Étude Pratique (1) – IAA – 2021

1-Etude des k-ppv sur des données en damier

1.2 Données en damier

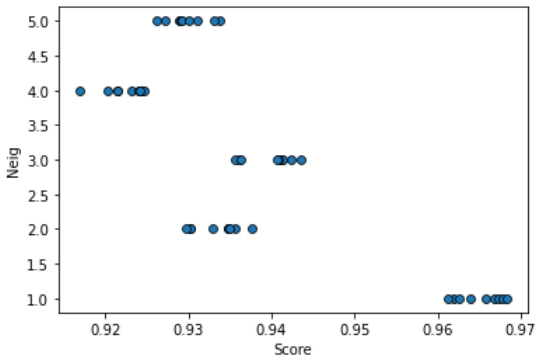
1)

Dim	Size	Neig	Nb_ex	Noise	Score
2	2	1	1000	0	0.995
3	2	1	1000	0	0.952
4	2	1	1000	0	0.918
5	2	1	1000	0	0.88
6	2	1	1000	0	0.832
7	2	1	1000	0	0.811
8	2	1	1000	0	0.788
9	2	1	1000	0	0.77
10	2	1	1000	0	0.767
2	3	1	1000	0	0.98
2	4	1	1000	0	0.977
2	5	1	1000	0	0.967
2	6	1	1000	0	0.93
2	7	1	1000	0	0.95
2	8	1	1000	0	0.93
2	2	1	2000	0	0.993
2	2	1	3000	0	0.992
2	2	1	4000	0	0.995
2	2	1	5000	0	0.996
2	2	1	6000	0	0.006
2	2	1	7000	0	0.996
2	2	1	8000	0	0.997
2	2	1	9000	0	0.996
2	2	1	10000	0	0.997
2	2	1	1000	0.05	0.953
2	2	1	1000	0.1	0.926
2	2	1	1000	0.15	0.894
2	2	1	1000	0.2	0.855
2	2	2	1000	0	0.978
2	2	3	1000	0	0.971
2	2	4	1000	0	0.982
2	2	5	1000	0	0.987

3)
Voici le programme que j'ai utilisé afin de répondre à la question :

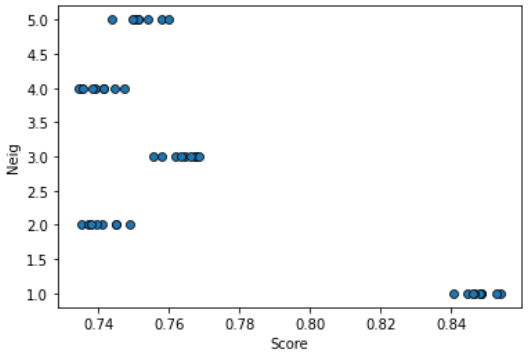
```
dict_tab_2 = {
            "dim":2,
            "size":[2,3,4,5,6,7,8,9,10],
            #"size":[2],
            "nb_ex":1000,
            "noise":0,
            "n_neig":[1,2,3,4,5],
            "score":0,
        }


tab_dict_res = []

for i in dict_tab_2["size"]:

    for j in dict_tab_2["n_neig"] :

        moy = 0

        for i in range(10) :

            nb_data_test = dict_tab_2["nb_ex"] * 0.3
            nb_data_train = dict_tab_2["nb_ex"] * 0.7

            data, labels = damier(dict_tab_2["dim"], i, dict_tab_2["nb_ex"], dict_tab_2["noise"])

            x_train = data[:int(nb_data_train)]
            y_train = labels[:int(nb_data_train)]

            x_test = data[int(nb_data_test):]
            y_test = labels[int(nb_data_test):]

            knn=KNeighborsClassifier(n_neighbors= j )
            knn.fit(x_train, y_train)
            pred_test=knn.predict(x_test)

            #Evaluer le modèle en utilisant le score :
            score = knn.score(x_test, y_test)

            moy += score

        tab_dict_res.append({"dim" :2,
                        "size" : i,
                        "nb_ex" : 1000,
                        "noise" : 0,
                        "n_neig" : j,
                        "score" : moy/10,
                        })
```

Modèle sans bruit :



Nous pouvons observer que le score le plus élevé est atteint pour n_neighbors égal à 1. Les autres données ne sont pas linéairement séparables les unes des autres.

Modèle avec bruit :



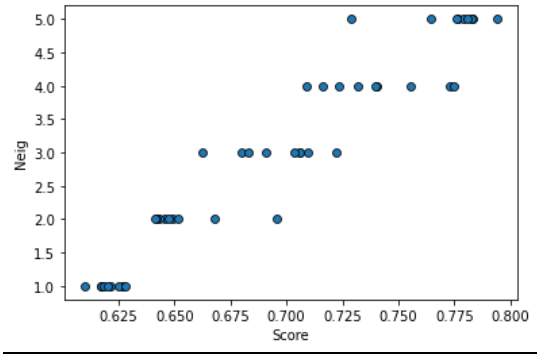
Avec la présence de bruit on peut observer une meilleure séparation des données, mais toujours seulement avec n_neighbors égal à 1, par rapport au modèle sans bruit. Les scores sont significativement plus bas ce qui s'explique par la présence du bruit.

2-Quid sur arbres de décisions

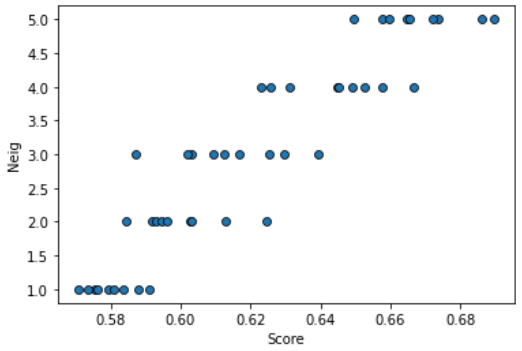
Pour répondre à cette partie j'ai changé le classifier par celui-ci :

```
clf = tree.DecisionTreeClassifier(max_depth=j)
clf = clf.fit(x_train, y_train)

pred_test=clf.predict(x_test)

#Evaluer le modèle en utilisant le score :
score = clf.score(x_test, y_test)
```

Modèle sans bruit :



Modèle avec bruit :



Le modèle permettant la meilleure classification pour résoudre le problème est le Kneighbors, et contrairement à ce qu'on aurait pu attendre, avec une valeur égale à 1.