GOKTEKIN Muhammed Emin
KARATAS Musab

Rapport projet IA

Partie 1 :

1)Les données comportent 14 attributs.

2)Les données sont classées en 4 instances : Class 0,1,2,3

3)
Class 0 : 550
Class 1 : 720
Class 2 : 366
Class 3 : 202

4)
Selon la figure du schéma, les données ne sont pas linéairement séparables.
J'ai aussi utilisé sns.airplot pour vérifier avec les autres attributs et je n'en ai pas trouvé qui permet de séparer linéairement.

5)
One-hot :
Pour l'arbre on a pas besoin d'encodage one-hot car les arbres peuvent gérer les variables catégorielles.
Pour le réseau de neurones pas besoin car dans la BDD les valeurs de classes sont des int, entre 0 et 3, ce ne sont pas des variables catégorielles avec un string (comme « iris-setosa » par exemple).

Normalisation :
Les arbres ne sont pas sensibles à l'échelle des données donc pas besoin de normalisation pour ce genre de modèles.
Cependant c'est intéressant pour le réseau de neurones pour que les caractéristiques se situent dans une même plage spécifique et qu'on ai pas une caractéristique qui domine les autres à cause de leur petite échelle.
Quand on regarde les données de synthetic on voit qu'il y a des attributs avec de petites valeurs aux alentours de 10 comme les attributs A et B, et il y a des attributs avec de grandes valeurs aux alentours de 1000 comme les attributs C et F. Donc c'est important de faire une normalisation ici pour les réseaux de neurones.

6)
Cela permet une évaluation impartiale des données, car on évalue les performances du modèle sur des données qu'il n'a jamais vu durant son entrainement. Cela permet voir sa capacité à généraliser sur de nouvelles données (les données de tests).

Partie 2 :
Les quartiles sont des valeurs qui divisent l'ensemble des données en quatre parties égales pour mieux voir la répartition des données.
Pour les calculer j'ai utilisé la fonction 'quantile' de Pandas.
Pour un attribut on regarde le gain qu'on a avec la valeur de split à chaque quartile, et on regarde quelle valeur de split est la meilleure (celle qui sépare le mieux les instances) et on la garde.

Partie 3 :
Je voudrai préciser que je n'ai pas utilisé mes propres données mais celles qui sont fournies.
J'ai mis les matrices à la fin de mon rapport

Partie 4 :
Question 1 :
Le choix de notre modèle dépendra de notre utilisation et du contexte d'utilisation. Si on veut éviter à tout prix les faux négatifs, par exemple dans le cas d'un diagnostic médical où l'on doit vraiment détecter les personnes positives, on peut se concentrer sur le recall et prendre le modèle avec le meilleur recall. Mais si on préfère une moyenne harmonique entre le recall et la précision, le F1-score est meilleur, surtout si on a des classes déséquilibrées. Chez nous les classes sont plutôt déséquilibrés, on a beaucoup plus de class1 (720) que de class3 (202), donc je privilégierais le F1-score et je prendrais le réseau de neurones tanh 10-8-6.

Question 2 :
Si je devais justifier les décisions du modèle pour que mes patients puissent comprendre les décisions prisent par le modèle je choisirai sans doute l'arbre de décision. En effet l'arbre de décision est beaucoup plus facile à comprendre qu'un réseau de neurone. C'est déjà compliqué pour un informaticien de comprendre le réseau de neurones donc pour des gens lambda ça doit l'être encore plus. Donc pour que mes patients puissent comprendre au mieux les décisions il faut leur présenter le modèle le plus simple à comprendre (ici l'arbre de décisions). De plus les différences de performances ne sont pas très grandes.

Pour les arbres

Matrice confusion arbre y-pred-DT4

	130	19	6	1	class0
	15	147	2	2	class1
actual	25	20	44	0	class2
	12	8	23	6	class3

predicted

Metrics arbre y-pred-DT4

	Accuracy	Precision	Recall	F1-score
Class0	0.83043478	0.71428571	0.83333333	0.76923077
Class1	0.85652174	0,757732	0,885542	0,816667
Class2	0.83478261	0,58667	0,494382	0,536585
Class3	0.9	0,666667	0,122449	0,206897

Matrice confusion arbre y-pred-DT5

	126	14	10	6	class0
	12	140	9	5	class1
actual	12	5	67	5	class2
	7	5	16	21	class3

predicted

Metrics arbre y-pred-DT5

	Accuracy	Precision	Recall	F1-score
Class0	0.8673913	0.802548	0.807692	0.805112
Class1	0.89130435	0.853659	0.843373	0.848485
Class2	0.87608696	0.656863	0.752809	0.701571
Class3	0.90434783	0.567568	0.428571	0.488372

Matrice confusion arbre y-pred-DT6

	130	14	10	2	class0
	9	147	3	7	class1
actual	9	7	71	2	class2
	5	7	11	26	class3

predicted

Metrics arbre y-pred-DT6

	Accuracy	Precision	Recall	F1-score
Class0	0.89347826	0.849673	0.833333	0.841424
Class1	0.89782609	0.840000	0.885542	0.862170
Class2	0.90869565	0.747368	0.797753	0.771739
Class3	0.92608696	0.702703	0.530612	0.604651

Pour les réseaux de neurones

Matrice confusion NN Relu 10-8-4

	136	8	5	7	class0
	0	163	2	1	class1
actual	2	4	81	2	class2
	2	12	9	26	class3
			predicted		

Metrics arbre NN Relu 10-8-4

	Accuracy	Precision	Recall	F1-score
Class0	0.94782609	0.971429	0.871795	0.918919
Class1	0.94130435	0.871658	0.981928	0.923513
Class2	0.94782609	0.835052	0.910112	0.870968
Class3	0.92826087	0.722222	0.530612	0.611765

Matrice confusion NN Relu 10-8-6

	135	9	5	7	class0
	0	155	4	7	class1
actual	2	3	80	4	class2
	2	5	10	32	class3
			predicted		

Metrics arbre NN Relu 10-8-6

	Accuracy	Precision	Recall	F1-score
Class0	0.94565217	0.971223	0.865385	0.915254
Class1	0.93913043	0.901163	0.933735	0.917160
Class2	0.93913043	0.808081	0.898876	0.851064
Class3	0.92391304	0.640000	0.653061	0.646465

Matrice confusion NN tanh 10-8-6

	144	5	1	6	class0
	0	162	2	2	class1
actual	3	4	81	1	class2
	2	10	5	32	class3
			predicted		

Metrics arbre NN tanh 10-8-6

	Accuracy	Precision	Recall	F1-score
Class0	0.96304348	0.966443	0.923077	0.944262
Class1	0.95	0.895028	0.975904	0.933718
Class2	0.96521739	0.910112	0.910112	0.910112
Class3	0.94347826	0.780488	0.653061	0.711111