



Université Cadi Ayyad



Ecole Nationale des sciences appliquées-SAFI

Année Universitaire 2021-2022

Filière : Génie Informatique et Intelligence Artificielle

Projet : Machine learning

Classification d'une base de données avec python

Réalisé par :

Monir EL OUARROUDI

Abdelghani AABA

Oussama DARFI

Sous la supervision de :

M. Mohammed Mediafi

2021/2022

Table des matières

Introduction générale.....	3
Chapitre 1 :	4
1. 1 -Le domaine de l'intelligence artificielle.....	5
1. 2 -Machine Learning.....	6
1. 2. 1 -Définition.....	6
1. 2. 2 -Les principaux algorithmes de machine Learning.....	6
Chapitre 2 : Classification des données.....	8
2. 1 -Les techniques de classification	9
2. 1. 1 -L'apprentissage supervisé.....	9
2. 2 -La différence entre l'apprentissage supervisé et non-supervisé.....	9
Chapitre 3 : Expérimentation et résultats	11
3. 1 -introduction	12
3. 2 -remarque.....	12
3. 3 -tensorflow.....	13
3. 4 –scikit-learn	19
Webographie	26

Introduction générale:

Dans le cadre de notre première année du cycle ingénieur en Génie informatique et intelligence artificielle, il nous est proposé un projet, dans le domaine de machine learning, permettant de faire la classification d'une base de données sous le langage de programmation Python, en utilisant les deux bibliothèques **Scikit-learn** et **Tensorflow**.

L'intelligence artificielle comme on la connaît est un domaine qui chaque jour évolue un peu plus en s'appropriant des capacités cognitives humaines, en les développant et parfois même en surpassant ce que les meilleurs êtres humains dans leur domaine sont capables de réaliser.

Ce projet nous a donné l'opportunité de cumuler nos connaissances théoriques, et également de rentrer dans la vie active et la découverte du processus des projets dans les milieux professionnels .

Chapitre 1

1. 1 Le domaine de l'intelligence artificielle

On utilise le terme « d'intelligence artificielle » ou d'IA pour désigner les ordinateurs et programmes informatiques capables de performances habituellement associées à l'intelligence humaine.

L'intelligence artificielle peut se définir comme l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence, soit des ordinateurs ou des programmes avec des puissances de calcul capables de performances habituellement associées à l'intelligence humaine, et amplifiées par la technologie :

- Capacité de raisonner.
- Capacité de traiter de grandes quantités de données.
- Faculté de discerner des patterns et des modèles indétectables par un humain.
- Aptitude à comprendre et analyser ces modèles.
- Capacités à interagir avec l'homme.
- Faculté d'apprendre progressivement.
- Et d'améliorer continuellement ses performances.

La révolution actuelle de l'intelligence artificielle et de la science qui en découle est rendue possible par une combinaison de 3 facteurs :

1. Une vaste quantité de données.
2. Une puissance informatique extraordinaire.
3. Les algorithmes révolutionnaires basés sur le deep learning.

L'IA est présente dans notre quotidien. Elle est par exemple utilisée par les services de détection des fraudes des établissements financiers, pour la prévision des intentions d'achat et dans les interactions avec les services clients en ligne.

1. 2 Machine Learning

1. 2.1 définition

Le Machine Learning ou apprentissage automatique est un domaine scientifique, et plus particulièrement une sous-catégorie de l'intelligence artificielle. Elle consiste à laisser des algorithmes découvrir des « patterns », à savoir des motifs récurrents, dans les ensembles de données. Ces données peuvent être des chiffres, des mots, des images, des statistiques...

Tout ce qui peut être stocké numériquement peut servir de données pour le Machine Learning. En décelant les patterns dans ces données, les algorithmes apprennent et améliorent leurs performances dans l'exécution d'une tâche spécifique.

En effet, les algorithmes de machine learning apprennent de manière autonome à effectuer une tâche ou à réaliser des prédictions à partir de données et améliorent leurs performances au fil du temps. Une fois entraîné, l'algorithme pourra retrouver les patterns dans de nouvelles données.

1. 2.2 Les principaux algorithmes de Machine Learning

Il existe une large variété d'algorithmes de Machine Learning. Certains sont toutefois plus couramment utilisés que d'autres. Tout d'abord, différents algorithmes sont utilisés pour les données étiquetées.

Les algorithmes de régression, linéaire ou logistique, permettent de comprendre les relations entre les données. La régression linéaire est utilisée pour prédire la valeur d'une variable dépendante base sur la valeur d'une variable indépendante. Il s'agirait par exemple de prédire

les ventes annuelles d'un commercial en fonction de son niveau d'études ou de son expérience.

La régression logistique est quant à elle utilisée quand les variables dépendantes sont binaires. Un autre type d'algorithme de régression appelé machine à vecteur de support est pertinent quand les variables dépendantes sont plus difficiles à classifier.

Un autre algorithme ML populaire est l'arbre de décision. Cet algorithme permet d'établir des recommandations basées sur un ensemble de règles de décisions en se basant sur des données classifiées. Par exemple, il est possible de recommander sur quelle équipe de football parier en se basant sur des données telles que l'âge des joueurs ou le pourcentage de victoire de l'équipe.

Pour les données non étiquetées, on utilise souvent les algorithmes de « clustering ». Cette méthode consiste à identifier les groupes présentant des enregistrements similaires et à étiqueter ces enregistrements en fonction du groupe auquel ils appartiennent.

Chapitre 2

1. 1 Les techniques de classification

Dans les modèles de classification, la valeur de sortie est discrète. Ceci implique que le modèle est entraîné de telle manière à restituer une valeur de sortie discrète (y) pour une valeur d'entrée (x). Cette valeur de sortie correspond donc à une classe ou une étiquette, reflétant le terme de « classification ».

Les modèles de classification sont le plus communément utilisés pour la classification d'images, la classification de documents, le traitement automatique de langage naturel (ou NLP) ou encore pour la détection de fraude.

1.2 Apprentissage supervisé

Dans le cas de l'apprentissage supervisé, le plus courant, les données sont étiquetées afin d'indiquer à la machine quelles patterns elle doit rechercher.

Le système s'entraîne sur un ensemble de données étiquetées, avec les informations qu'il est censé déterminer. Les données peuvent même être déjà classifiées de la manière dont le système est supposé le faire.

Cette méthode nécessite moins de données d'entraînement que les autres, et facilite le processus d'entraînement puisque les résultats du modèle peuvent être comparés avec les données déjà étiquetées. Cependant, l'étiquetage des données peut se révéler onéreux. Un modèle peut aussi être biaisé à cause des données d'entraînement, ce qui impactera ses performances par la suite lors du traitement de nouvelles données.

2.1 La différence entre l'apprentissage supervisé et non-supervisé

A la différence de l'apprentissage supervisé qui fait appel à des données étiquetées ou annotées pour réaliser des prédictions, l'apprentissage non

supervisé n'a pas besoin d'étiquette. Puisque les données ne sont pas étiquetées, il n'est pas possible pour le modèle de calculer des scores de réussite. En conséquence, alors que les systèmes supervisés se concentrent sur les tâches de régression et classification, la technique non supervisée est utilisée pour effectuer des regroupements ou des mises en grappe de données en fonction de leurs ressemblances ou différences.

Chapitre 3

3.1 introduction

L'algorithme de classification est une technique d'apprentissage supervisé qui est utilisée pour identifier la catégorie de nouvelles observations sur la base de données d'apprentissage. Dans la classification, un programme apprend à partir d'un ensemble de données ou d'observations données, puis classe les nouvelles observations dans un certain nombre de classes ou de groupes. Par exemple, Oui ou Non, 0 ou 1, Spam ou Non Spam, chat ou chien, etc. Les classes peuvent être appelées cibles/étiquettes ou catégories.

Les algorithmes de classification peuvent être divisés en deux catégories principales :

Modèles linéaires :

- Régression logistique
- Support Vector Machines – SVM

Modèles non linéaires :

- Naïve Bayes (réseaux bayésiens)
- Classification par arbre de décision
- Classification Random Forest

Nous allons maintenant vous présenter le projet qui consiste à faire la classification des yeux dans une base de données, nous allons déterminer à quelle personne appartient chaque œil. Pour cela nous allons procéder en étapes :

3.2 remarque

On a expliqué toutes les classes et toutes méthodes utilisée, et on a fait tous les détails dans des commentaires au sein du code.

3.3Tensorflow

- Étape 1 : Charger l'ensemble de bibliothèques

```
1 import tensorflow as tf
2 from tensorflow import keras # TensorFlow est un outil open source
3                               # d'apprentissage automatique developpe par Google.
4
5 import numpy as np # bibliotheque nmpy permet d'effectuer des calculs numeriques avec Python.
6                     # Elle introduit une gestion facilitee des tableaux de nombres.
7 from PIL import Image # pour traitement et conversion des images
8 from matplotlib import pyplot # pour afficher l'image sur un graphe
9 from os import listdir # il permet de gérer l'arborescence des fichiers (boucler sur le fichier)
```

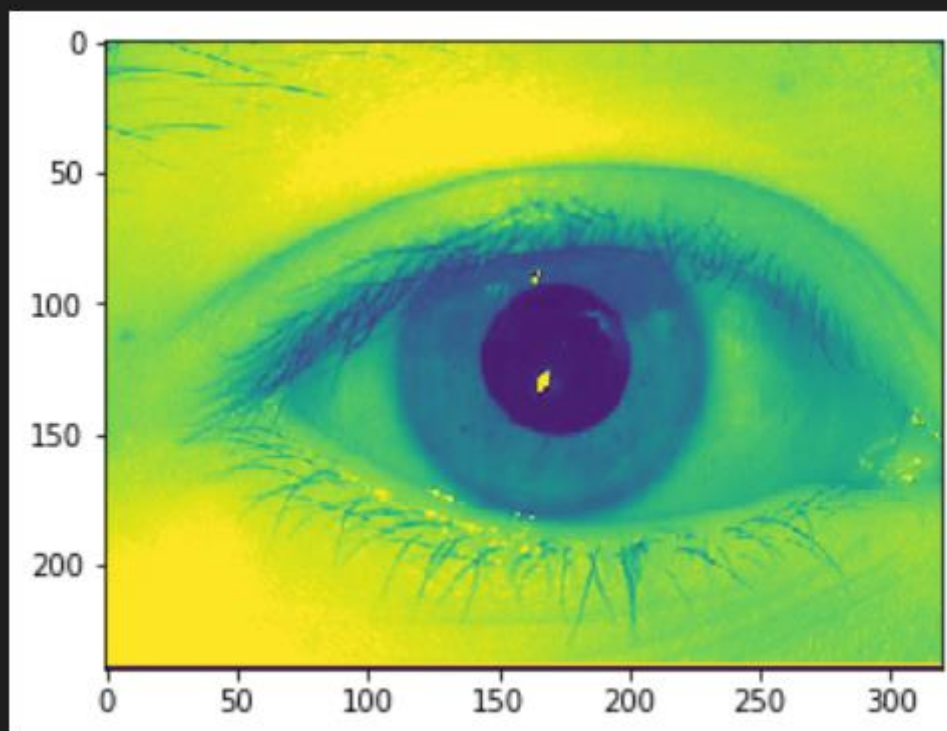
- Étape 2 : Charger l'ensemble de données et affichage d'un exemple

```
1 """=====
2     Chargement de la base de données
3     ====="""
4 X_train = np.zeros((460, 240, 320)) # une matrice qui a 460 matrice du taille 240 lignes*320 colonnes (les photos)
5 Y_train = np.zeros((460)) # une matrice de 460 lignes et 1 colonne, contient les resultats attendus
6
7 i = 0 # un parametre pour boucler sur 460 images
8
9 for k in range(1, 47): # 46 personnes
10     folder_left = "C:/Users/abdou/Desktop/New folder/MMU/" + str(k) + "/left"
11     folder_right = "C:/Users/abdou/Desktop/New folder/MMU/" + str(k) + "/right"
12
13     for j in os.listdir(folder_left): # os.listdir liste les fichiers qui sont au sein du folder_left
14         if j.endswith('.bmp'): # tester c'est le fichier est un photo
15             #ouvrir chaque image, convertir en blanc & noir,
16             #puis la stoker dans la variable img
17             img = np.array(Image.open(folder_left + '/' + j).convert(mode = 'L'))
18             X_train[i] = img
19             Y_train[i] = k - 1
20             i += 1
21
22     # maintenant pour les fichiers des yeux droite.
23     for j in os.listdir(folder_right):
24         if j.endswith('.bmp'):
25             img = np.array(Image.open(folder_right + '/' + j).convert(mode = 'L'))
26             X_train[i] = img
27             Y_train[i] = k - 1
28             i += 1
29
30
31 X_train = X_train.astype(int) # changer le type
32 Y_train = Y_train.astype(int)
```



```
1 print("Photo 459 dans X_train:")
2 pyplot.imshow(X_train[459], interpolation="nearest")
3 # afficher la derniere image avec la fonction de pyplot:imshow
4 pyplot.show()
```

Photo 459 dans X_train:




- Étape 3 : adapter les données et crée un dataset

```
1  #--preprocess
2  #changer le type de X_train en float et les diviser sur 255.0
3  #changer le type de Y_train
4
5  def preprocess(x, y):
6      x = tf.cast(x, tf.float32) / 255.0
7      y = tf.cast(y, tf.int64)
8      return x, y
9
10 # ---create_dataset
11 # --one_hot-- convertir chaque l'element de vecteur Y_train en vecteur
12 # de 46 zero sauf le numero du personne qui sera remplacer par 1.
13 # et cree une dataset a partir de la bibliotheque tensorflow
14 # --shuffle-- melanger la place des personne (par exemple le deuxieme personne )
15 def create_dataset(xs, ys, n_classes=46):
16     ys = tf.one_hot(ys, depth=n_classes)
17     return tf.data.Dataset.from_tensor_slices((xs, ys)) \
18         .map(preprocess) \
19         .shuffle(len(ys)) \
20         .batch(128)
21
22 dataset = create_dataset(X_train, Y_train)
```

- Étape 4 : créer un réseau de neurones et l'entraîner


```
1  #cree un réseau de neurones de 4 cauches et chaque une de cette mots en question contient 256,192,128,46 neurones.
2  #la dimension des donnees d'entree sont : (240,)
3  model = keras.Sequential([
4      keras.layers.Reshape(target_shape=(240 * 320,), input_shape=(240, 320)),
5      keras.layers.Dense(units=256, activation='relu'),
6      keras.layers.Dense(units=192, activation='relu'),
7      keras.layers.Dense(units=128, activation='relu'),
8      keras.layers.Dense(units=46, activation='sigmoid')
9  ])
```



```

1  #definir les parametre l'optimisation et la fonction loss du model
2  model.compile(optimizer='adam',
3                loss=tf.losses.CategoricalCrossentropy(from_logits=True),
4                metrics=['accuracy'])
5  # entraînement du model
6  history = model.fit(
7      dataset.repeat(),
8      epochs=10,
9      steps_per_epoch=100
10 )

```



```

Epoch 1/10
100/100 [=====] - 12s 116ms/step - loss: 6.7812 - accuracy: 0.1046
Epoch 2/10
100/100 [=====] - 12s 119ms/step - loss: 2.4596 - accuracy: 0.3539
Epoch 3/10
100/100 [=====] - 12s 119ms/step - loss: 1.3815 - accuracy: 0.6331
Epoch 4/10
100/100 [=====] - 12s 123ms/step - loss: 0.6698 - accuracy: 0.8370
Epoch 5/10
100/100 [=====] - 12s 124ms/step - loss: 0.2863 - accuracy: 0.9447
Epoch 6/10
100/100 [=====] - 13s 126ms/step - loss: 0.1548 - accuracy: 0.9680
Epoch 7/10
100/100 [=====] - 13s 129ms/step - loss: 0.0922 - accuracy: 0.9768
Epoch 8/10
100/100 [=====] - 13s 126ms/step - loss: 0.0664 - accuracy: 0.9770
Epoch 9/10
100/100 [=====] - 13s 128ms/step - loss: 0.0639 - accuracy: 0.9765
Epoch 10/10
100/100 [=====] - 13s 132ms/step - loss: 0.0533 - accuracy: 0.9771

```


- Étape 5 : la prédiction et la précision




```
1 #changer le type de X_train en float et les diviser sur 255.0
2 #changer le type de Y_train
3 #pour les utiliser dans le model
4 X_train = tf.cast(X_train, tf.float32) / 255.0
5 Y_train = tf.cast(Y_train, tf.int64)
```



```
1 #faire des prediction sur X_train et les stocker dans predictions
2 predictions = model.predict(X_train)
```



```
1 # one_hot fonction deja expliqué
2 # np.argmax prend l'indice du max dans le vecteur
3 print("La resultat attendu: (personne 43)")
4 print(np.argmax(tf.one_hot(Y_train, depth=46)[435]))
5 print()
6 print("La resultat obtenu:")
7 # et on le compare par la resultat obtenu
8 print(np.argmax(predictions[435]))
```




La resultat attendu: (personne 43)
43

La resultat obtenu:
43



```
1 # model.evaluate : donne le pourcentage du precision et du loss
2 score = model.evaluate(X_train, tf.one_hot(Y_train, depth=46), verbose = 0)
3 # afficher les deux pourcentages
4 print('Test loss:', score[0]*100 , "%")
5 print('Test accuracy:', score[1]*100 , "%")
```



```
Test loss: 4.885188490152359 %  
Test accuracy: 97.826087474823 %
```

3.4 scikit-learn

- [Étape 1](#) : Charger l'ensemble de bibliothèques



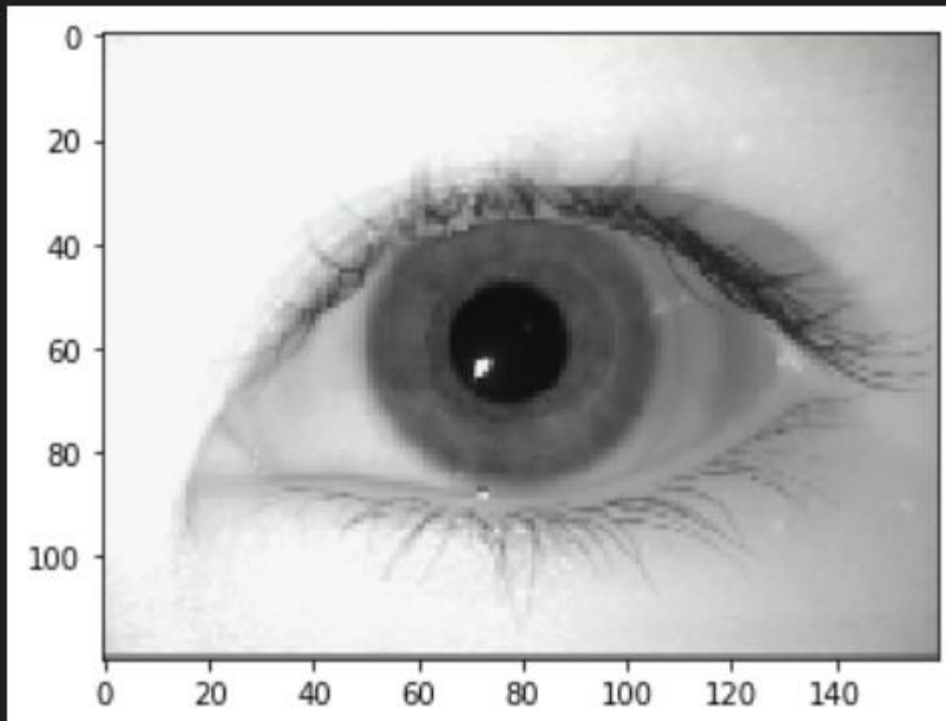
```
1 import sklearn # Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique.  
2 import sklearn.datasets # sklearn.datasets contient plusieurs datasets pour deep learning  
3 import numpy as np  
4  
5 from skimage.io import imread # pour ouvrir les images et les stocker dans des variables  
6 from skimage.transform import resize # pour changer les dimensions d'une photo  
7  
8 import pandas as pd # pour la visualisation des données, des tableaux  
9  
10 from matplotlib import pyplot # pour l'affichage des photos dans des figures  
11 from os import listdir # il permet de gérer l'arborescence des fichiers (boucler sur le fichier)
```

- Étape 2 : Charger l'ensemble de données et affichage d'un exemple

```
1  """=====
2      Chargement de la base de données "iris"
3  ===== """
4  X = []
5  Y = []
6
7  i = 0
8
9  for k in range(1, 47):
10     folder_left = "C:/Users/abdou/Desktop/New folder/MMU/" + str(k) + "/left"
11     folder_right = "C:/Users/abdou/Desktop/New folder/MMU/" + str(k) + "/right"
12
13     for j in os.listdir(folder_left):
14         if j.endswith('.bmp'):
15             img_resized = resize(imread(folder_left + '/' + j), (120, 160, 3))
16             img_array = img_resized.flatten()
17             X.append(img_array)
18             Y.append(k - 1)
19
20
21     for j in os.listdir(folder_right):
22         if j.endswith('.bmp'):
23             img_resized = resize(imread(folder_right + '/' + j), (120, 160, 3))
24             img_array = img_resized.flatten()
25             X.append(img_array)
26             Y.append(k - 1)
27
28
29  X = np.array(X)
30  Y = np.array(Y)
```

```
1  print("Photo 0 dans X:")
2  pyplot.imshow(X[0].reshape(120, 160, 3), interpolation="nearest")
3  pyplot.show()
```

Photo 0 dans X:



- Étape 3 : visualisation des images sous formes des matrices avec leurs résultats

```
1 # visualisation des données en utilisant pandas
2 dataframe = pd.DataFrame(X)
3 # Creation du colonne pour les resultats attendus
4 dataframe['Resultat'] = Y
5
6 dataframe
```

5	6	7	8	9	...	57591	57592	57593	57594	57595	57596	57597	57598	57599	Resultat
02	0.934740	0.948979	0.932675	0.925491	...	0.424197	0.429198	0.416266	0.423002	0.430275	0.415425	0.414501	0.424018	0.423087	0
03	0.923067	0.925075	0.909061	0.871459	...	0.421768	0.429897	0.420928	0.415067	0.419211	0.406658	0.406658	0.410999	0.405819	0
49	0.972549	0.972549	0.972549	0.972549	...	0.417013	0.424770	0.422341	0.415248	0.422849	0.407586	0.412165	0.411090	0.405819	0
22	0.946033	0.965547	0.945194	0.963955	...	0.424582	0.430979	0.415250	0.409083	0.423225	0.408334	0.426915	0.425840	0.427752	0
19	0.775158	0.792710	0.768061	0.939589	...	0.396388	0.404650	0.390130	0.396297	0.401473	0.396204	0.390128	0.397177	0.388452	0
...
02	0.652575	0.663171	0.650897	0.726388	...	0.532683	0.533524	0.531184	0.533345	0.536608	0.525593	0.532687	0.533900	0.533342	45
59	0.708218	0.715734	0.705884	0.724075	...	0.538479	0.541699	0.536984	0.532312	0.538947	0.536986	0.538569	0.538616	0.538479	45
86	0.658844	0.665012	0.652496	0.692788	...	0.532222	0.537777	0.536234	0.537066	0.538149	0.536143	0.538479	0.538195	0.537640	45
71	0.623338	0.634922	0.627169	0.676737	...	0.536137	0.540059	0.535387	0.536976	0.538055	0.536051	0.536978	0.537023	0.536137	45
22	0.638563	0.645567	0.632306	0.685819	...	0.534367	0.538245	0.528766	0.535116	0.535956	0.533620	0.534365	0.535161	0.534279	45

- Étape 4 : traitement des données

```

1 # Diviser les données : des données pour l'entraînement et autres pour l'évaluation
2 from sklearn.model_selection import train_test_split
3 # X : Les données d'entrée
4 # Y : Les données de sortie (les résultats attendus)
5 print("Les dimensions des données d'entrée:",X.shape)
6 print("Les dimensions des données de sortie:",Y.shape)
7 x_train,x_test,y_train,y_test = train_test_split(X,Y,shuffle=True,test_size = 0.2,random_state=109,stratify=Y)
8 print("Les dimensions des données d'entraînement d'entrée:",x_train.shape)
9 print("Les dimensions des données d'évaluation d'entrée:",x_test.shape)
10 print("Les dimensions des données d'entraînement de sortie:",y_train.shape)
11 print("Les dimensions des données d'évaluation d'entrée:",y_test.shape)

```

```

Les dimensions des données d'entrée: (460, 57600)
Les dimensions des données de sortie: (460,)
Les dimensions des données d'entraînement d'entrée: (368, 57600)
Les dimensions des données d'évaluation d'entrée: (92, 57600)
Les dimensions des données d'entraînement de sortie: (368,)
Les dimensions des données d'évaluation d'entrée: (92,)

```

- Étape 5 : création d'un réseau de neurone avec des meilleurs paramètres et l'entraîner

```
1 # SVM
2 from sklearn.model_selection import GridSearchCV
3 from sklearn.svm import SVC
4 from sklearn.model_selection import StratifiedKFold , KFold
5
6 # Des parametres pour le reseau de neurones
7 tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
8                       'C': [1, 10, 100, 1000]}]
9 inner_cv = KFold(n_splits=15, shuffle=False)
10
11 # GridSearchCV pour trouver les meilleurs parametres
12 # verbose : les pas pour trouver ces parametres
13 cv = GridSearchCV(SVC(), tuned_parameters, refit = True, verbose= 3, cv=inner_cv)
14 cv.fit(x_train, y_train)
```

```
Fitting 15 folds for each of 8 candidates, totalling 120 fits
[CV 1/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.320 total time= 26.5s
[CV 2/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.520 total time= 27.5s
[CV 3/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.400 total time= 28.8s
[CV 4/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.400 total time= 26.4s
[CV 5/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.480 total time= 26.9s
[CV 6/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.520 total time= 26.1s
[CV 7/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.480 total time= 26.0s
[CV 8/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.440 total time= 26.1s
[CV 9/15] END .....C=1, gamma=0.001, kernel=rbf;; score=0.708 total time= 26.1s
[CV 10/15] END ....C=1, gamma=0.001, kernel=rbf;; score=0.542 total time= 26.5s
...
[CV 12/15] END C=1000, gamma=0.0001, kernel=rbf;; score=0.792 total time= 17.5s
[CV 13/15] END C=1000, gamma=0.0001, kernel=rbf;; score=0.833 total time= 17.6s
[CV 14/15] END C=1000, gamma=0.0001, kernel=rbf;; score=0.667 total time= 18.4s
[CV 15/15] END C=1000, gamma=0.0001, kernel=rbf;; score=0.750 total time= 17.6s
```



```
1 # afficher les parametres selectionnées par GridSearchCV
2 print("Les meilleurs parameters sont:",cv.best_params_)
3 # Creation du model avec les meilleurs parametres trouvées
4 svm = cv.best_estimator_
```



```
Les meilleurs parameters sont: {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
```

- Étape 6 : la prédiction et la précision



```
1 # La prediction du model apres l'application des meilleurs parametres
2 y_prediction = svm.predict(x_test)
3 # afficher les resultats attendus et obtenus
4 print("Les resultats attendus: ",y_test)
5 print("Les resultats obtenus:",y_prediction)
```



```
Les resultats attendus: [26 23  5 37 21 20 36 40 41 16 24 27 10 10 38  6  7 40  1 22 45 39  8 35
19  2 25 34 42 44 31  8 14 41  9  2 12 44 11 39 38 32 23 30 17 45 11 35
17 13 18 43 26 43 37  9 13  7  0 12 31 15 21  3 33 25  5 30  0 15  4  4
 1 22 19  6 20 29 14  3 18 34 42 32 16 24 28 28 27 33 36 29]
Les resultats obtenus: [19 23  5 37 21 20 36 40 41 16 24 27 10 10 34 21  7 16 29 17 45 39  8 35
19  3 25 34 42 44 31  8 14 41  9  2 12 44 11 39 38 32 23 30 17 45 11 35
 2 13 18 43 26 43 37 25 13  9 13 12 31 15 21  2 33 25  5 30  1 15  4 36
31 22 19  6  1 27 14  2 18 34 42 32 16 24  6  7 27  3 36 20]
```

```
1 from sklearn.metrics import accuracy_score
2
3 print("Pourcentage de precision de ce model:")
4 print(accuracy_score(y_test, y_prediction)*100, "%")
5
6 # on peut ameliorer cette valeur par l'incrementation du nombre d'iterations (n_splits)
```

```
Pourcentage de precision de ce model:
76.08695652173914 %
```

Webographie :

- [1]. <https://www.netapp.com/fr/artificial-intelligence/what-is-artificial-intelligence/>
- [2]. <https://experiences.microsoft.fr/articles/intelligence-artificielle/>
- [3]. <https://www.journaledunet.fr/web-tech/guide-de-l-intelligence-artificielle/>