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 :

```
import ...

path = os.path.join(os.path.dirname(__file__), '../csv/sonar.all-data.csv')

observer = pandas.read_csv(path)
"""test"""
print(observer.columns.values)
```

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

```
['0.0200' '0.0371' '0.0428' '0.0207' '0.0954' '0.0986' '0.1539' '0.1601'
 '0.3109' '0.2111' '0.1609' '0.1582' '0.2238' '0.0645' '0.0660' '0.2273'
 '0.3100' '0.2999' '0.5078' '0.4797' '0.5783' '0.5071' '0.4328' '0.5550'
 '0.6711' '0.6415' '0.7104' '0.8080' '0.6791' '0.3857' '0.1307' '0.2604'
 '0.5121' '0.7547' '0.8537' '0.8507' '0.6692' '0.6097' '0.4943' '0.2744'
 '0.0510' '0.2834' '0.2825' '0.4256' '0.2641' '0.1386' '0.1051' '0.1343'
 '0.0383' '0.0324' '0.0232' '0.0027' '0.0065' '0.0159' '0.0072' '0.0167'
 '0.0180' '0.0084' '0.0090' '0.0032' 'R']
PS C:\cours UT\IA\IA_M1_2021_12_28\sonar>
```

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"])

```python
path = os.path.join(os.path.dirname(__file__), '../csv/sonar.all-data.csv')

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"])
"""main"""
# print(observer.columns.values)
# print(observer.shape)

# Deactivation of the maximum number of columns of the dataframe to be displayed
pandas.set_option('display.max_columns', None)

# Display of the first line
print(observer.head(1))
```

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

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\sonar> py app.py
     F1      F2      F3      F4      F5      F6      F7      F8      F9  \
0  0.02  0.0371  0.0428  0.0207  0.0954  0.0986  0.1539  0.1601  0.3109

      F10     F11      F12     F13     F14     F15     F16   F17     F18  \
0  0.2111  0.1609  0.1582  0.2238  0.0645  0.066  0.2273  0.31  0.2999

      F19     F20     F21     F22     F23     F24     F25     F26     F27  \
0  0.5078  0.4797  0.5783  0.5071  0.4328  0.555  0.6711  0.6415  0.7104

     F28     F29     F30     F31     F32     F33     F34     F35     F36  \
0  0.808  0.6791  0.3857  0.1307  0.2604  0.5121  0.7547  0.8537  0.8507

      F37     F38     F39     F40    F41     F42     F43     F44     F45  \
0  0.6692  0.6097  0.4943  0.2744  0.051  0.2834  0.2825  0.4256  0.2641

      F46     F47     F48     F49     F50     F51     F52     F53     F54  \
0  0.1386  0.1051  0.1343  0.0383  0.0324  0.0232  0.0027  0.0065  0.0159

      F55     F56    F57     F58    F59     F60 OBJET
0  0.0072  0.0167  0.018  0.0084  0.009  0.0032     R
```

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())
```

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\sonar> py app.py
               F2          F3          F4          F5          F6          F7
count  208.000000  208.000000  208.000000  208.000000  208.000000  208.000000
mean     0.038437    0.043832    0.053892    0.075202    0.104570    0.121747
std      0.032960    0.038428    0.046528    0.055552    0.059105    0.061788
min      0.000600    0.001500    0.005800    0.006700    0.010200    0.003300
25%      0.016450    0.018950    0.024375    0.038050    0.067025    0.080900
50%      0.030800    0.034300    0.044050    0.062500    0.092150    0.106950
75%      0.047950    0.057950    0.064500    0.100275    0.134125    0.154000
max      0.233900    0.305900    0.426400    0.401000    0.382300    0.372900
```

Combien d'exemples ?

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\sonar> py app.py
(208, 61)
```

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))
```

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\sonar> py app.py
OBJET
M    111
R     97
dtype: int64
```

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 :

```
# Split dataset into train and test
array = observer.values

# Convertion des données en type decimal
X = array[:, 0:-1].astype(float)

# On choisit la dernière colonne comme feature de prédiction
Y = array[:, -1]

# Création des jeux d'apprentissage et de tests
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42)
```

Le score obtenu en apprentissage :

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\sonar> py app.py
Train score : 1.0
```

Le score obtenu en test :

```
Test score : 0.8846153846153846
```

Affichons la matrice de confusion en utilisant la commande :
print(confusion_matrix(predictions,Y_test))

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\sonar> py app.py
[[25  3]
 [ 5 19]]
```

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.

```
def accuracy(k, x_train, y_train, x_test, y_test):
    """
    compute accuracy of the classification based on k values
    """
    # instantiate learning model and fit data
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(x_train, y_train)

    # predict the response
    pred = model.predict(x_test)

    # evaluate and return  accuracy
    return accuracy_score(y_test, pred)
```

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.

```
"""Tableau de K"""
tab_k = []
for item in range(1, int(rows_nbr / 2)):
    tab_k.append(item)

print("Tab K")
print(tab_k)
print("----------------------------------------------------------\n
```

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.

```
"""Tableau de Score"""
rows_nbr = observer.shape[0]
tab_score = np.array([accuracy(k, X_train, Y_train, X_test, Y_test)
                      for k in range(1, int(rows_nbr / 2))])
print("Tab Score")
print(tab_score)
print("------------------------------------------------------------\n")
```

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

```
plt.plot(tab_score, linewidth=2)
plt.title("la courbe de k en fonction des scores", fontsize=16)
plt.xlabel("Nombre d'iterations", fontsize=14)
plt.ylabel("Score", fontsize=14)
plt.axis([0, 104, 0, 1])
plt.grid()
plt.show()
```

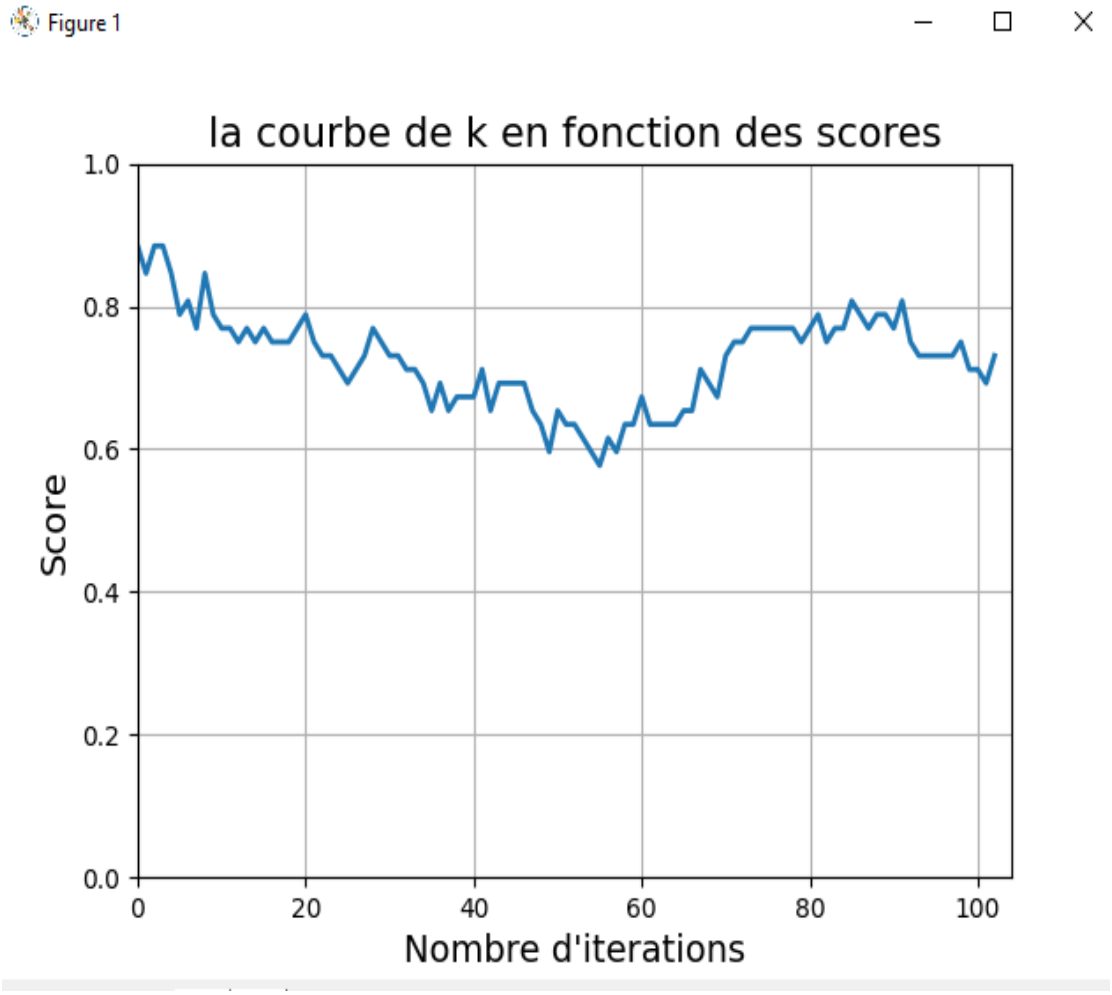


Figure 1
la courbe de k en fonction des scores
Score
Nombre d'iterations
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

TP2 : La régression linéaire

Pour ce TP, nous allons travailler sur la base Pokemon_dataset. Commençons d'abord par l'importer.

```
import os

import pandas

# Deactivation of the maximum number of columns of the dataframe to be displayed
pandas.set_option('display.max_columns', None)

path = os.path.join(os.path.dirname(__file__), '../csv/Pokemon_dataset.csv')

observer = pandas.read_csv(path)
```

Afficher les premières observations de la base de données ?

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

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\pokemon> py app.py
   Unnamed: 0  NUMERO         NOM TYPE_1  TYPE_2  POINTS_DE_VIE  \
0           0       1  Bulbizarre  Herbe  Poison             45

   NIVEAU_ATTAQUE  NIVEAU_DEFENSE  NIVEAU_ATTAQUE_SPECIALE  \
0              49              49                       65

   NIVEAU_DEFENSE_SPECIALE  VITESSE  GENERATION  LEGENDAIRE  Premier_Pokemon  \
0                       65       45           1           0             37.0

   Second_Pokemon  NBR_COMBATS  NBR_VICTOIRES  POURCENTAGE_DE_VICTOIRE
0            37.0        133.0           37.0                 0.278195
```

Combien de caractéristiques descriptives ? De quels types ?

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

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\pokemon> py app.py
       Unnamed: 0    NUMERO  POINTS_DE_VIE  NIVEAU_ATTAQUE  NIVEAU_DEFENSE  \
count    800.0000  800.0000     800.000000      800.000000      800.000000
mean     399.5000  400.5000      69.258750       79.001250       73.842500
std      231.0844  231.0844      25.534669       32.457366       31.183501
min        0.0000    1.0000       1.000000        5.000000        5.000000
25%      199.7500  200.7500      50.000000       55.000000       50.000000
50%      399.5000  400.5000      65.000000       75.000000       70.000000
75%      599.2500  600.2500      80.000000      100.000000       90.000000
max      799.0000  800.0000     255.000000      190.000000      230.000000

       NIVEAU_ATTAQUE_SPECIALE  NIVEAU_DEFENSE_SPECIALE     VITESSE  \
count               800.000000               800.000000  800.000000
mean                 72.820000                71.902500   68.277500
std                  32.722294                27.828916   29.060474
min                  10.000000                20.000000    5.000000
25%                  49.750000                50.000000   45.000000
50%                  65.000000                70.000000   65.000000
75%                  95.000000                90.000000   90.000000
max                 194.000000               230.000000  180.000000

       GENERATION  LEGENDAIRE  Premier_Pokemon  Second_Pokemon  NBR_COMBATS  \
count   800.00000   800.00000       783.000000      783.000000   783.000000
mean      3.32375     0.08125        63.856960       63.856960   127.541507
std       1.66129     0.27339        32.925941       32.925941    11.397402
```


```
       GENERATION  LEGENDAIRE  Premier_Pokemon  Second_Pokemon  NBR_COMBATS  \
count   800.00000   800.00000       783.000000      783.000000   783.000000
mean      3.32375     0.08125        63.856960       63.856960   127.541507
std       1.66129     0.27339        32.925941       32.925941    11.397402
```


```
min       1.00000     0.00000         3.000000        3.000000    92.000000
25%       2.00000     0.00000        36.000000       36.000000   120.000000
50%       3.00000     0.00000        62.000000       62.000000   128.000000
75%       5.00000     0.00000        91.000000       91.000000   135.000000
max       6.00000     1.00000       152.000000      152.000000   164.000000

       NBR_VICTOIRES  POURCENTAGE_DE_VICTOIRE
count     783.000000               783.000000
mean       63.856960                 0.501538
std        32.925941                 0.254993
min         3.000000                 0.021739
25%        36.000000                 0.284228
50%        62.000000                 0.491071
75%        91.000000                 0.717644
max       152.000000                 0.984496
```

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

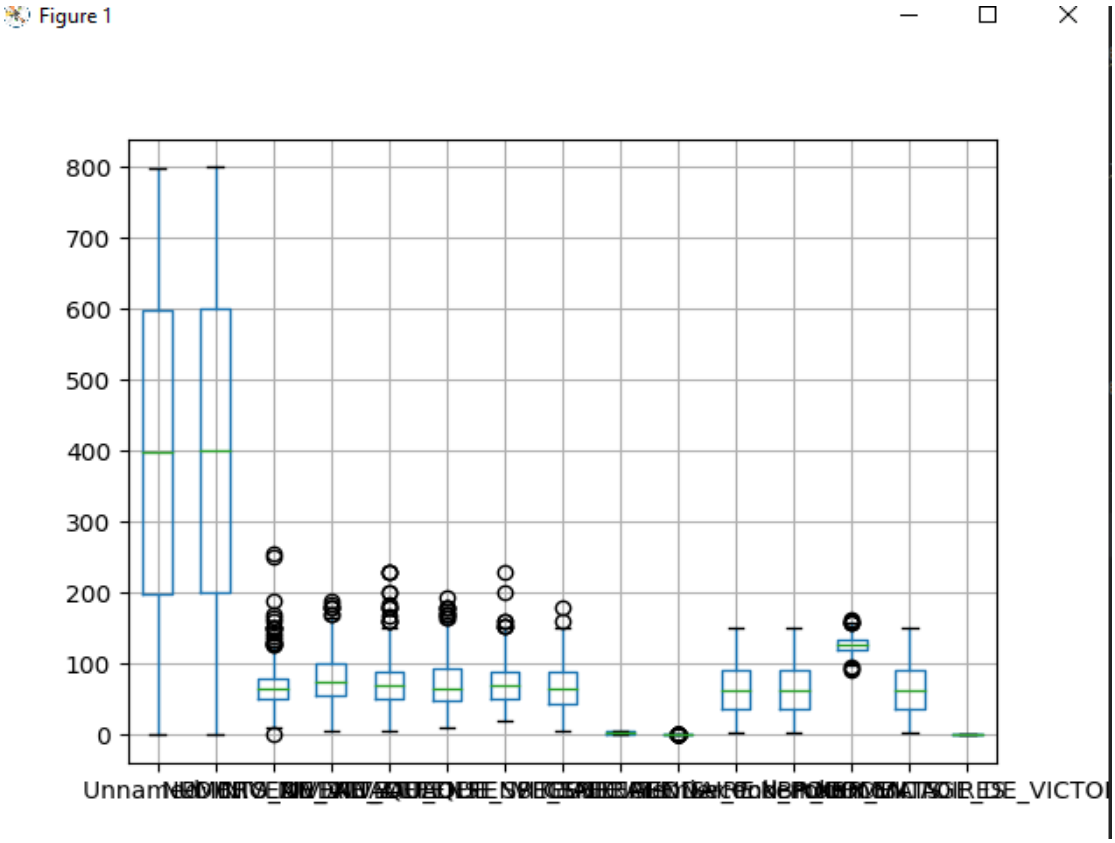
La moyenne de victoires est de 50% pour les pokemons

Comment sont organisés les exemples ?

Tracer les boxplots de toutes les variables

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

```
#Tracer les boxplots de toutes les variables
observer.boxplot()
```



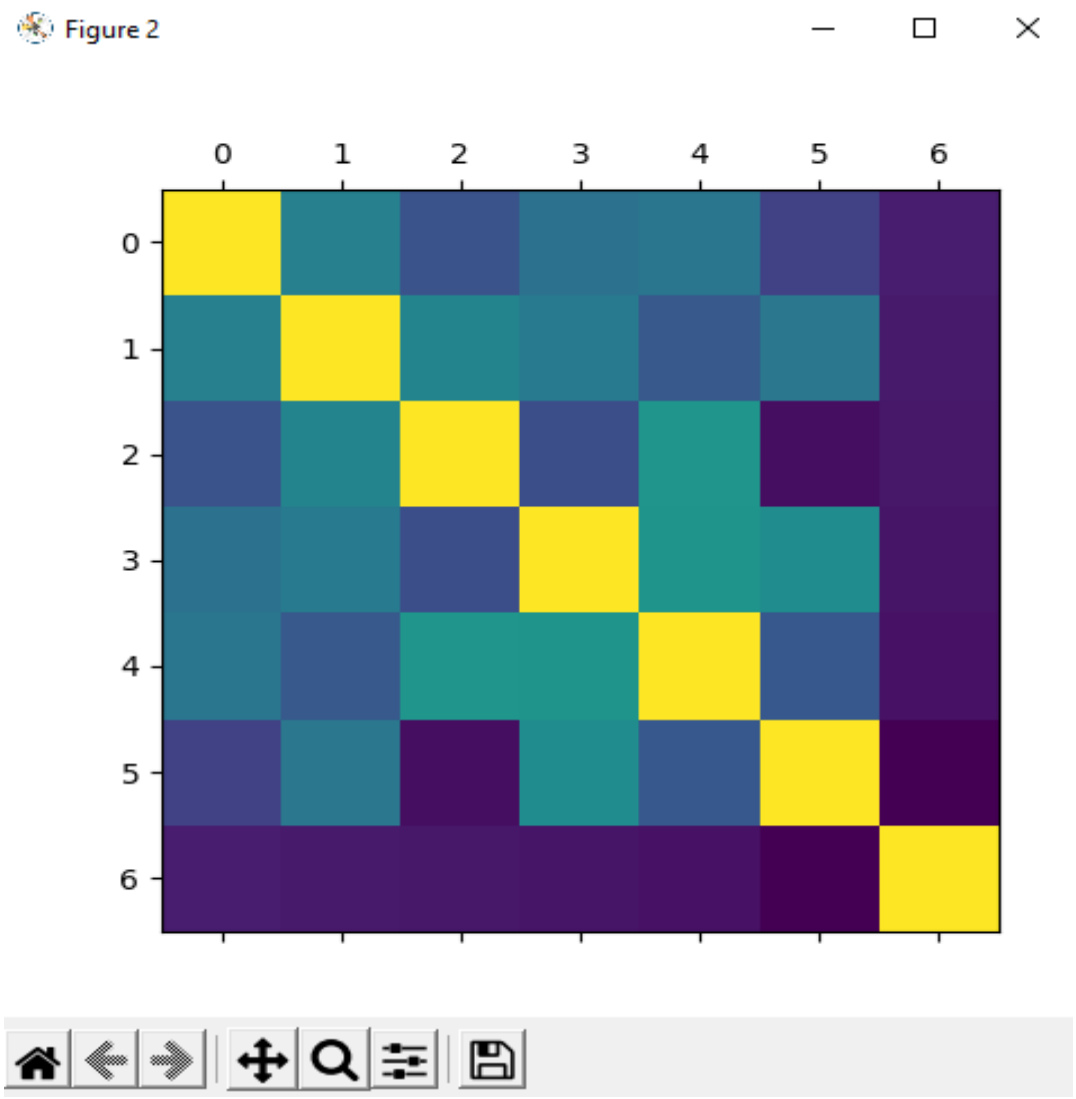
Calculer et tracer la matrice de corrélation des différentes features

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.

```
#Calculer et tracer la matrice de corrélation des différentes features
features = observer[["POINTS_DE_VIE", "NIVEAU_ATTAQUE", "NIVEAU_DEFENSE", "NIVEAU_ATTAQUE_SPECIALE",
                     "NIVEAU_DEFENSE_SPECIALE", "VITESSE","GENERATION"]]
var = features.corr()
print("FEATURES")
print(var)
print("----------------------------------------------------------\n\n----------------------------------------------------------")
plt.matshow(var)
plt.show()
```

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.

```
                       POINTS_DE_VIE  NIVEAU_ATTAQUE  NIVEAU_DEFENSE  \
POINTS_DE_VIE               1.000000        0.422386        0.239622
NIVEAU_ATTAQUE              0.422386        1.000000        0.438687
NIVEAU_DEFENSE              0.239622        0.438687        1.000000
NIVEAU_ATTAQUE_SPECIALE     0.362380        0.396362        0.223549
NIVEAU_DEFENSE_SPECIALE     0.378718        0.263990        0.510747
VITESSE                     0.175952        0.381240        0.015227
GENERATION                  0.058683        0.051451        0.042419

                       NIVEAU_ATTAQUE_SPECIALE  NIVEAU_DEFENSE_SPECIALE  \
POINTS_DE_VIE                         0.362380                 0.378718
NIVEAU_ATTAQUE                        0.396362                 0.263990
NIVEAU_DEFENSE                        0.223549                 0.510747
NIVEAU_ATTAQUE_SPECIALE               1.000000                 0.506121
NIVEAU_DEFENSE_SPECIALE               0.506121                 1.000000
VITESSE                               0.473018                 0.259133
GENERATION                            0.036437                 0.028486

                        VITESSE  GENERATION
POINTS_DE_VIE          0.175952    0.058683
NIVEAU_ATTAQUE         0.381240    0.051451
NIVEAU_DEFENSE         0.015227    0.042419
NIVEAU_ATTAQUE_SPECIALE  0.473018    0.036437
NIVEAU_DEFENSE_SPECIALE  0.259133    0.028486
```



Les valeurs obtenues en apprentissage :

```
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\pokemon> py app.py
Train score : 0.8675818388650588
```

Les valeurs obtenues en test :

```
Test score : 0.7342234153524025
(venv) PS C:\Users\Levono\Desktop\Projet IA\IA_M1_2021_12_28\pokemon>
```

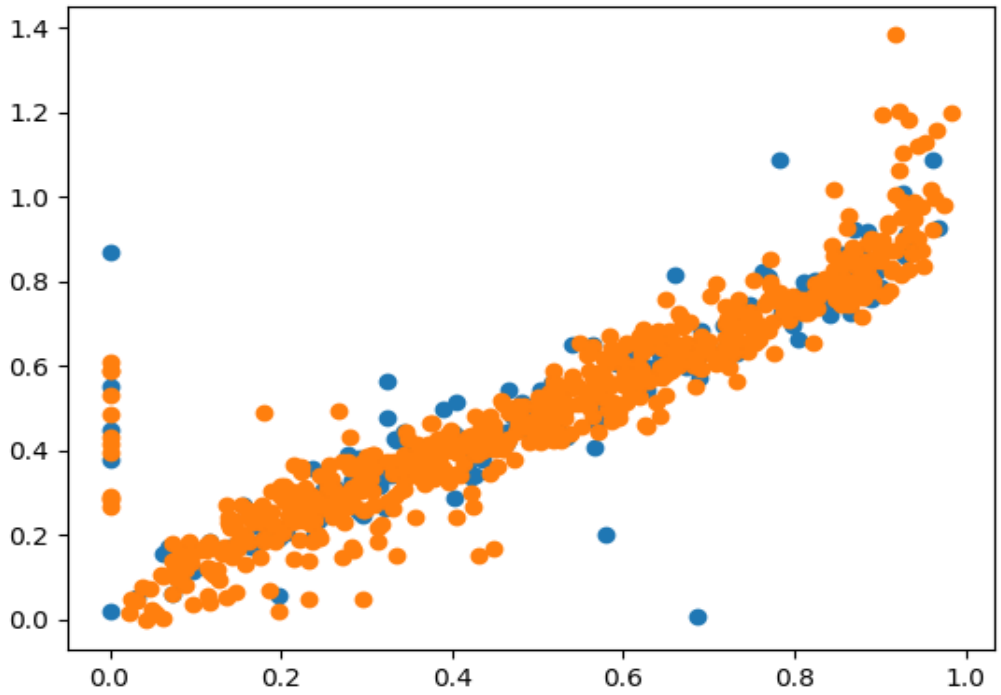
Le nuage de points :

```
# Création des jeux d'apprentissage et de tests
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
Y_test = numpy.nan_to_num(Y_test)
Y_train = numpy.nan_to_num(Y_train)
# Choix de l'algorithme
algo = LinearRegression()
# Apprentissage à l'aide de la fonction fit
algo.fit(X_train, Y_train)
print("Train score : " + str(algo.score(X_train, Y_train)))
print("Test score : " + str(algo.score(X_test, Y_test)))

#test prediction
predictions = algo.predict(X_test)
plt.scatter(Y_test, predictions)

#train prediction
prediction = algo.predict(X_train)
plt.scatter(Y_train, prediction)
#affichage des nuages de points
plt.show()
```

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.

```
#coefficients de corrélation
print("coefficient de corrélationn test : " +str(numpy.corrcoef(Y_test,predictions, rowvar=False)))
print("coefficient de corrélationn train : " +str(numpy.corrcoef(Y_train,prediction, rowvar=False)))
```

```
coefficient de corrélationn test : [[1.        0.8585991]
 [0.8585991 1.       ]]
coefficient de corrélationn train : [[1.         0.93144073]
 [0.93144073 1.        ]]
```

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.