RAPPORT INTELLIGENCE ARTIFICIELLE
2021 – 2022

Nom : TOURE
Prénom : Boubacar
Numéro étudiant : 18 00 94 22
Niveau : Master 1 informatique
Date : 06 Avril 2022 à Angers
Professeur : Madame Béatrice DUVAL
Cours : Intelligence Artificielle
Objectif : Implémenter l'algorithme de sélection (relief) et le tester sur plusieurs jeux d'exemples.

FACULTÉ
DES SCIENCES
UNIVERSITÉ D'ANGERS

FACULTÉ
DES SCIENCES
UNIVERSITÉ D'ANGERS

SOMMAIRE

I. Introduction

■ Description

Dans le cadre de ce projet, nous avons pour but d'observer et d'expérimenter l'application de la méthode de sélections de variables pour les K Plus Proches Voisins. En intelligence artificielle, plus précisément en apprentissage automatique, la **méthode des *k* plus proches voisins** est une méthode d'apprentissage supervisé. Elle est aussi appelé k-NN ou KNN, de l'anglais *k-nearest neighbors*.

Pour réaliser cette expérience, nous allons observer expérimentalement comment ce phénomène se traduit sur différents jeux de données. Parmi les différents jeux de données qui seront traités dans notre projet, nous avons les jeux de données suivants :

- **Iris2Classes :** Traitement d'un problème de classification binaire afin d'assurer la reconnaissance des deux classes (Iris versicolor et Iris virginica). Puis nous appliquerons la méthode des KNN, avec K = 5 et nous vous exposerons nos constats.
- **HeartDisease :** Traitement d'un problème de classification binaire afin d'assurer la prédiction d'un problème cardiaque. Puis nous appliquerons la méthode des KNN, avec K = 5 et nous vous exposerons nos constats.
- **Diabète :** Traitement d'un problème de classification binaire afin d'assurer la reconnaissance du diabète en fonction des caractéristiques des patients. Puis nous appliquerons la méthode des KNN, avec K = 5 et nous vous exposerons nos constats.

■ Méthode de recherches appliquée

Pour la réalisation de notre projet, nous allons appliquer l'algorithme Relief [Kira and Rendell, 1992] afin de contourner la difficulté concernant la sélection de variables. Dans Relief, le critère d'intérêt d'un attribut augmente si les valeurs de cet attribut diffèrent plus sur des voisins de classes différentes que sur des voisins de même classe. Une fois le vecteur de poids calculé, on peut sélectionner les p attributs qui ont les poids les plus forts. Le poids d'un attribut augmente donc quand il a une valeur éloignée sur NearMiss. Le poids d'un attribut diminue donc quand il a une valeur éloignée sur NearHit. Dans la suite de notre traitement, nous appliqueront la technique de normalisation des attributs afin que les valeurs de chaque attribut puissent être comprises entre [0, 1].

Dans notre projet, nous allons appliquer des améliorations sur l'algorithme Relief. Ce qui permettra la sélection de k voisins au lieu d'un seul. On déterminera ainsi le barycentre des k voisins de même classe comme la valeur de la variable nearestHit dans le calcul de l'algorithme Relief, et le barycentre des k voisins de classe opposée comme valeur de la variable nearestMiss. Le paramètre m est remplacé par n le nombre d'instances (***voir Figure : Pseudo-Code de Relief*).**

■ Langage de programmation utilisé

Pour la réalisation de notre projet, nous avons décidé d'utiliser le langage **Python.** Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes.

Nous utiliserons aussi d'autres bibliothèques python afin d'assurer le traitement de nos données. Parmi les différentes bibliothèques utilisées, nous avons :

- **Numpy** : Est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.
- **Pandas** : Est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Il propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Il nous permet de lire et d'écrire dans des fichiers. Pour notre projet, Il nous a été très utile pour la récupération des données dans les fichiers CSV.
- **MatPlotLib** : Est une bibliothèque Python capable de produire des graphes de qualité. Il nous a été très utile pour la génération des graphes représentatifs des jeux de données que nous avions à étudier.

II. Développement

■ Structure du code

Dans notre projet, chaque fichier a été créé et positionné et à un emplacement bien précis. De ce fait, déplacer un seul fichier de son emplacement initiale pourrait avoir des répercutions sur l'exécution de notre code. C'est pourquoi, en cas d'usage de ce projet, il vous est conseillé de bien vouloir respecter la hiérarchie ainsi que l'emplacement de chaque fichier afin qu'il puisse être utiliser dans notre programme et assurer une bonne exécution du code ***(Voir l'image ci-dessous)***.

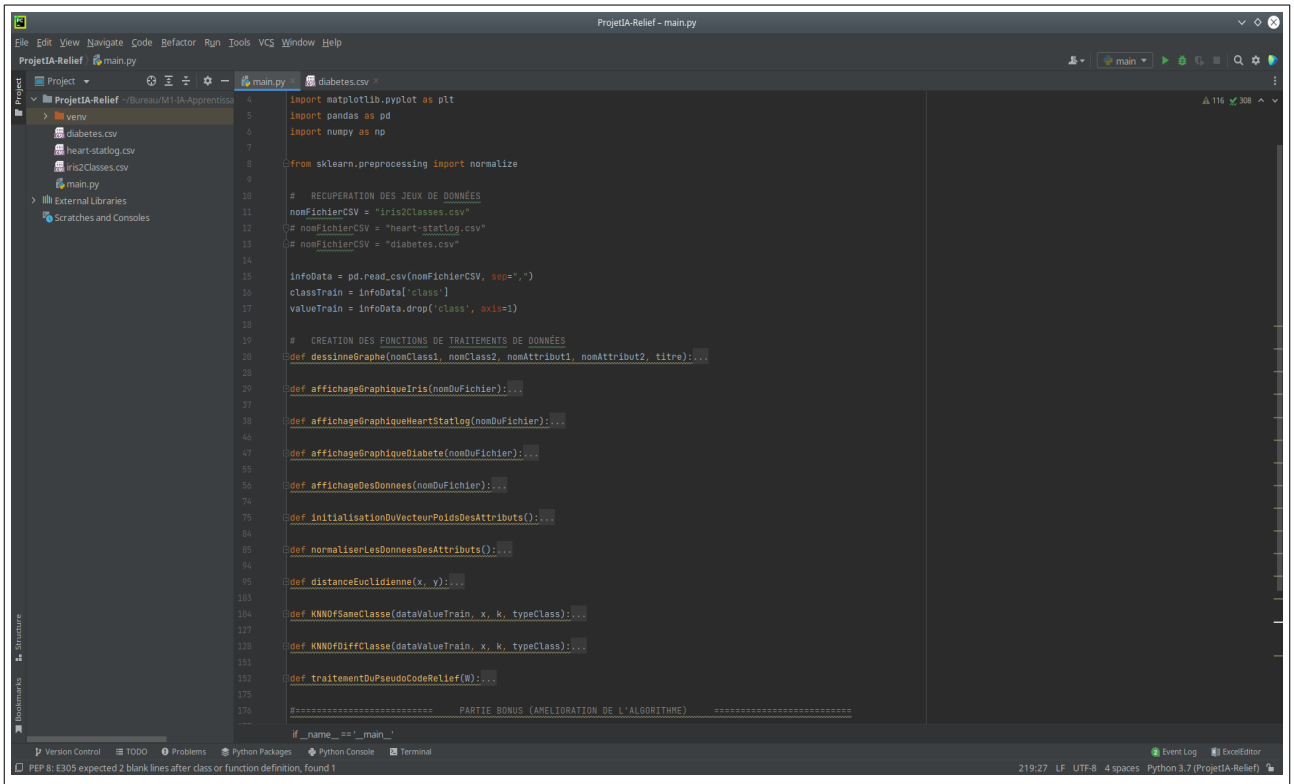


Figure : Plan du Code de l'algorithme

■ Fonctions et procédures déclarées

Les différentes fonctions et procédures déclarées pour la réalisation de notre projet sont :

- **dessinneGraphe(nomClass1, nomClass2, nomAttribut1, nomAttribut2, titre) :**

Cette fonction nous permet de dessiner un graphe en fonction : du nom des classes à déterminer (classification binaire), de deux attributs les plus importants assurant la classification des données fournis ainsi que du titre du graphe. Elle sera mise en place grâce à la bibliothèque Matplotlib.

- **affichageGraphiqueIris(nomDuFichier) :**

Cette fonction nous permet de dessiner le graphe correspondant à la classification binaire du jeu de données Iris2Classes.

- **affichageGraphiqueHeartStatlog(nomDuFichier) :**

Cette fonction nous permet de dessiner le graphe correspondant à la classification binaire du jeu de données HeartStatlog.

- **affichageGraphiqueDiabete(nomDuFichier) :**

Cette fonction nous permet de dessiner le graphe correspondant à la classification binaire du jeu de données Diabete.

- **affichageDesDonnees(nomDuFichier) :**

Cette fonction nous permet d'afficher toutes les informations concernant le jeu de données correspondant au nom du fichier passer en paramètre.

- **InitialisationDuVecteurPoidsDesAttributs() :**

Cette fonction nous permet d'initialiser les vecteurs de poids des attributs de notre jeu de données à 0.

- **normaliserLesDonneesDesAttributs() :**

Cette fonction nous permet de retourner un tableau (vecteur) contenant l'ensemble des données de notre jeu de données avec la valeur des attributs normalisées (comprise entre [0, 1]).

- **distanceEuclidienne(x, y) :**

Cette fonction nous permet de retourner la distance euclidienne entre deux vecteurs X et Y.

- **KNNOfSameClasse(dataValueTrain, x, k, typeClass) :**

Cette fonction nous permet de retourner un vecteur contenant l'ensemble des indices des K Plus proches voisin de même classe.

- **KNNOfDiffClasse(dataValueTrain, x, k, typeClass) :**

Cette fonction nous permet de retourner un vecteur contenant l'ensemble des indices des K Plus proches voisin de classe différente.

- **traitementDuPseudoCodeRelief(W) :**

Cette fonction nous permet de réaliser l'implémentation du pseudo code de l'algorithme de Relief avec W = le vecteur de poids initialisé.

Pour améliorer notre algorithme, nous allons utiliser les fonctions suivantes :

- **determineBarycentre(vecteurKNN, tabAttribut) :**

Cette fonction nous permet de déterminer un vecteur barycentre en ce basant sur le calcul des K plus proches voisins. On déterminera alors le barycentre des k voisins de même classe et il remplacera nearestHit ainsi que le barycentre des k voisins de classe opposée et il remplacera nearestMiss dans l'algorithme Relief. Le paramètre vecteurKNN peut contenir le résultat des fonctions KNNOfSameClasse KNNOfDiffClasse en fonction du calcul que vous souhaitez réaliser.

- **traitementDuPseudoCodeReliefAvecBarycentre(W, k) :**

Cette fonction nous permet de réaliser l'implémentation du pseudo code de l'algorithme de Rélief avec W = le vecteur de poids initialisé et K = le nombre de voisins prise en compte pour le calcul du barycentre.

Pour lancer notre programme, nous allons utiliser la fonction main suivante :

- **main**() :

La fonction main nous permet tout simplement de lancer l'exécution de l'algorithme relief afin de nous permettre d'effectuer nos constats ainsi que de tirer une conclusion sur les différents résultats qui seront observés.

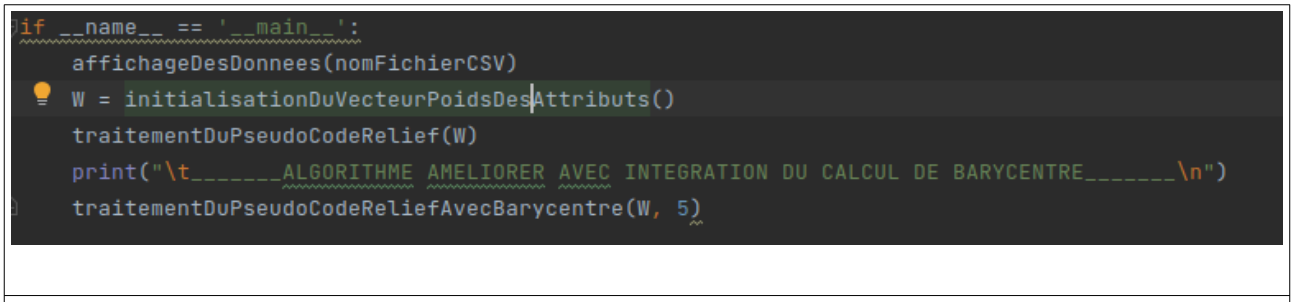


Figure : Code du main

■ ReadMe

Pour pouvoir prendre en main le dossier permettant de traiter notre projet, il est conseillé d'installer l'IDE **pycharm-community** (https://idroot.us/install-pycharm-debian-10/). Puis de lancer l'exécution du programme principale à l'aide du fichier **main.py.**

Ouvrir **pycharm-community** sur votre machine et récupéré le dossier contenant nos différents fichiers sources. Dans le cas où l'IDE pycharm détectera des erreurs dans le code. Il vous suffira de créer un nouveau projet en saisissant le nom du projet ainsi que l'interpréteur **python3.7**. La fenêtre ci-dessous s'ouvrira puis appuyer sur le bouton **Create** (***voir Figure :*** Creation de projet dans pycharm-community***).***

N.B. Les erreurs qui pourraient survenir, seront dues au fait que notre projet contient certaines variables d'environnements propres à ma machine et qui ne sont pas les mêmes sur la vôtre. Donc il vous faudra créer un nouveau dossier et importer nos fichiers sources tout en respectant l'emplacement de chaque fichier.

Si vous remarquez des erreurs dans le code, pensez à importer les fichiers sources de notre projet dans votre nouveau dossier tout en respectant l'emplacement de chaque fichier (***voir Figure :*** Hierarchie des fichiers sources***)***.

Importer les librairies Matplotlib, Pandas, Numpy et Sklearn. Pour ce faire, il vous suffit de passer votre souris sous les mots contenant les erreurs et cliquer sur le lien « install package nomPackage » (***voir Figure :*** **Installation des packages***)*.

Et enfin, pour finir, vous pourriez choisir le jeu de donnée que vous souhaitez observer (il suffit de vous rendre dans le code du fichier main.py et d'enlever en commentaire nomFichierCSV qui correspond au jeu de donnée que vous souhaitez étudier et de remettre en commentaire tous les autres variables nomFichierCSV). Puis lancer le programme et observer les résultats de notre implémentation.

Une fois votre programme lancé, une fenêtre s'ouvrira avec la représentation graphique du jeu de données choisi en fonction des attributs les plus précis du jeu de données choisis. Il vous suffira de la fermer si vous voulez voir le reste de l'exécution de notre programme.

Si vous souhaitez observer plus en détail les résultats obtenus durant l'exécution de notre programme. Rendez vous dans le fichier **main.py** et enlever en commentaire l'ensemble de fonction *print(« message explicatif»)* qui y sont présents.

N.B. Le code que nous vous avons fournit est portable et peut être utiliser avec toutes les jeux de données permettant de traiter des classifications binaires ou plus. Si vous souhaitez utiliser vos propres jeux de données, il vous suffira de créer un fichier CSV contenant les données qu'il faut et de renseigner leur emplacement dans la variable nomFichierCSV (voir la syntaxe de nos fichiers.csv)

III. Conclusion

■ Analyse

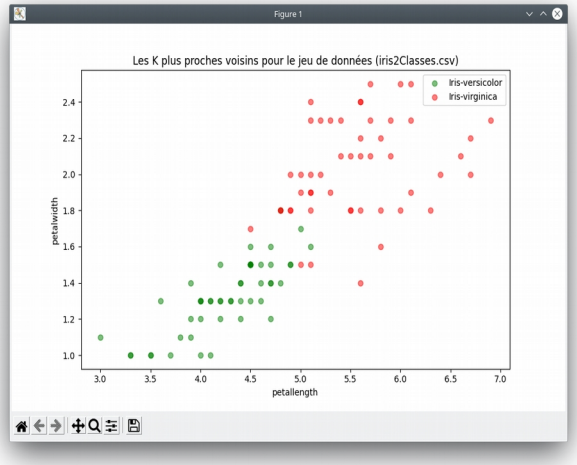


Figure : Graphe de Iris2Classes avec les deux attributs les plus pertinents

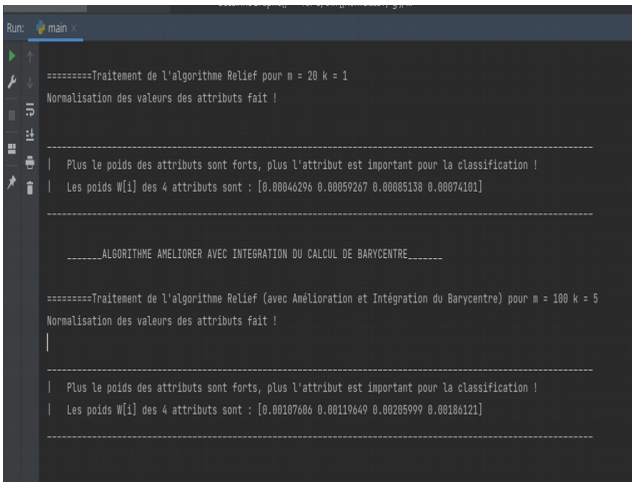
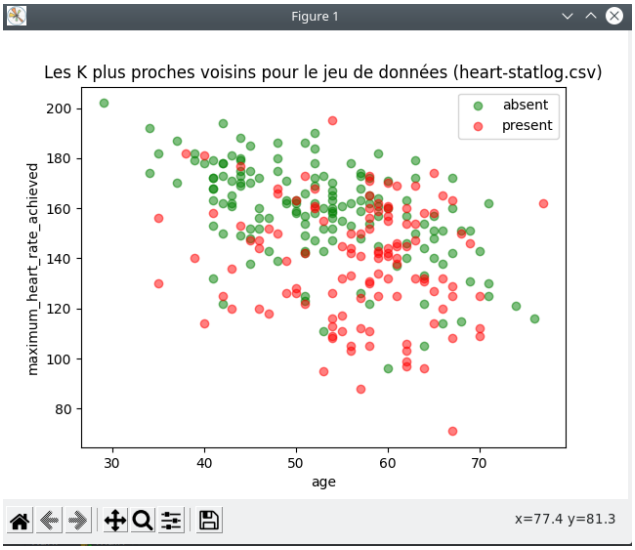


Figure : Analyse tirée de Iris2Classes

Pour après plusieurs exécutions de notre programme sur le jeu de données Iris2Classes, nous pouvons conclure que l'attribut le plus pertinent pour déterminer la classe de l'iris est le 3^{e} attribut « **petallength »** (observable dans la sortie console de mon code, juste après l'affichage du graphe). Et comme on pouvait s'y attendre, le traitement des données à l'aide de la méthode du barycentre est bien plus précis et plus efficace pour k = 5.



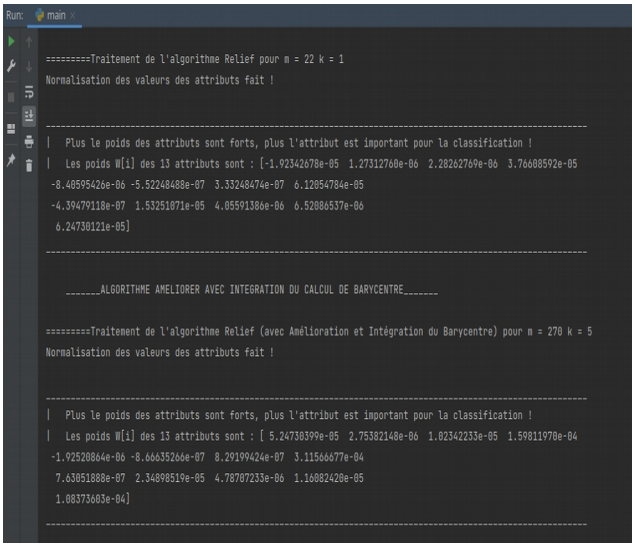


Figure : Graphe de Heart-statlog avec les deux attributs les plus pertinents

Figure : Analyse tirée de Heart-statlog

Lors de l'exécution de notre programme, nous avons constaté que le 8e attribut « **maximum_heart_rate_achieved** » est le plus pertinent dans la classification des données pour le jeu de données *Heart-statlog. Il est suivi dans la course par plusieurs attributs qui sont très pertinents tels que : l'attribut n°1 « **age** », n°4 « **resting_blood_pressure** » et n°5 « **serum_cholestoral** ». Si on se refaire au calcul des données sans utiliser le calcul du barycentre, on se rend compte qu'après l'attribut n8, c'est l'attribut n°1 qui est le plus pertinent. L'application du barycentre nous permet de pousser la recherche un peu loin et de voir plus de possibilités et ainsi observer en détail la performance de certains attributs.*

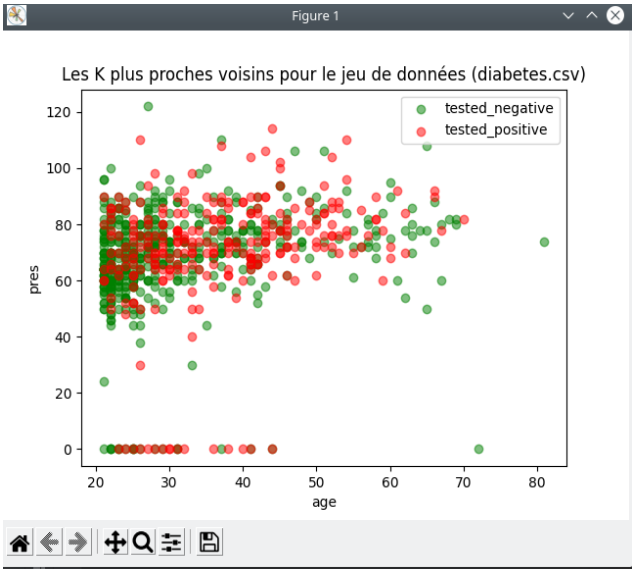


Figure : Graphe de Diabète avec les deux attributs les plus pertinents

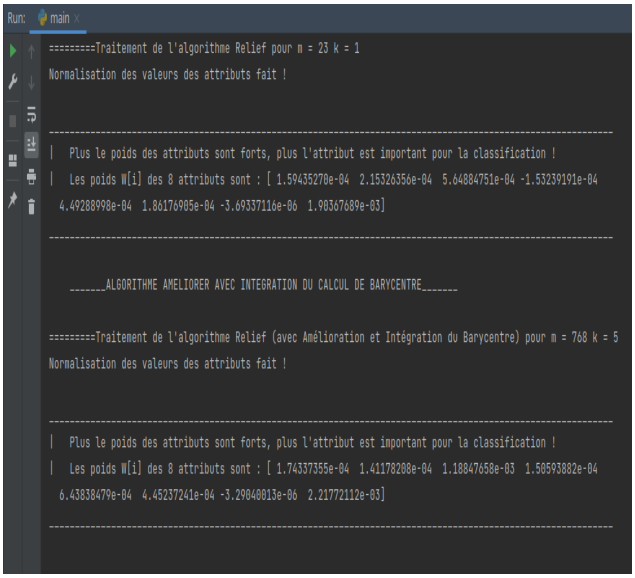


Figure : Analyse tirée de Diabète

Lors de l'exécution de notre programme, nous avons constaté que le 8e attribut « **age** » est le plus pertinent dans la classification des données pour le jeu de données Diabète. *Il est suivi dans la course par plusieurs attributs qui sont très pertinents tels que : l'attribut n°2 « **plas** », n°3 « **pres** », n°4 « **skin** », n°1 « **preg** » et n°5 « **insu** ».*

A travers la sortie du graphe, on peut conclure que la classification des données pour le jeu d'exemples Diabètes est très complexes et difficiles à réaliser malgré tous les données dont on dispose.

■ Annexe

Pseudo-Code de Relief

Données S : ensemble de n instances, décrites dans un espace de d attributs ; les instances sont de classe + ou de classe -
Paramètres m nombre d'itérations dans le calcul des poids
Résultat W[1..d] : vecteur de poids des attributs

Pour i de 1 à d
    Initialiser W[i] à 0
Pour j de 1 à m
    Choisir au hasard une instance X dans S
    Déterminer *nearestHit* le plus proche voisin de X de la même classe que X
    Déterminer *nearestMiss* le plus proche voisin de X de la classe opposée à celle de X
    Pour i de 1 à d
        $W[i] = W[i] - 1/m\ (X_i - nearestHit_i)^2 + 1/m\ (X_i - nearestMiss_i)^2$

Figure : Pseudo-Code de Relief

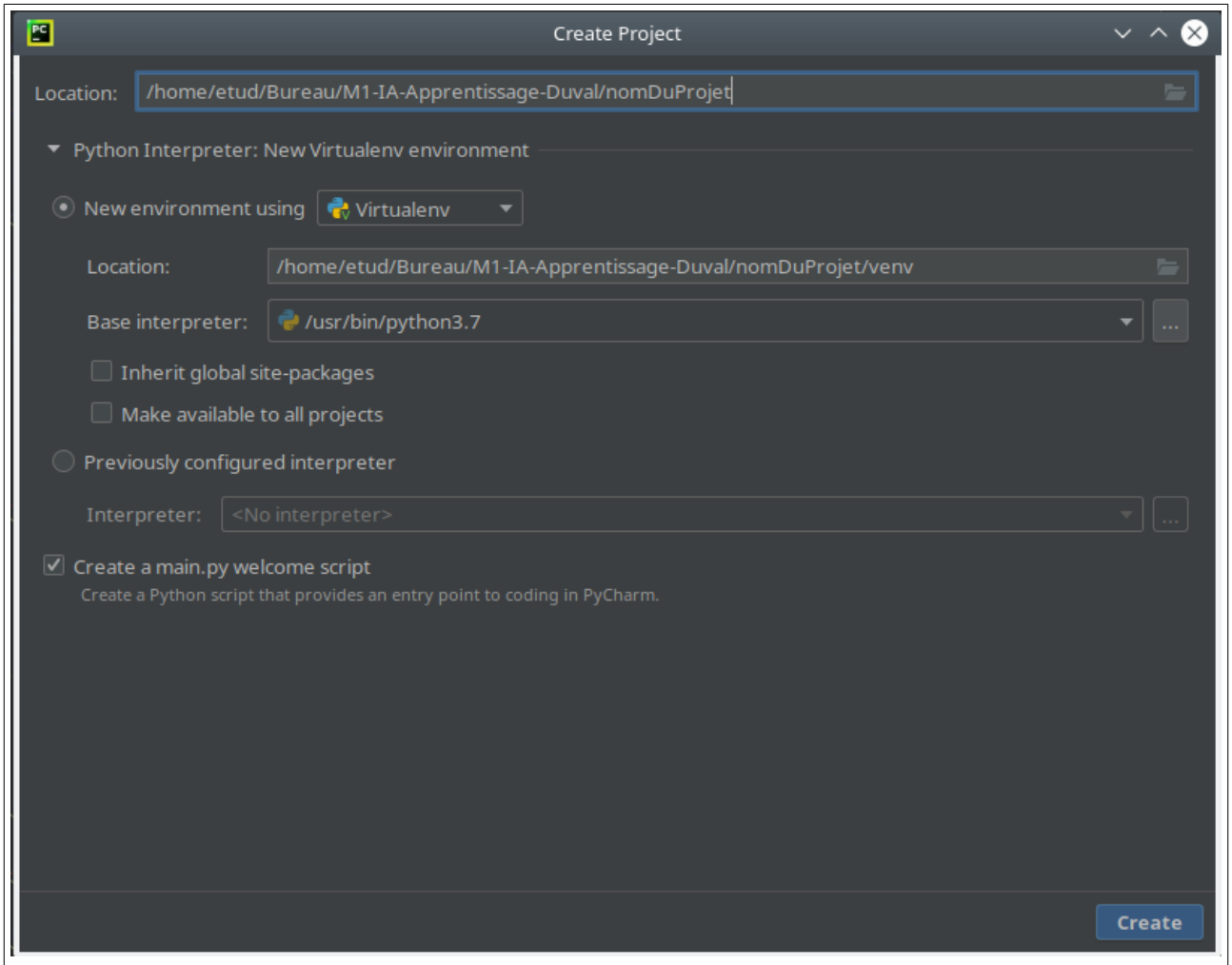


Figure : Création de projet dans pycharm-community

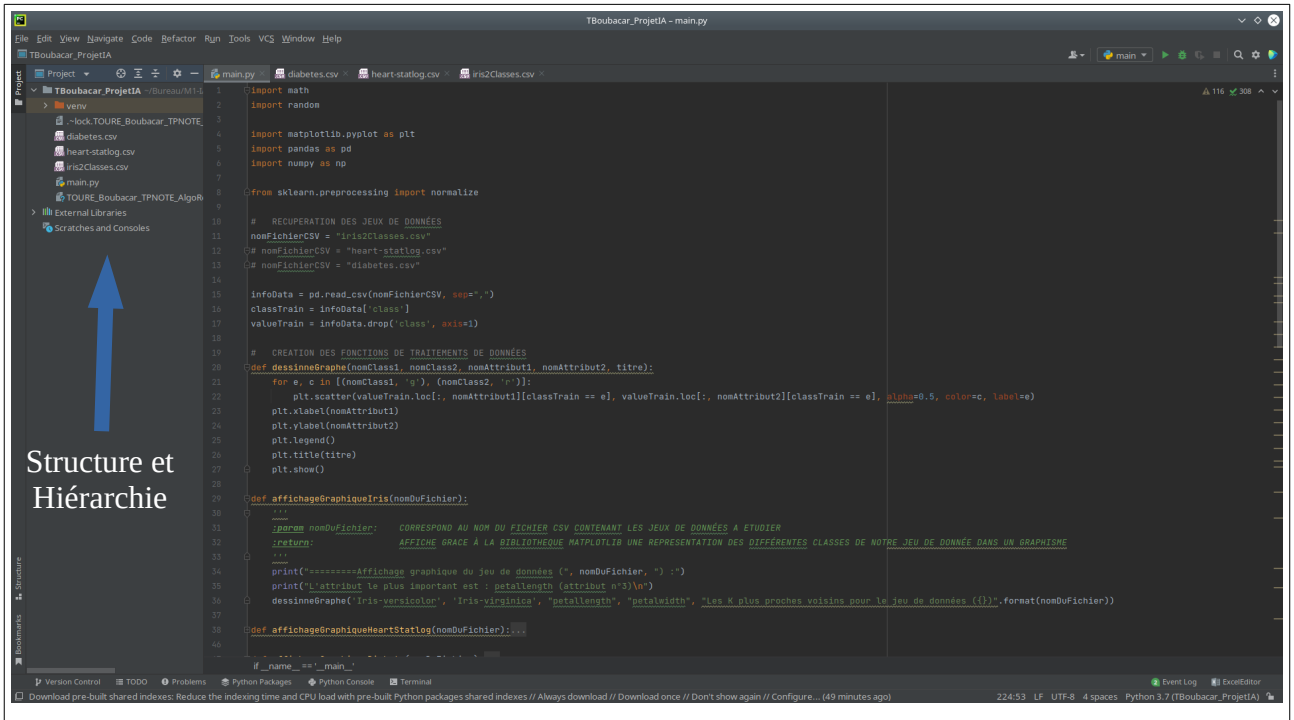


Figure : Hiérarchie des fichiers sources

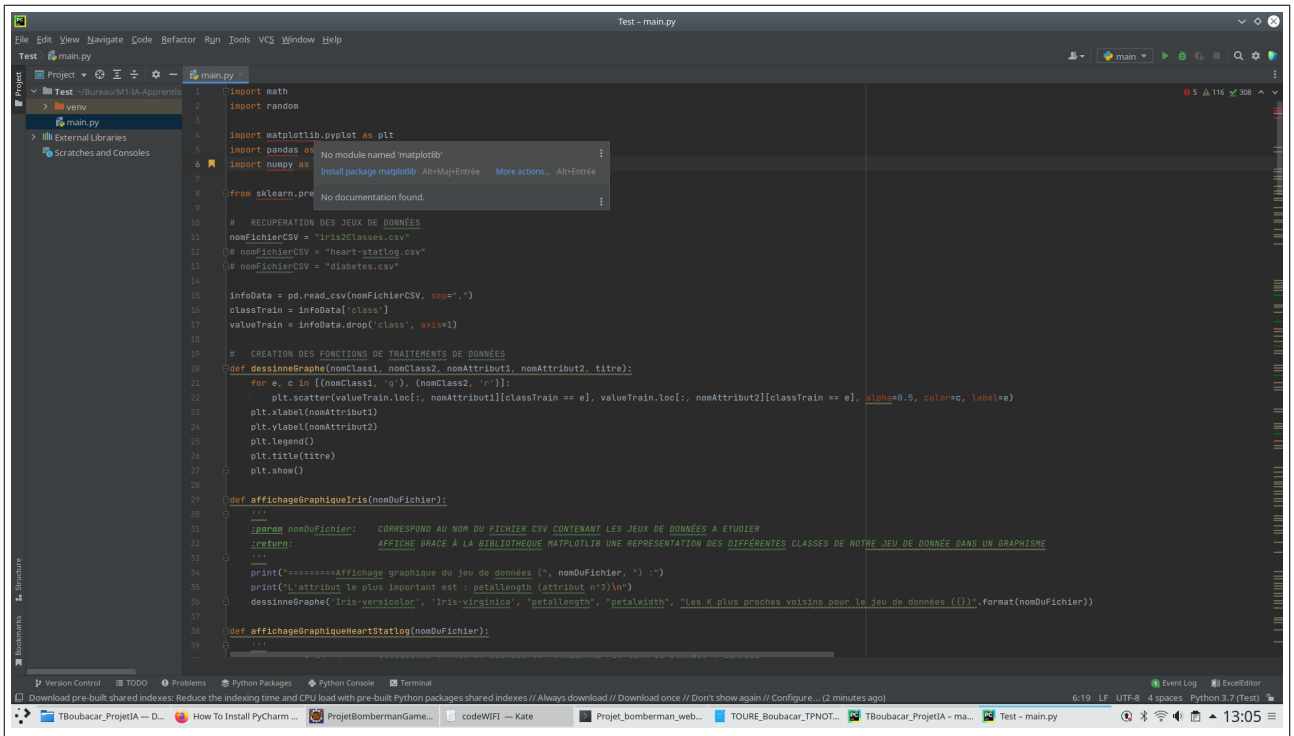


Figure : Installation des packages

main.py | diabetes.csv | heart-statlog.csv | iris2Classes.csv

```
preg,plas,pres,skin,insu,mass,pedi,age,class
6,148,72,35,0,33.6,0.627,50,tested_positive
1,85,66,29,0,26.6,0.351,31,tested_negative
8,183,64,0,0,23.3,0.672,32,tested_positive
1,89,66,23,94,28.1,0.167,21,tested_negative
0,137,40,35,168,43.1,2.288,33,tested_positive
5,116,74,0,0,25.6,0.201,30,tested_negative
3,78,50,32,88,31,0.248,26,tested_positive
10,115,0,0,0,35.3,0.134,29,tested_negative
2,197,70,45,543,30.5,0.158,53,tested_positive
8,125,96,0,0,0,0.232,54,tested_positive
4,110,92,0,0,37.6,0.191,30,tested_negative
10,168,74,0,0,38,0.537,34,tested_positive
10,139,80,0,0,27.1,1.441,57,tested_negative
1,189,60,23,846,30.1,0.398,59,tested_positive
5,166,72,19,175,25.8,0.587,51,tested_positive
7,100,0,0,0,30,0.484,32,tested_positive
0,118,84,47,230,45.8,0.551,31,tested_positive
7,107,74,0,0,29.6,0.254,31,tested_positive
1,103,30,38,83,43.3,0.183,33,tested_negative
1,115,70,30,96,34.6,0.529,32,tested_positive
3,126,88,41,235,39.3,0.704,27,tested_negative
```

Figure : Syntaxe de nos fichiers CSV