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



APPRENTISSAGE AUTOMATIQUE

ET RESEAUX DE NEURONNES

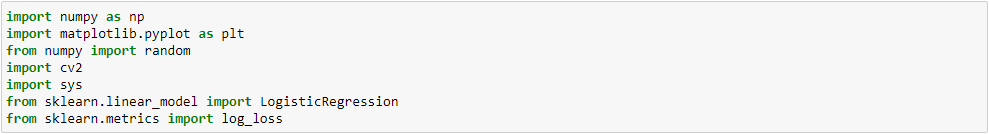
# **« RAPPORT TP 5 »**

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

***OBJECTIFS***

* Dans ce TP, nous aimerions faire une classification multi classes. Dans la 1ere partie nous allons utiliser la régression logistique, et une autre classification en utilisant les réseaux de neurones dans la 2 -ème partie. Par la suite on va comparer entre ces deux approches.
* Pour ce faire, nous étudierons un ensemble de données qui sont des images de chiffres manuscrits. Composées de 400 pixels (20x20) chacune, et chaque image représente une classe (chiffres de 0 à 9) .
* La prédiction se fera avec l'algorithme de descente du gradient (pour 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)

***CV2 :*** pour traiter les images

***SYS :*** pour interagir avec l’interpréteur Python

***SKLEARN :*** pour utiliser les modèles prédéfini de machine learning

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

**Code :**

****

**Résultat :**

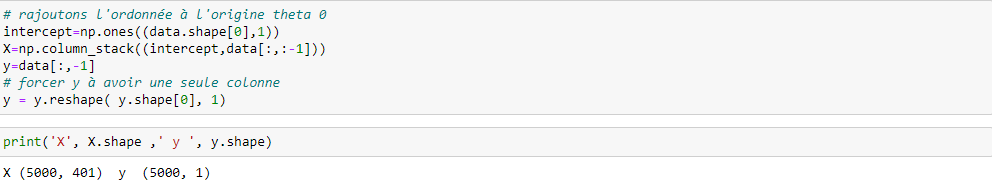
===> 

***5000 :*** signifie le nombre de données (ou exemples) dans le fichier (***data.csv***) c.à.d. il traite ***5000 images***

***401 :*** représente 401 colonnes dans le fichier

**(colonnes 1 à 400 X: Les pixels d’une image 20\*20 =400**

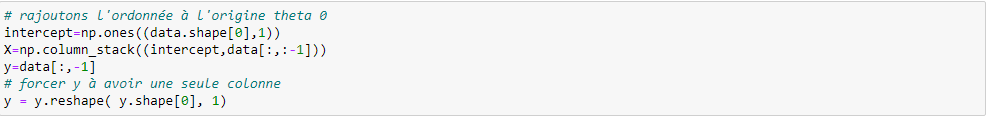
**colonne 401 Y: La classe d’une image de 0 à 9)**

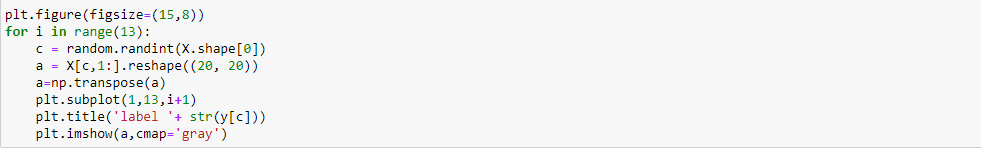


On met dans X nos caractéristiques (les pixels 20\*20) et dans y les classes (labels) des images . on a 5000 exemples . X aura 401 colonnes (400 données +X0) .

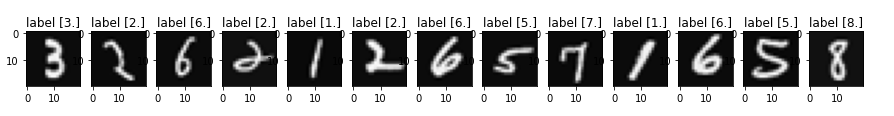
**Visualisation aléatoire de quelques données :**

**Code :**





**Résultat :**



**13** images sont représentées avec ses classes correspondantes (les labels).

**Partie 1: régression logistique**

# **Préparation des fonctions:**

**1. La Fonction Sigmoid :**

**Code:**



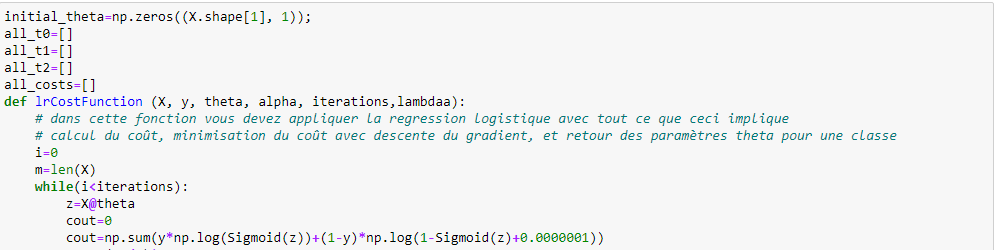
Cette fonction nous permettra de calculer la Sigmoid d’une valeur donnée, nous aurons besoin de la sigmoid pour calculer l’hypothèse de la régression logistique on lui donnera en entrée quand on l’appellera :

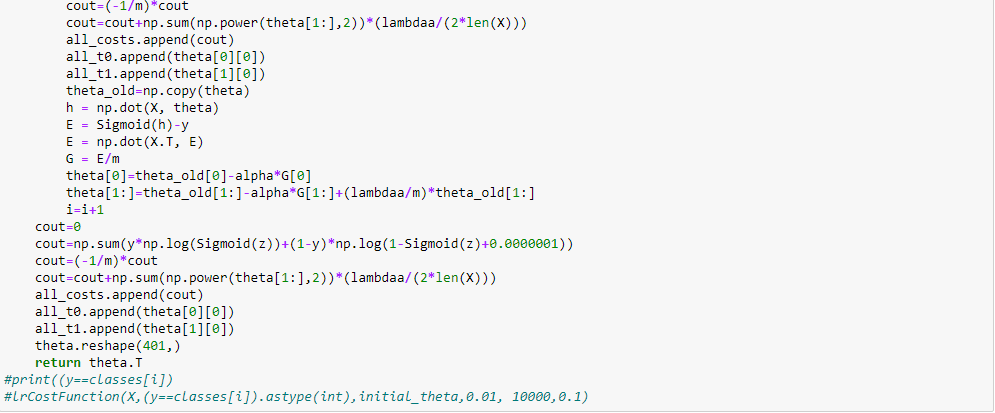
theta0 \*X0+theta1\*X1+theta2\*X2 pour avoir hθ(x).

**2. La Fonction lrCostFunction :**

Dans cette fonction, on a appliqué la régression logistique avec tout ce qui implique : calcul du coût, minimisation du coût avec descente du gradient, et retour des paramètres theta pour une classe.

**Code :**





Exécuter la fonction avec les paramètres suivants :

X : nos données

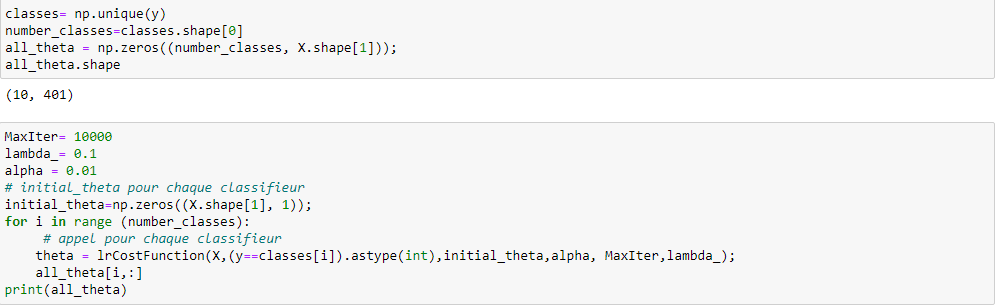
Y : nos classes

All\_theta= [[0,…,0],… ,[0,…,0]]

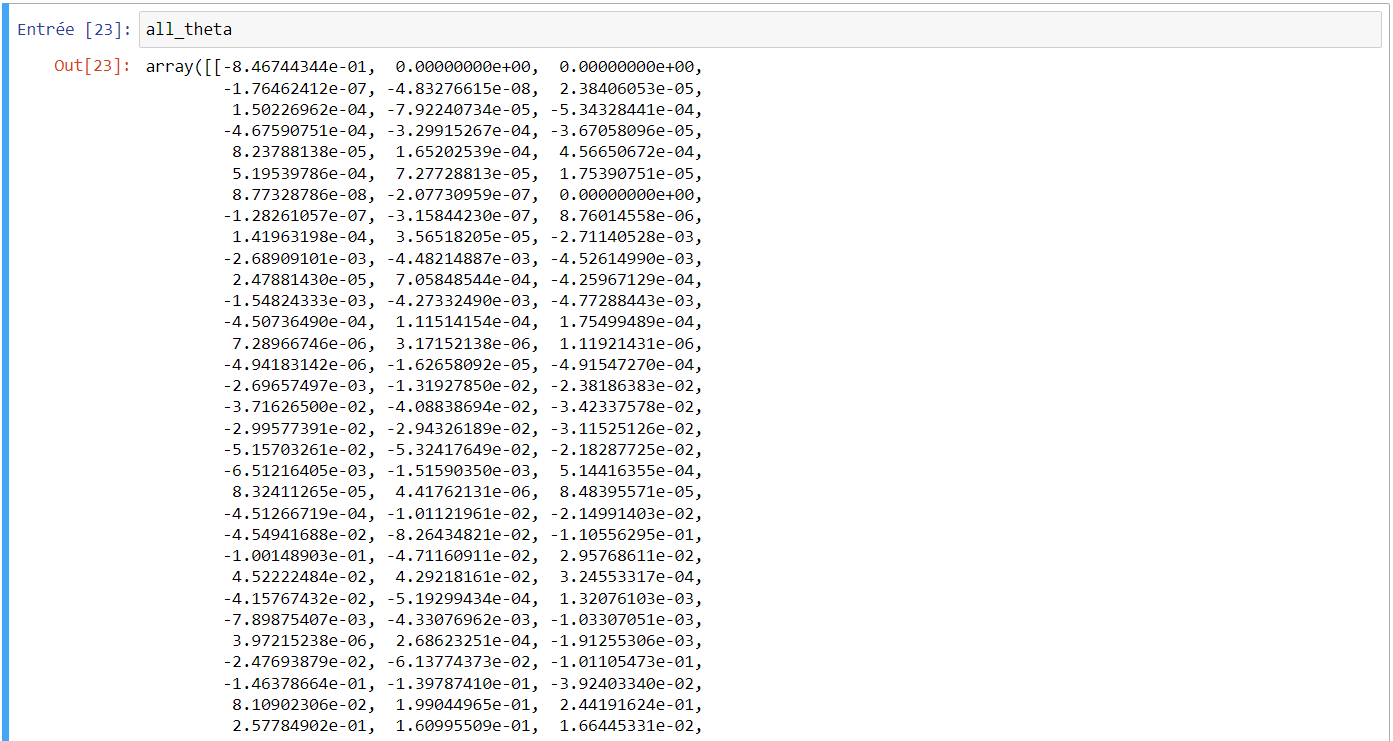
MaxIter= 10000

lambda\_= 0.1

alpha = 0.01



**Résultat :**

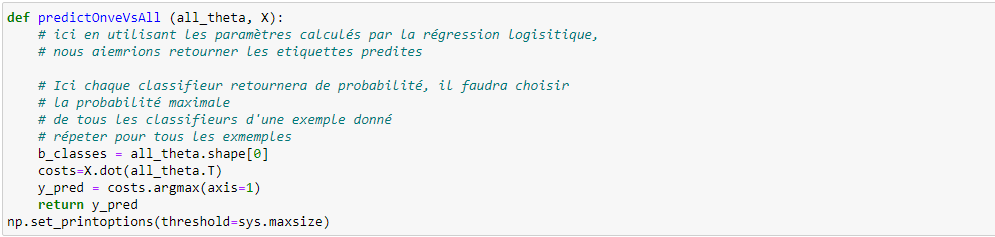


Pour chaque classe on a appliqué la descente de gradient pour minimiser les theta on obtient alors une matrice all\_theta de theta finaux .

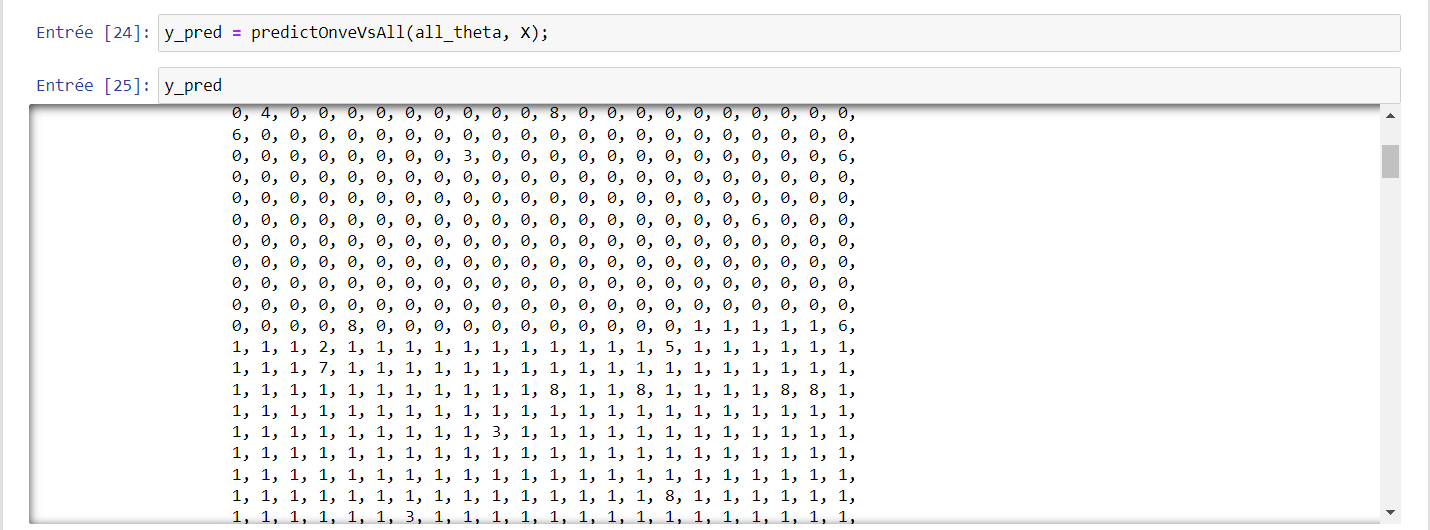
**3. prédire la classification :**

**Code :**

Dans cette partie on va calculer les hypothèses pour toute les classes qui seront des probabilités d’appartenance à la classe ensuite on prend l’indice de l’hypothèse maximum pour chaque classe .



**Résultat :**

**4. la précision :**

**Code et Résultat :**



On obtient une précision de 88,48 % un résultat plutôt satisfaisant par rapport au nombre de données qu’on a traité .

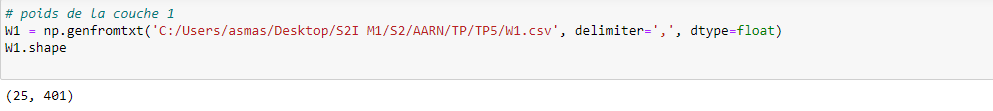
# **Partie 2 : Réseaux de neurones**

Dans cette partienous allons prédire la classe d’une image en utilisant un réseau de neurones.

Ce réseau contient 3 couches :

* Input layer : avec 401 neurones (taille d’image 20\*20 plus intercept).
* Hiden layer : avec 25 neurones.
* Output layer : avec 10 neurones (les classes de 0 à 9).

Nous allons charger les poids de chaque layer:

****

****

Initialiser notre paramètres à :

input\_layer\_size = 400

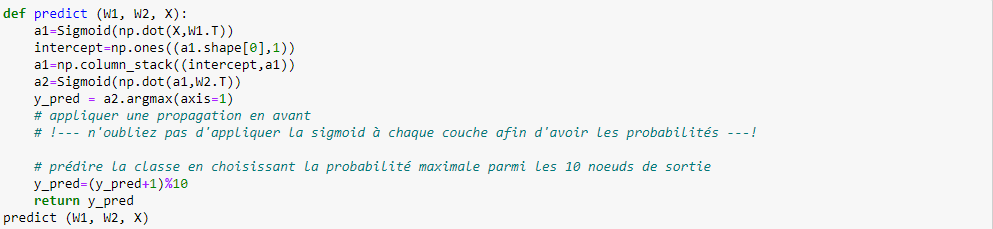
hidden\_layer\_size = 25

num\_labels = 10

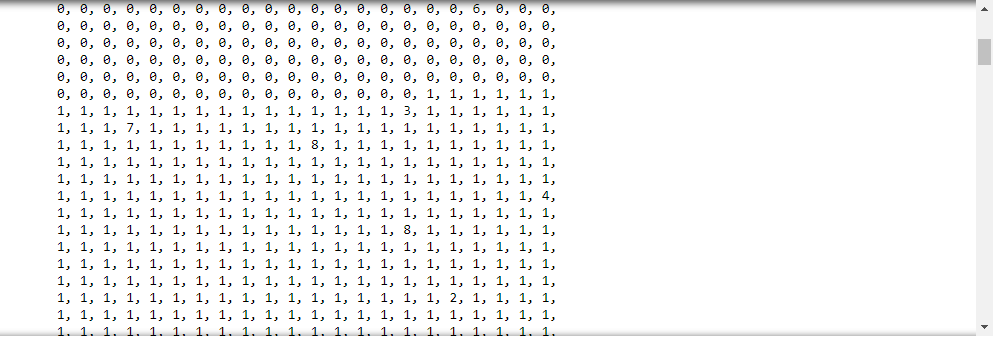


**1.prédir les output en utilisant RN :**

**Code :**

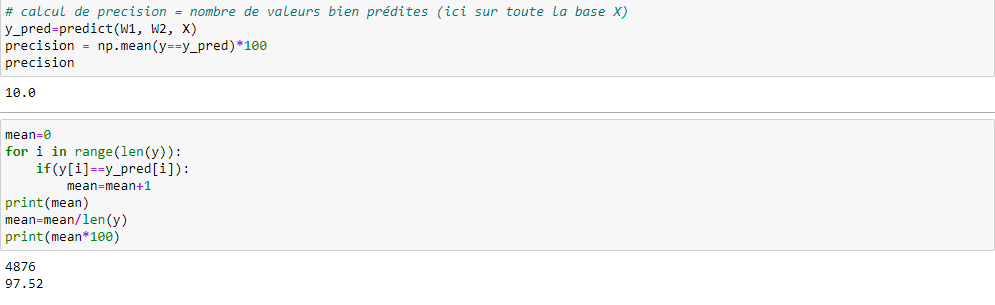


**Résultat :**



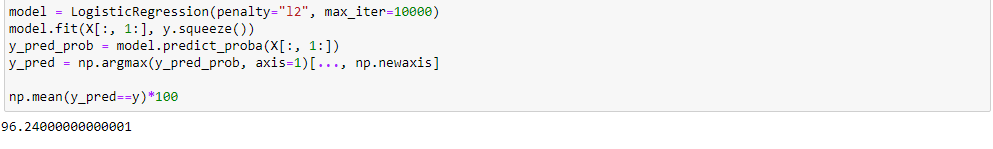
**2. la précision :**

**Code et résultat :**



On obtient une précision de 97.52 % qui est un résultat très satisfaisant et meilleur que la régression logistique .

**Vérification de l'implémentation :**

****

Comparaison avec scikitlearn:

Cet algorithme nous a permis de calculer la précision donnée par scikitlearn, et on remarque qu’on a obtenu un résultat très rapprocher à celui de notre algorithme de RN, mais un petit peu plus élevé par rapport le résultat de l’algorithme de régression logistique.

La précision donnée par scikitlearn : 96.24

La précision donnée par algorithme de RN : 97.52

La précision donnée par l’algorithme de régression logistique : 88.48.

Donc en conclusion on peut dire que la meilleur approche pour classifier ces données est les réseaux de neurones. Cette méthode n’est pas toujours la meilleur pour d’autres données et pour une autre classification car il se peut qu’elle soit très couteuse .

**FIN**