Projet Noté

Algorithmes Avancés

José Ferro Pinto Fabrice Ceresa

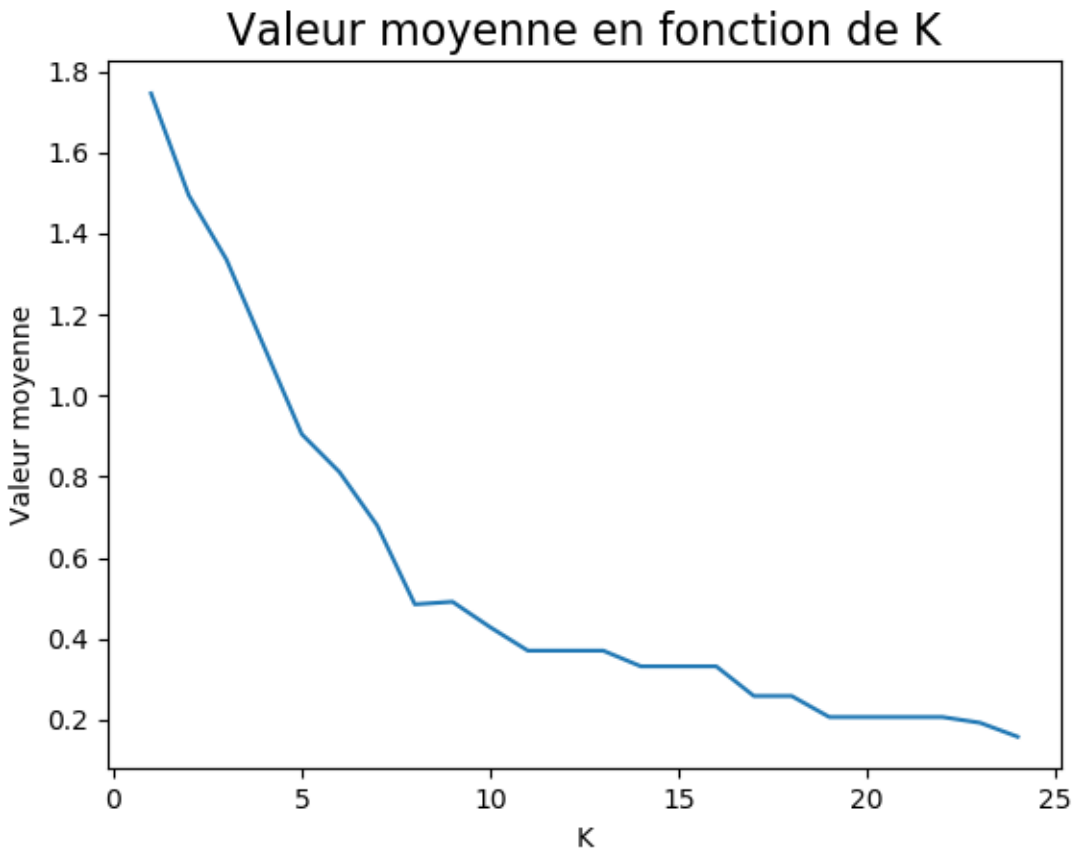
10 juin 2018

1 Partie 1

1.1 Valeur optimale pour K

Nous avons utilisé la méthode des K-Means avec un nombre de clusters K qui varie entre 1 et 25 et comme paramètres : `init='random'` pour que l'algorithme choisisse comme barycentres initiaux des données aléatoires et `random_state=0` comme seed pour le générateur de nombres aléatoires.

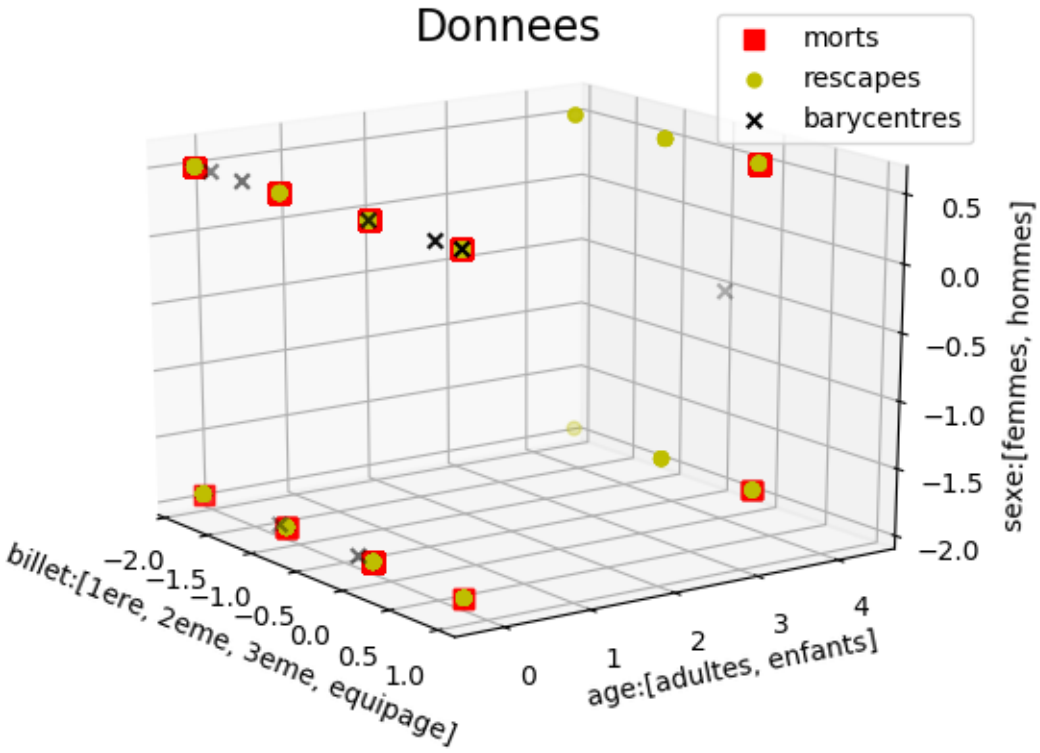
Pour chaque valeur de K, nous avons déterminé la moyenne des distances moyennes entre les barycentres et leurs points. Nous obtenons comme résultats :



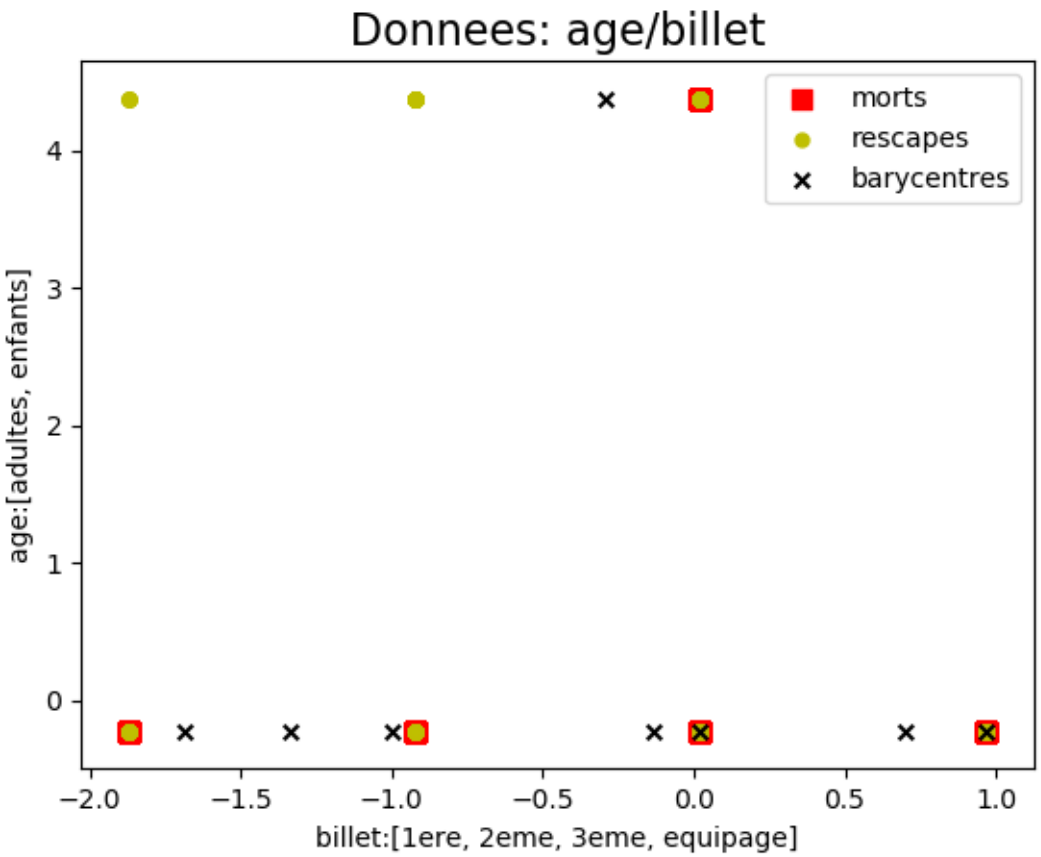
Nous pouvons voir que le nombre optimal de clusters à choisir est 8.

En effet, plus K augmente plus la valeur moyenne diminue mais on peut voir qu'à partir de K = 8 la distance entre les barycentres et leurs points diminue beaucoup moins vite. Choisir un nombre de clusters plus élevé n'apporterait donc pas beaucoup plus d'information sur les données.

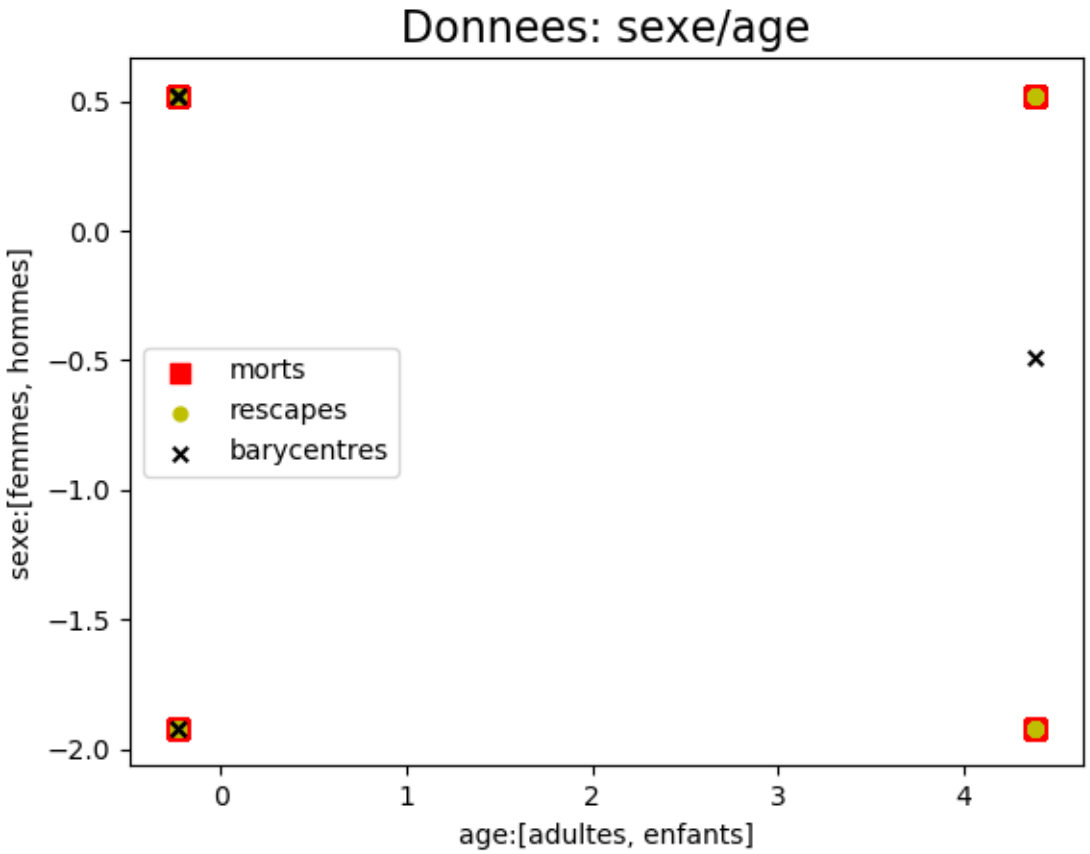
Représentation des données :



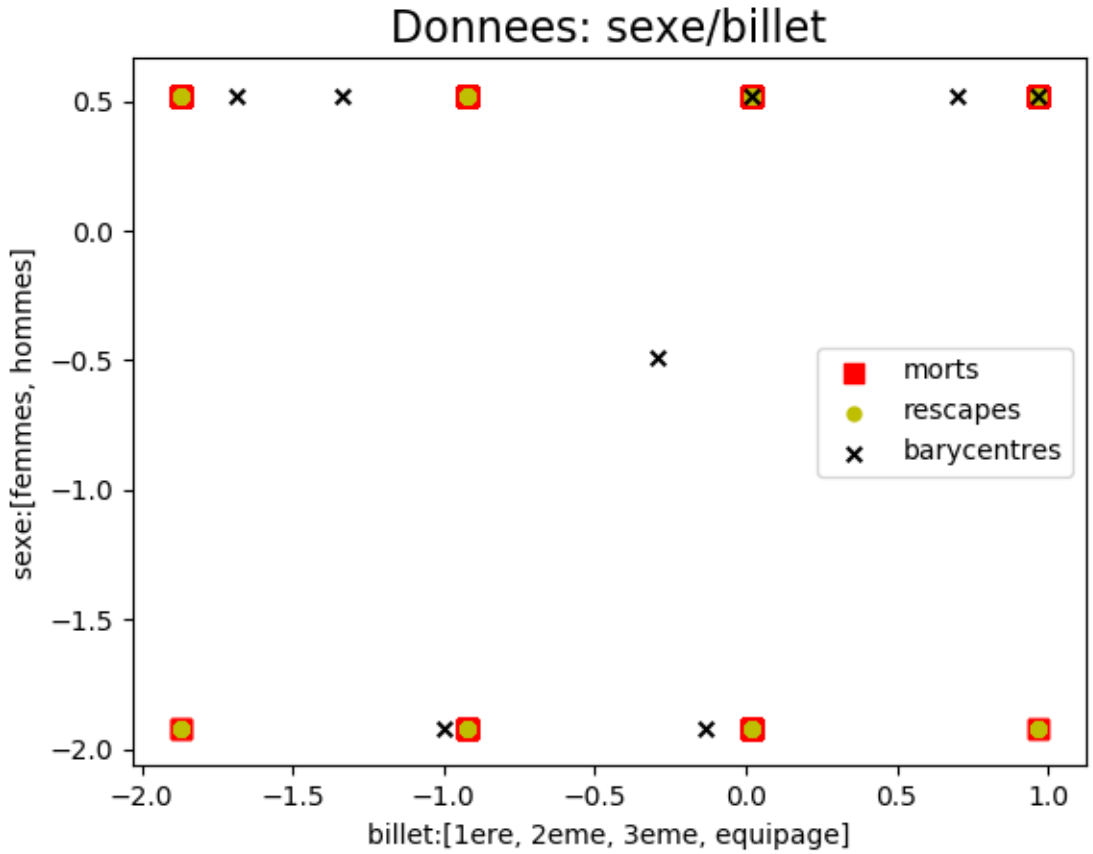
A partir de la représentation en 3 dimensions, nous pouvons déjà constater qu'aucun enfant en 1ère et 2ème classes n'est mort et que le sexe n'a que peu d'influence sur le taux de survie des enfants.



Nous pouvons voir par la position des barycentres que la grande majorité des passagers étaient des adultes et que parmi eux la plupart étaient en 3ème classe ou faisaient partie de l'équipage. C'est dans ces groupes que se trouvaient la majeure partie des personnes qui n'ont pas survécu.



Ici nous constatons (mieux avec la représentation 3d) qu'il y avait bien plus d'hommes que de femmes sur le Titanic et que leur taux de survie était plus faible.



Finalement ici, nous pouvons voir que les femmes étaient presque pas présentes parmi les membres de l'équipage. C'est surtout le fait d'être une femme qui offrait une meilleure chance de survie mais aussi le fait d'être en 1ère ou 2ème classe.

2 Partie 2

Pour normaliser les données, nous avons utilisé `preprocessing.normalize(raw['data'])` comme demandé dans le TP.

Le programme se passe en 4 boucles intriquées :

1. Pour chacun des datasets
2. Pour chacun des classificateurs
3. Pour chacun des paramètres
4. Pour chacune des étapes de la validation croisée

Les datasets utilisés :

— `load_breast_cancer` pour les données du cancer du sein

— `load_wine` pour les données sur du vin

Les différents classificateurs utilisés et leurs paramètres :

— `KNeighborsClassifier` pour la méthode des K-plus proches voisins.
Paramètre variable : `n_neighbors` nombre de voisins pris en compte : [1,51] avec un pas de 5

— `DecisionTreeClassifier` pour les arbres de décisions.
Paramètre variable : `min_samples_leaf` Nombre d'objets à partir duquel on comptabilise une feuille [2, 52] avec un pas de 5

— `MLPClassifier` pour le Perceptron multi-couche avec une couche utilisant "stochastic gradient descent"
Paramètres fixes : `solver='sgd', activation='logistic', max_iter=1000, verbose=False, learning_rate_init=0.`
Paramètre variable : `hidden_layer_sizes=(nodes,)` donc une couche et nodes varie entre 2 et 20 par pas de 3.

— `MLPClassifier` pour le Perceptron multi-couche avec une couche utilisant "a stochastic gradient-based optimizer"
Paramètres fixes : `solver='adam', activation='logistic', max_iter=1000, verbose=False, learning_rate_init=0`
Paramètre variable : `hidden_layer_sizes=(nodes,)` donc une couche et nodes varie entre 2 et 20 par pas de 3.

— `MLPClassifier` pour le Perceptron multi-couche avec deux couches utilisant “stochastic gradient descent”
Paramètres fixes : `solver='sgd', activation='logistic', max_iter=1000, verbose=False, learning_rate_init=0.`
Paramètre variable : `hidden_layer_sizes=(nodes,5)` donc deux couches et nodes varie entre 2 et 20 par pas de 3.

— `MLPClassifier` pour le Perceptron multi-couche avec deux couches utilisant “a stochastic gradient-based optimizer”
Paramètres fixes : `solver='adam', activation='logistic', max_iter=1000, verbose=False, learning_rate_init=0`
Paramètre variable : `hidden_layer_sizes=(nodes, 5)` donc deux couches et nodes varie entre 2 et 20 par pas de 3.

La validation est donc la validation croisée : `RepeatedKFold`
Nous avons décidé de prendre les mesures suivantes :

— le temps d'exécution

— la moyenne des scores

— l'écart type des scores

La sortie de l'exécution du script python donne la sortie suivante :

```
Breast Cancer data of size 569 :
KNeighborsClassifier
        mean is 0.9156607528475251 with 0.005261918472447842 as standard deviation.
        Worked for 58.37147793300028 seconds
DecisionTreeClassifier
        mean is 0.918515193857532 with 0.007526398539306361 as standard deviation.
        Worked for 1.9232283050005208 seconds
MLPClassifier one layer sgd_solver
        mean is 0.727365574703721 with 0.012868801670619141 as standard deviation.
        Worked for 29.68843971000024 seconds
MLPClassifier one layer adam_solver
        mean is 0.8277927857993066 with 0.01944967972932608 as standard deviation.
        Worked for 6.480671232000532 seconds
MLPClassifier two layers, second layer 5 nodes, sgd_solver
        mean is 0.7151894115820524 with 0.017601386032980122 as standard deviation.
        Worked for 204.33885359600026 seconds
MLPClassifier two layers, second layer 5 nodes, adam_solver
        mean is 0.8806233504114266 with 0.007611351937622535 as standard deviation.
        Worked for 19.824140925999927 seconds
Wine data of size 178 :
KNeighborsClassifier
        mean is 0.7270447330447332 with 0.05395925687595904 as standard deviation.
        Worked for 58.24855370300065 seconds
DecisionTreeClassifier
        mean is 0.8258658008658007 with 0.06566382345753549 as standard deviation.
        Worked for 0.2911996069997258 seconds
MLPClassifier one layer sgd_solver
        mean is 0.511431216931217 with 0.013653796940041013 as standard deviation.
        Worked for 52.15133309700013 seconds
MLPClassifier one layer adam_solver
        mean is 0.46363756613756607 with 0.04200778922986875 as standard deviation.
        Worked for 4.492833305000204 seconds
MLPClassifier two layers, second layer 5 nodes, sgd_solver
        mean is 0.37066137566137564 with 0.013903555873703817 as standard deviation.
        Worked for 113.28683143299986 seconds
MLPClassifier two layers, second layer 5 nodes, adam_solver
        mean is 0.49454232804232806 with 0.011857834905633243 as standard deviation.
        Worked for 7.685992364000413 seconds
```

On peut voir qu'avec un échantillon plus grand (569 échantillons) on trouve des résultats plus probants qu'avec moins d'échantillons (178) avec un score moyen général de 0.830857845 avec un écart-type moyen général de 0.011719923 pour le cancer du sein contre 0.565530503 avec un écart-type moyen général de 0.033507676 pour le vin.

Pour ce qui est du temps de travail en fonction du nombre d'échantillons, c'est globalement plus rapide avec moins d'échantillons.

Par contre, c'est pas parce que le temps de travail est plus grand que le score final sera meilleur.

3 Partie 3

Pour la partie 3, nous avons décidé de partie sur les données suivantes : “Letter Image Recognition Data”, donc la reconnaissance de lettres par caractéristiques d'image.

Cette base de donnée contient 20000 éléments.

Il y a 17 attributs différents :

```
1.  lettr   capital letter                (26 values from A to Z)
2.  x-box   horizontal position of box    (integer)
3.  y-box   vertical position of box      (integer)
4.  width   width of box                  (integer)
5.  high    height of box                 (integer)
6.  onpix   total # on pixels             (integer)
7.  x-bar   mean x of on pixels in box    (integer)
8.  y-bar   mean y of on pixels in box    (integer)
9.  x2bar   mean x variance               (integer)
10. y2bar   mean y variance               (integer)
11. xybar   mean x y correlation          (integer)
12. x2ybr   mean of x * x * y             (integer)
13. xy2br   mean of x * y * y             (integer)
14. x-ege   mean edge count left to right (integer)
15. xegvy   correlation of x-ege with y   (integer)
16. y-ege   mean edge count bottom to top (integer)
17. yegvx   correlation of y-ege with x   (integer)
```

On a commencé par transformer les lettres en nombres (i.e. A->0, B->1, ..., Z->25) pour avoir que des valeurs numériques pour simplifier l'import des données avec `numpy.genfromtxt()`

Nous avons repris le code de la partie 2 pour la comparaison des modèles d'apprentissage.

Les résultats sont les suivants :

```
KNeighborsClassifier
        mean is 0.9203122727272728 with 0.020939017447170083 as standard deviation.
        Worked for 360.20789452599956 seconds
DecisionTreeClassifier
        mean is 0.733379090909091 with 0.04240275735128243 as standard deviation.
        Worked for 174.38097832700032 seconds
MLPClassifier one layer sgd_solver
        mean is 0.5157916666666665 with 0.17533825530163752 as standard deviation.
        Worked for 700.9990525189996 seconds
MLPClassifier one layer adam_solver
        mean is 0.6683058333333333 with 0.16247980283245123 as standard deviation.
        Worked for 254.53706921899993 seconds
MLPClassifier two layers, second layer 5 nodes, sgd_solver
        mean is 0.039465 with 0.0011532309684823187 as standard deviation.
        Worked for 102.64895842199985 seconds
MLPClassifier two layers, second layer 5 nodes, adam_solver
        mean is 0.319565 with 0.07064751617478612 as standard deviation.
        Worked for 232.7630711769998 seconds
```

Les résultats avec ces données sont beaucoup plus variables que dans la partie 2.

Le perceptron travaille longtemps pour un résultat pas très probant ($\sim$0.66), ensuite l'arbre de décision ($\sim$0.73) et le meilleur résultat revient à la méthode des K-plus proches voisins ($\sim$0.92).