Question A :

Parce qu'on a deux algorithmes d'optimisation de la procedure de prediction à base de k plus proches voisins : K-D Tree et BallTree. On a testé tout les mesure de distance pour ces deux algorithmes :

Les mesures de distance sont comme ci-dessous :

identifier	class name	args	distance function
"euclidean"	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
"manhattan"	ManhattanDistance	•	`sum(|x - y|)`
"chebyshev"	ChebyshevDistance	•	`max(|x - y|)`
"minkowski"	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`
"wminkowski"	WMinkowskiDistance	p, w	`sum(w * |x - y|^p)^(1/p)`
"seuclidean"	SEuclideanDistance	V	`sqrt(sum((x - y)^2 / V))`
"mahalanobis"	MahalanobisDistance	V or VI	`sqrt((x - y)' V^-1 (x - y))`

Pour K-D Tree, les résultats de test comme la suivante :

Mesure de distance	Taux d'erreur	Temps de calcule(s)
Euclidean	0.0554	78.44
Manhattan	0.068	68.02
Chebyshev	0.304	44.98
Minkowski	0.0554	80.21

Pour BallTree, les résultats de test comme la suivante :

Mesure de distance	Taux d'erreur	Temps de calcule(s)
Euclidean	0.0554	61.18
Manhattan	0.068	68.92
Chebyshev	0.309	51.94
Minkowski	0.0554	79.21

Selon le tabeau, on peut savoir que pour notre données , si on utilise le mesure de distance 'Euclidean', on peut prendre le meilleur résultat. Mais si on veut que le résultat est moins précise et tourne plus vite, on peut choisir le mesure 'Chebyshev'

Question B :

En principal, il y a deux façons de pondérer les points du voisinage. On peut changer le variable de "weight" pour choisir la façon de pondérer les points. Si on considère que les points du voisinage vaut pareille. On fait weight = "uniform". Par contre, on si on considère que les points plus proche vaut plus, on fait weight = "distance".

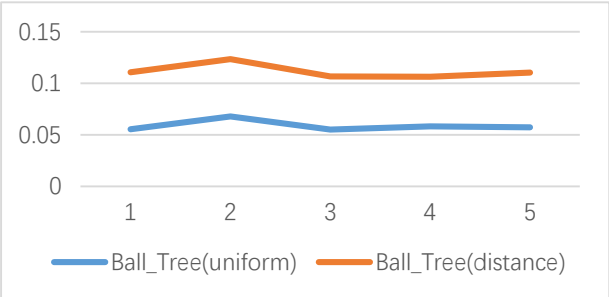
Comme on sait qu'on peut prendre le meilleur résultat en utilisant metric='euclidean'. Donc

ici on utilise ‘metric=euclidean’.
On a les résultat de test en utilisant K_D_Tree comme la suivante :
K-D-Tree :

N_neighbors	Weights	Temps de calcul	Taux d’erreur
1	Uniform	79.10	0.0554
1	Distance	78.16	0.0554
2	Uniform	79.01	0.068
2	Distance	78.05	0.0554
3	Uniform	78.13	0.0552
3	Distance	78.4	0.0516
4	Uniform	78.9	0.0584
4	Distance	78.2	0.048
5	Uniform	80.9	0.0574
5	Distance	81.68	0.0532

Le résultat de BallTree

N_neighbors	Weights	Temps de calcul	Taux d’erreur
1	Uniform	59.10	0.0554
1	Distance	60.06	0.0554
2	Uniform	58.21	0.068
2	Distance	57.15	0.0554
3	Uniform	59.13	0.0552
3	Distance	60.04	0.0516
4	Uniform	58.92	0.0584
4	Distance	57.02	0.048
5	Uniform	61.09	0.0574
5	Distance	59.28	0.0532



Selon le tableau, on peut savoir que :

(1) En utilisant l'algo BallTree et K-D Tree, le taux d'erreur est pareille. Mais l'algo BallTree, il utilise moins de temps.

(2) Si on utilise plus d'un voisin, il vaut mieux qu'on utilise 'weight = 'distance'',car le résultat est plus precise.

(3) Le temps varie pas beaucoup en utilisant le nombre des voisins différents.

(4) Le meilleur resultat est qu'on utilise 4 voisins avec 'Weights='distance'' et 'algo='BallTree'', car il peut prendre le meilleur résultat avec moins de temps.

Quesiton C :

Pour faire la pré-traitement des données. On utilise la fonction dans le site scikit. Learn PCA (n_components=compo). On peut préciser la dimension qu'on veut utiliser par la paramètre 'n_components=compo'.

Comme on sait qu'on a le meilleur résultat en utilisant 4 voisins avec 'Weights='distance'' et 'algo='BallTree''. Donc pour tester la prediction avec le pré-traitement des donnees, on utilise la condition : 4 voisins avec 'Weights='distance'' et 'algo='BallTree''

On fait le test comme la suivante:

Dimension	Temps de calcule	Taux d'erreur
20	1.3	0.0486
25	1.6	0.0488
30	2	0.0448
35	2.4	0.0484
40	2.9	0.0456
45	3.6	0.044
50	4.7	0.0434
55	5.5	0.0416
60	6.3	0.043
70	7.67	0.043

Donc avec le pré-traitement des données, on peut savoir si on utilise la dimension 55 de PCA, on peut avoir le meilleur résultat. Le temps est aussi très petit.

Pour faire le matrice de fusion, on va utiliser les condition comme ci-dessous :

PCA(n_componant=55)

Nb_neighbour = 4

Algorithme=Ball_tree

Weight = 'distance'

Metric = 'euclidean'