notre classifieur

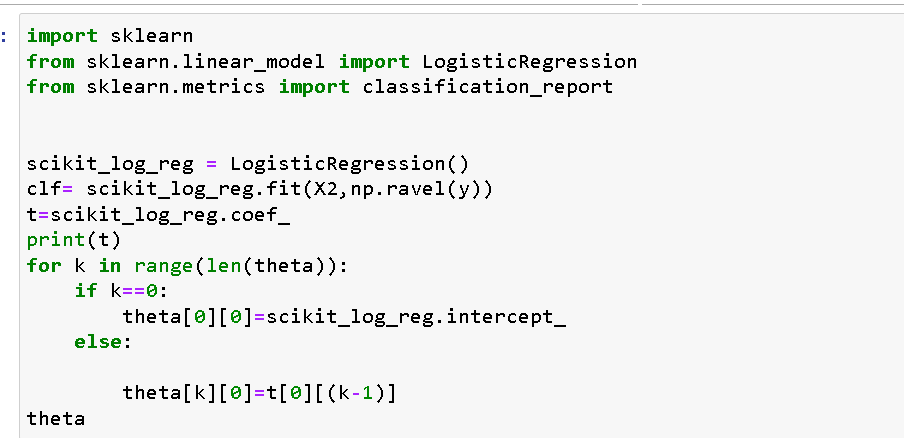
Essayons de calculer ça avec

moyenne(y==y-pred) \* 100

Ceci donnera un pourcentage de precision

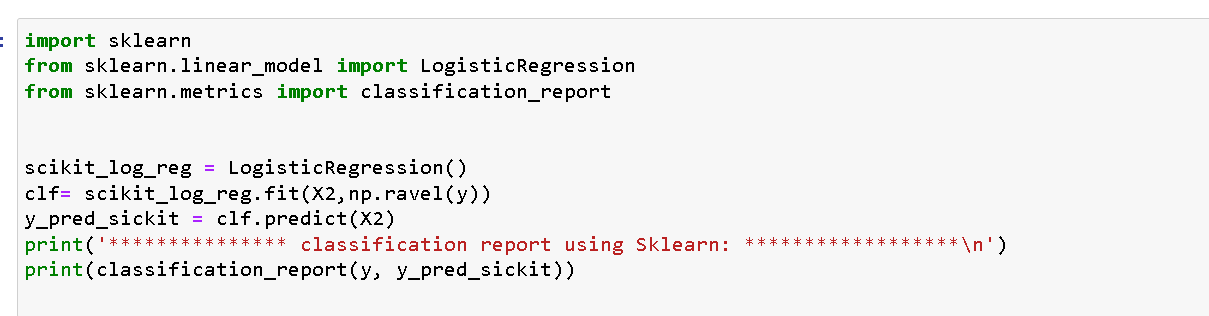


**Verification de l’implementation :**



On obtient les memes valeurs de theta donc on deduit que notre implementation a bien été faite.

**Dans ce qui suit on va tester les mesures de performances :**



**Resultat:**

