****

**Ministre de l’enseignement supérieur et de la recherche**

**Université Iba Der THIAM de Thiès**

**Département Informatique**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Projet:**

**IA : Machine Learning**

***MEMBRES DU GROUPE :***

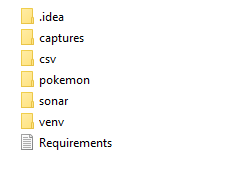
**MAMADOU NIANG**

**CHEIKH TALL**

**TP : Apprentissage Automatique**

Afin de valider les compétences du cours, nous avons reçu un projet de fin de module. Ce projet est divisé en deux parties et ce rapport va détailler les différentes étapes qui nous ont permis de résoudre ces différents exercices.

Pour se faire, afin de mieux comprendre le fonctionnement, nous avons décidé d’utiliser directement python dans notre éditeur PYCHARM au lieu d’utiliser ANACONDA. Nous avons donc créé un projet python nommé IA\_M1\_2021\_12\_28. La capture suivante va permettre de détailler la structure du projet.



Le dossier capture contient les différentes captures qui montrent les résultats obtenus.

Le dossier csv regroupe les différents fichiers Excel aves lesquels nous allons travailler.

Le dossier pokemon contient le fichier python qui va contenir les codes pour traiter le tp2.

Le dossier sonar contient le fichier python qui va contenir les codes pour traiter le tp1.

Le dossier venv est notre environnement virtuel. Requirements est un fichier texte qui contient les différents modules que nous devrons installer pour effectuer le travail.

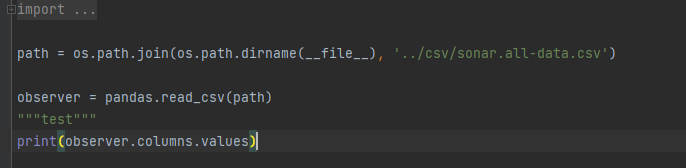
Il est nécessaire de l’installer pour pouvoir exécuter le projet. On a utilisé la commande **py -m venv venv.** Ensuite on l’active avec la commande **venv/Scripts/activate**. Puis on accède au dossier que l’on veut exécuter en faisant **cd nom\_du\_dossier.** Puis on lance l’exécution par **py app.py**.

Cette façon de travailler va nous permettre de mieux comprendre ce qui se passe avec les différentes fonctions que nous utilisons.

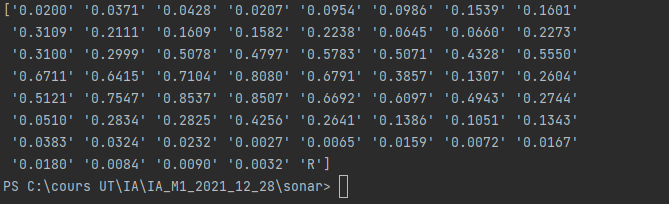
**TP1 : Les k plus proches voisins Classification**

Pour ce TP, nous allons travailler sur la base sonar.all-data.

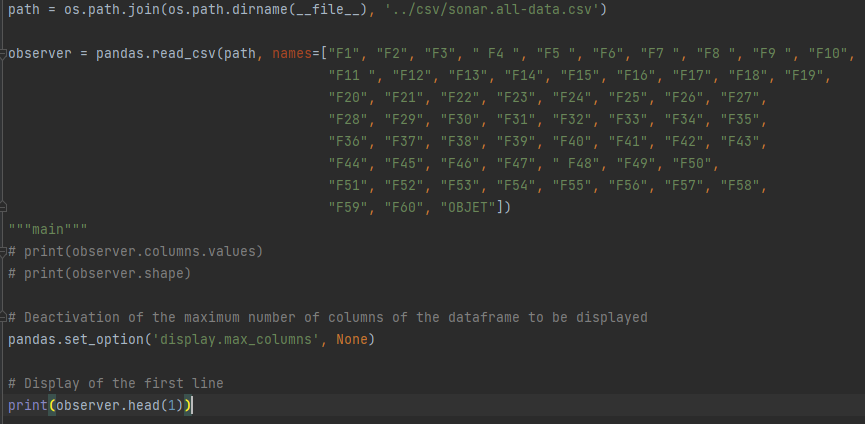
Nous allons d’abord charger la base de données sonar :



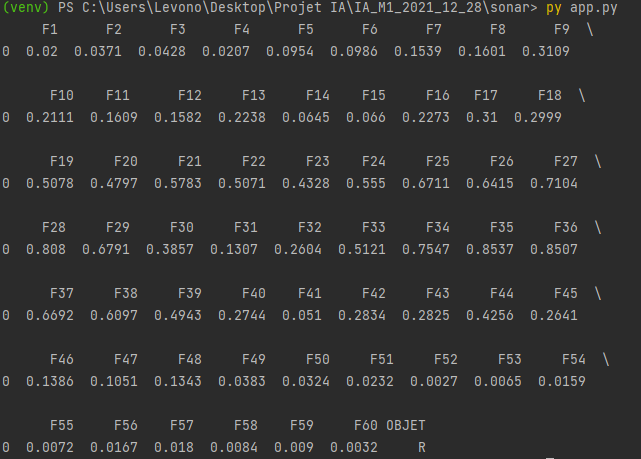
On affiche les entrées du fichier avec la commande print(fichier.columns.values) :



On renomme ensuite les variables avec la commande observer ***= pandas.read\_csv(path, names=["F1", "F2", "F3", "F4", "F5", "F6", "F7", "F8", "F9", "F10",  
 "F11 ", "F12", "F13", "F14", "F15", "F16", "F17", "F18", "F19",  
 "F20", "F21", "F22", "F23", "F24", "F25", "F26", "F27",  
 "F28", "F29", "F30", "F31", "F32", "F33", "F34", "F35",  
 "F36", "F37", "F38", "F39", "F40", "F41", "F42", "F43",  
 "F44", "F45", "F46", "F47", " F48", "F49", "F50",  
 "F51", "F52", "F53", "F54", "F55", "F56", "F57", "F58",  
 "F59", "F60", "OBJET"])***



On affche ensuite les premières lignes de la base de donnée **: print(observer.head())**



**Combien de classes ?**

Nous avons deux classes : les Rochers (R) et les mines (M).

**Combien de caractéristiques descriptives ? De quels types ?**

Pour les statistiques descriptives, nous en avons 6 :

Count qui donne le nombre d’entrées et qui est de type numérique

Mean : qui nous donne la moyenne et qui est de type numérique

Std : qui nous donne l’écart-type

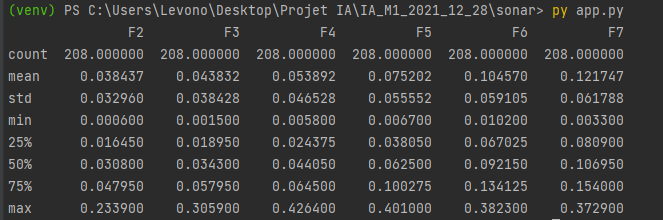
Min, Max qui nous donnent les minimum et maximum et sont de type numérique

Les quartiles qui nous donnent les valeurs à 25, 50 et 75%. Ils sont donnés en pourcentage

**Calculer les statistiques de base des variables 2 à 7**

Nous avons eu le résultat en faisant :

**stat\_base = observer[["F2", "F3", "F4", "F5", "F6", "F7"]]  
print(stat\_base.describe())**



**Combien d’exemples ?**

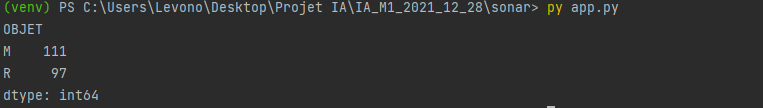


Nous avons 61 features avec 208 entrées chacune numérotées de 0 à 207.

**Combien d’exemples de chaque classe ?**

Nous allons utiliser la commande suivante :

**values\_expl = observer['OBJET'].value\_counts()  
print("Exemples de chaque classe : " + str(values\_expl))**



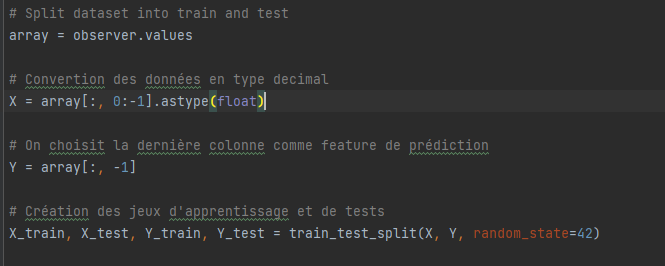
Nous avons 111 objets de type Mine et 97 de type Rocher.

**Comment sont organisés les exemples ?**

Chaque exemple représente une instanciation de la classe à laquelle elle appartient. Elle a toutes les caractéristiques qui permettent de la reconnaitre et de la différencier des éléments de l’autre classe.

**Apprentissage et test**

Créons le modèle knn en python et entrainons-le sur la base d’apprentissage :



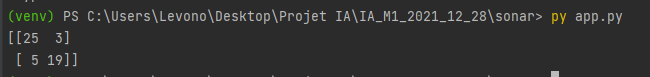
Le score obtenu en apprentissage :



Le score obtenu en test :



Affichons la matrice de confusion en utilisant la commande : **print(confusion\_matrix(predictions,Y\_test))**



Dans la ligne 0 il y’a 28 données. 25 sont correctement attribuées à 0 et 3 sont attribuées à tort à 1.

Dans la ligne 1 il y’a 24 données. 5 sont incorrectement attribuées à 0 et 19 sont correctement attribuées à 1.

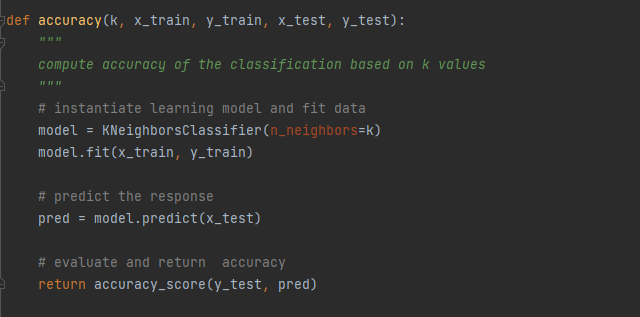
**Jouer avec le paramètre k**

K est influant par le fait que la modification de la valeur de K entraine un changement des résultats obtenus. Si K augmente, les valeurs obtenues en apprentissage et en test diminuent et inversement.

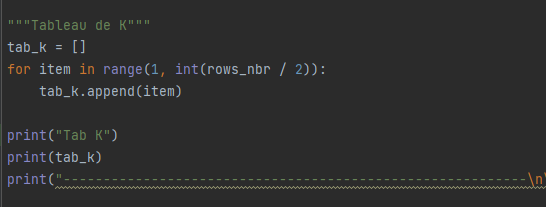
**Traçons la courbe de k en fonction des scores**

Nous avons utilisé des fonctions pour tracer la courbe.

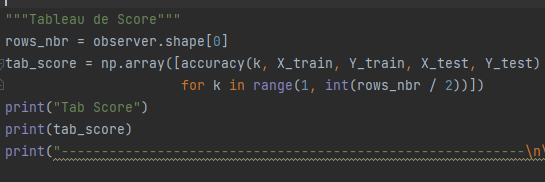
La fonction accuracy : il permet de générer les nombres qui vont être dans des tableaux.



On a d’abord le tableau des k-nombres qui représente le nombre d’exemples qu’o divise par 2 et on obtient 104. Ces éléments von représenter les abscisses.



Ensuite on a le tableau des scores. Les scores varient de 0 à 1 et vont représenter les ordonnées. Plus le score est proche de 1, la courbe va tendre vert le haut et inversement.



Ensuite, on trace notre courbe en utilisant la bibliothèque matplotlib.

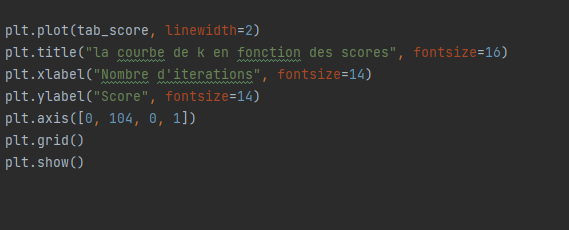
plt.title pour donner un titre à notre courbe ;

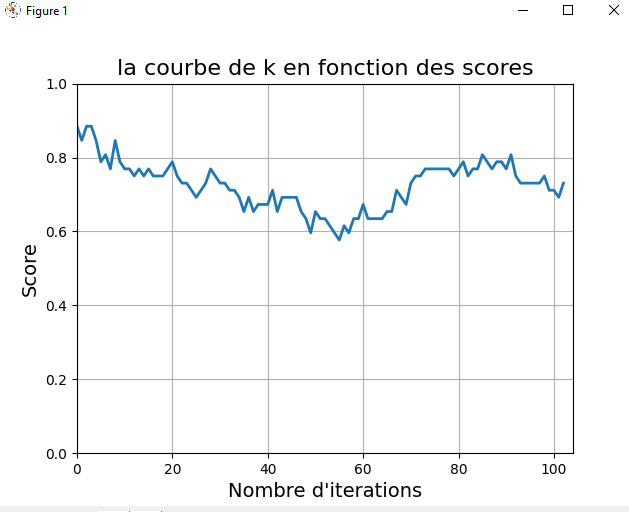
plt.xlabel, plt.ylabel pour définir les abscisses et les ordonnées en leur donnant des titres et une taille d’écriture ;

plt.axis pour définir les valeurs de départ et d’arrivées pour les éléments de la courbe ;

plt.grid pour afficher la grille

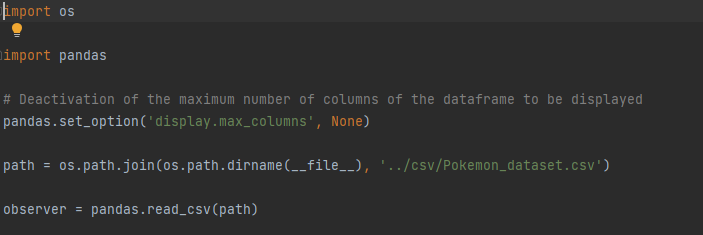
plt.show pour tracer la courbe et l’afficher à l’écran





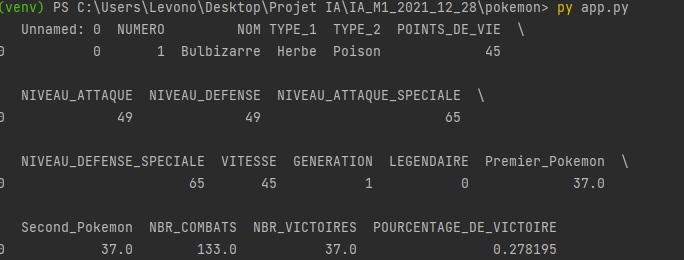
**TP2 : La régression linéaire**

Pour ce TP, nous allons travailler sur la base Pokemon\_dataset. Commençons d’abord par l’importer.



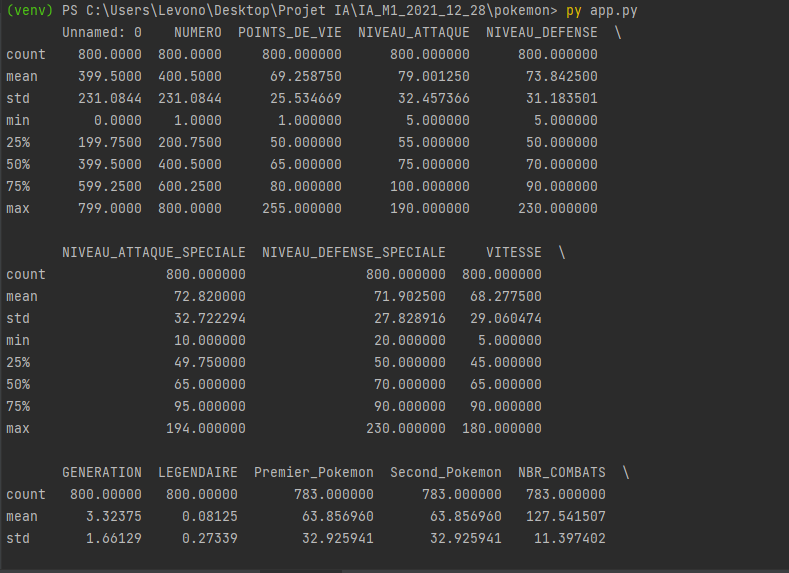
**Afficher les premières observations de la base de données ?**

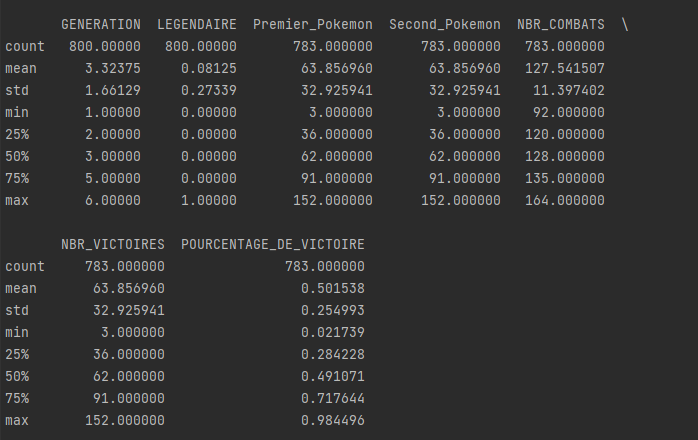
On affche ensuite les premières lignes de la base de donnée **: print(observer.head())**



**Combien de caractéristiques descriptives ? De quels types ?**

Les caractéristiques descriptives sont obtenues en faisant la commande **print(observer.describe()).**





Pour les statistiques descriptives, nous en avons 6 :

Count qui donne le nombre d’entrées et qui est de type numérique

Mean : qui nous donne la moyenne et qui est de type numérique

Std : qui nous donne l’écart-type

Min, Max qui nous donnent les minimum et maximum et sont de type numérique

Les quartiles qui nous donnent les valeurs à 25, 50 et 75%. Ils sont donnés en pourcentage

**Faire une analyse descriptive des données**

Nous pouvons dire à travers ces captures que :

25% des pokemons ont un pourcentage de victoire qui est au moins égal à 28%

50% des pokemons ont un pourcentage de victoire qui est au moins égal à 49%

75% des pokemons ont un pourcentage de victoire qui est au moins égal à 72%

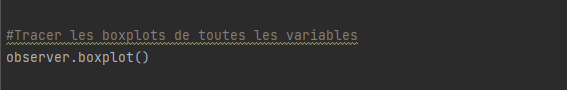
Le pokemon avec le plus grand pourcentage de victoire est à 98% de victoires

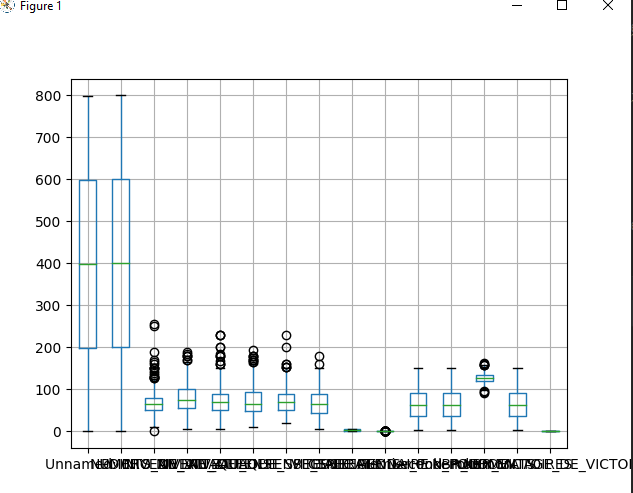
La moyenne de victoires est de 50% pour les pokemons

**Comment sont organisés les exemples ?**

**Tracer les boxplots de toutes les variables**

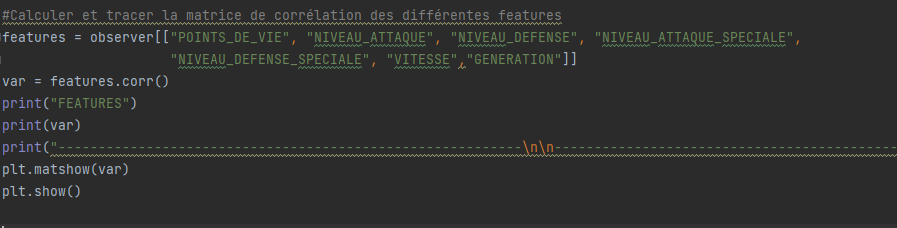
Pour tracer les boxplots nous avons utilisé la commande : **observer.boxplot ()**

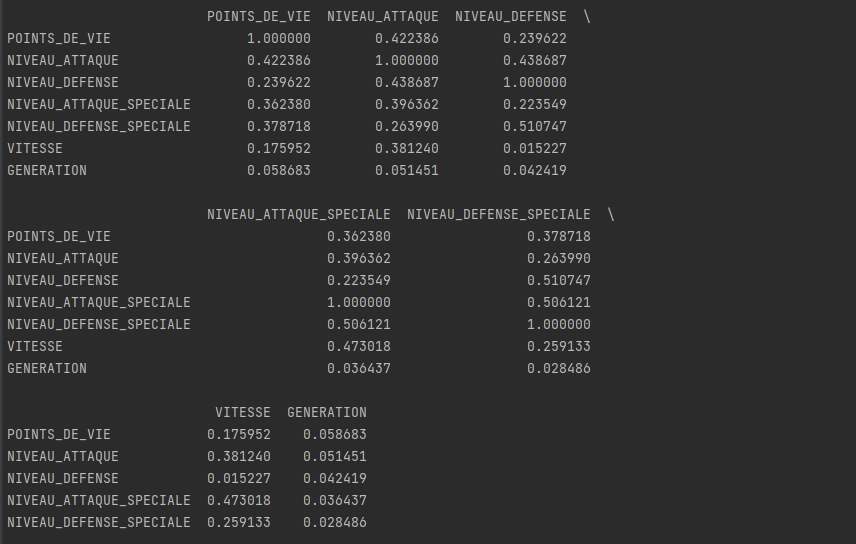


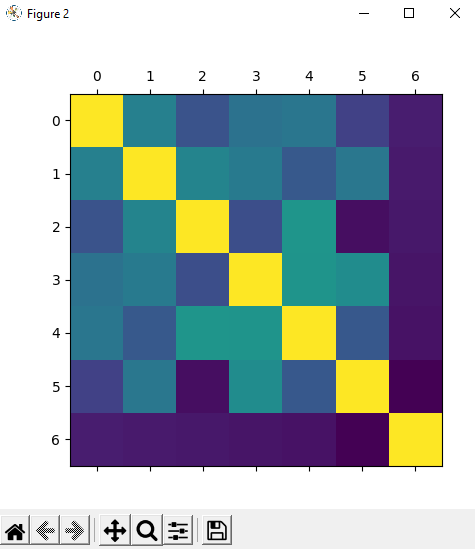
****

**Calculer et tracer la matrice de corrélation des différentes features**

Pour calculer la matrice de corrélation des features nous avons d’abord sélectionné les éléments représentant les features que nous avons mis dans une variable nommée features. Ensuite, nous avons utilisé la commande **features.corr ()** pour calculer la matrice de corrélation.

Nous avons ensuite affiché cette matrice en utilisant la fonction **print ()**. Enfin, on trace la matrice grâce à la fonction **matshow ()** puis on affiche le tracé.





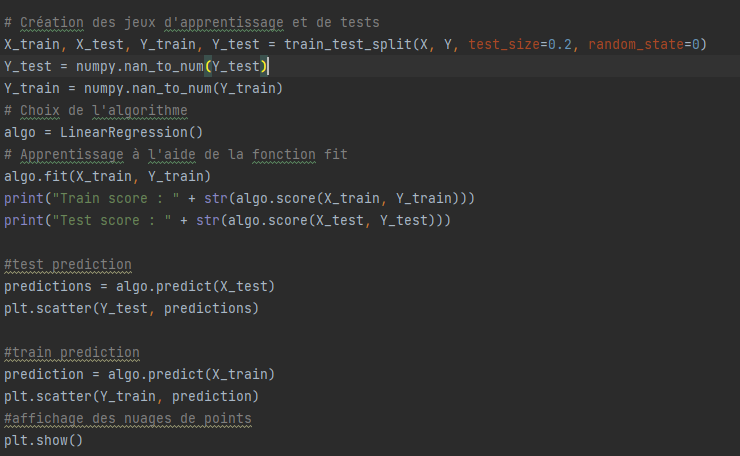
Les valeurs obtenues en apprentissage :



Les valeurs obtenues en test :



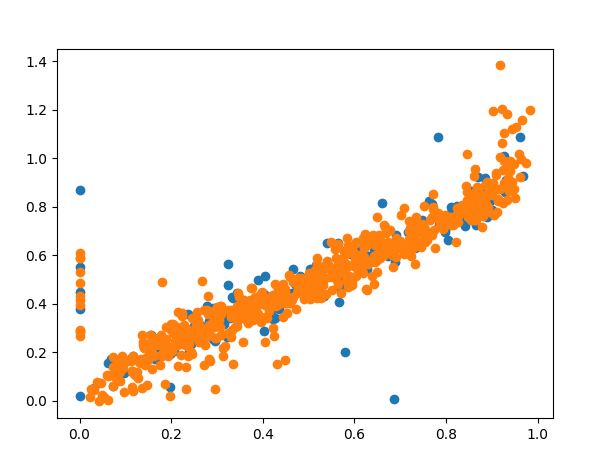
**Le nuage de points :**



Pour pouvoir tracer le nuage de points, on commence d’abord par créer les jeux d’apprentissage et de test. Ensuite on a choisi l’algorithme de régression linéaire faire l’apprentissage et le test.

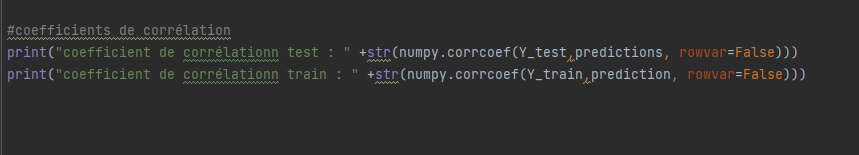
Ensuite pour chacun des cas on fait des prédictions et on crée le nuage de points en utilisant la fonction **scatter ()** qui va prendre comme paramètre l’apprentissage ou le test et la prédiction.

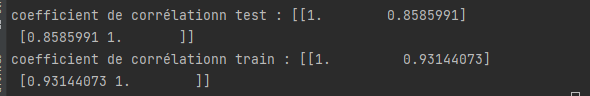
Les points bleus concernent les prédictions par rapport au test et les points oranges les prédictions par rapport à l’apprentissage.



**Le coefficient de corrélation :**

Pouur tracer le coefficient de corrélation nous avons utilisé la bibliothèque numpy eet avons fait appel à la fonction corrcoef (). Nous lui avons donné en paramètre la valeur ou test ou de l’apprentissage selon le cas étudié et la prédictiion associée au cas.





**CONCLUSION**

Ce projet nous a permis de mieux comprendre les concepts vus dans le cours. Il nous a permis d’éclaircir les points qui étaient encore mal compris mais aussi de faire des recherches et de voir de nouvelles choses en rapport avec le cours. Nous avons eu des difficultés à réussir certaines actions mais grâce au tutoriels suivis et aux explications du cours nous sommes arrivés à les surmonter.