Projet IA

Compte rendu

Partie n°1 - Préparation des données

Question n°1 : Combien d'attributs comportent ces données ?

Les données comportent 14 attributs, allant de A à N.

Question n°2 : En combien de classes différentes les instances sont-elles catégorisées ?

Les instances sont catégorisés en 4 classes : 0, 1, 2 et 3

Question n°3 : Combien d'instances chaque classe compte-elle ?

Grâce à la méthode `value_counts`, on peut connaitre le nombre d'occurence de chaque valeur d'une colonne. Ainsi, la classe 0 possède 674 instances, la classe 1 en possède 908, la classe 2 en possède 472 et la classe 3 en possède 244.

Question n°4 : Les données sont-elles linéairement séparable ?

Avec le module seaborn et sa fonction pairplot, on peut afficher toute les combinaisons d'attributs 2 à 2 dans des nuages de points (182 graphiques). On constate qu'aucun de ses graphiques permettent une séparation linéaire des données, ainsi, on sait qu'elles ne sont pas linéairement séparable.

Question n°5 : Aurez-vous besoin, pour l'un des types de modèle ou les deux, d'utiliser un encodage en one-hot ? de normaliser les données ?

Le codage *one-hot* des étiquettes est nécessaire pour le réseau de neurone. En effet, la sortie des ce modèle est probabiliste (Une probabilité d'appartenance sera attribué à chaque classe) il est donc nécessaire de segmenter clairement les différentes classes.

La normalisation sera aussi nécessaire pour le réseau de neurone, car certains attributs ne sont pas du tout dans le même ordre de grandeur (La B et le C par exemple). Les normaliser permettra d'éviter d'avoir des poids trop élevé, privilégiant un attribut plutôt qu'un autre.

Question n°6 : Rappelez l'intérêt de séparer les données en un jeu d'entrainement et un jeu de test.

Séparer les données permet par la suite d'évaluer le modèle. En effet, il ne sera pas pertinent d'évaluer le modèle sur des données qui lui ont servi à s'entrainer car il serait forcément très juste avec ces donnés, mais pas nécessairement sur des nouvelles données.

Nathanaël Derousseaux

Partie n°2 - Mise en oeuvre des modèles

Modèle n°1 : Arbre de décision binaire

Les quartiles d'un attribut correspondent aux 3 valeurs permettant de couper notre jeu de donnée en quatre parties d'effectif égal. Pour ce faire, il suffit d'appeler la méthode `quantile`, avec comme valeurs successives 0.25, 0.5 et 0.75.

```
for quartile in np.arange(0.25, 1, 0.25):
    split_value_l = self.data.quantile(quartile)[attribute]

```

Découper son jeu de donnée en quantiles avec pandas et numpy

Dans mes tests, j'ai pu obtenir les performances suivantes :

- Profondeur de 3 : **71.7%** de données correctement classifiées
- Profondeur de 4 : **74.3%** de données correctement classifiées
- Profondeur de 5 : **77.8%** de données correctement classifiées
- Profondeur de 6 : **76.7%** de données correctement classifiées
- Profondeur de 7 : **78.6%** de données correctement classifiées
- Profondeur de 8 : **77.4%** de données correctement classifiées

J'ai donc gardés les prédictions correspondant aux profondeurs 5 et 7 pour la partie 3 du projet.

Modèle n°2 : Réseaux de neurones artificiels

Lors de mes tests, j'ai obtenu les valeurs d'erreur suivantes :

- Réseau tanh, [10,8,6] : **0.0993**
- Réseau tanh, [10,8,4] : **0.1221**
- Réseau tanh, [6,4] : **0.1196**
- Réseau relu, [10,8,6] : **0.1415**
- Réseau relu, [10,8,4] : **0.1474**
- Réseau relu, [6,4] : **0.1537**

J'ai donc gardés les prédictions associées aux réseaux tanh, [10,8,6], tanh, [6,4], relu [10,8,6] et relu, [10,8,4].

Partie n°3 - Analyse des modèles

J'ai réussi à générer mes propres prédictions. Donc pour cette partie, je présenterai mes propres résultats.

A - Métriques de performances

Performances

	Classes	C0	C1	C2	C3
Arbre Profondeur: 5	**Accuracy**	85.22 %	88.48%	91.30%	90.65%
	Precision	77.50%	83.43%	78.16%	56.82%
	Recall	79.49%	84.94%	76.40%	51.02%
	F1-Score	78.48%	84.18%	77.27%	53.76%
Arbre Profondeur: 7	**Accuracy**	88.26 %	86.96%	91.30%	90.87%
	Precision	85.42%	79.44%	76.34%	58.14%
	Recall	78.85%	86.14%	79.78%	51.02%
	F1-Score	82.00%	82.66%	78.02%	54.35%
Réseau Activation: tanh Taille : [10,8,6]	**Accuracy**	95.65%	95.43%	95 %	93.91%
	Precision	97.22%	89.62%	83.67%	80 %
	Recall	89.74%	98.80%	92.13%	57.14%
	F1-Score	93.33%	93.98%	87.70%	66.67%
Réseau Activation: tanh Taille : [6,4]	**Accuracy**	93.91%	93.91%	93.91%	93.04%
	Precision	93.84%	87 %	83.52%	71.79%
	Recall	87.82%	96.99%	85.39%	57.14%
	F1-Score	90.73%	92.00%	84.44%	63.64%
Réseau Activation: relu Taille : [10,8,6]	**Accuracy**	94.35%	94.13 %	92.39 %	93.04 %
	Precision	87.36 %	86.77 %	89.71 %	79.31 %
	Recall	97.44 %	98.80 %	68.54 %	46.94 %
	F1-Score	92.12 %	92.39 %	77.71 %	58.97 %
Réseau Activation: relu Taille : [10,8,4]	**Accuracy**	91.30 %	93.48 %	89.35 %	87.61 %
	Precision	85.80 %	86.96 %	88.47 %	43.55 %
	Recall	89.10 %	96.39 %	51.69 %	55.10 %
	F1-Score	87.42 %	91.43 %	65.25 %	48.65 %

B - Matrices de confusion

Matrices de confusion

			Labels predis			
			0	**1**	**2**	**3**
Vrais labels	**Arbre Profondeur: 5**	**0**	124	16	7	13
		1	13	141	9	6
		2	9	5	68	5
		3	10	4	5	25
	Arbre Profondeur: 7	**0**	123	9	7	5
		1	19	143	6	12
		2	5	10	71	7
		3	9	4	5	25
	Réseau Activation: tanh Taille : [10,8,6]	**0**	140	0	2	2
		1	6	164	4	9
		2	5	1	82	10
		3	5	1	1	28
	Réseau Activation: tanh Taille : [6,4]	**0**	137	2	4	3
		1	6	161	7	10
		2	5	2	76	8
		3	8	1	2	28
	Réseau Activation: relu Taille : [10,8,6]	**0**	152	1	12	9
		1	3	164	11	11
		2	1	0	61	6
		3	0	1	5	23
	Réseau Activation: relu Taille : [10,8,4]	**0**	139	4	12	7
		1	5	160	9	10
		2	0	1	46	5
		3	12	1	22	27

Partie n°4 - Le meilleur modèle

Question n°1

Les réseaux de neurones ont une exactitude et une précision bien plus élevées que les arbres, particulièrement le réseau **tanh, [10,8,6]**. Cela signifie que dans un cas ou l'on souhaite trouver un maximum de bonne prédictions (sans grandes conséquences si l'on à un faux négatif ou un faux positif), celui-ci sera le meilleur.

Même réflexion pour le rappel : Dans un cas ou l'on veux rater aucun vrai positifs (diagnostic d'une maladie grave par exemple), le réseau **tanh, [10,8,6]** est encore le meilleur. L'analyse du F1-Score nous donne aussi ces résultats (le F1 score étant une "moyenne" de la pression et du rappel).

On peut aussi constater que la classe n°3 est difficile à classifier, et ce, peut-importe le modèle utilisé.

Question n°2

Si l'on devait justifier de la décision prise par un modèle, les arbres de décisions sont toujours meilleurs. En effet, les arbres de décisions sont extrêmement simple à comprendre, il suffit de remonter l'arbre et de regarder quelles règles de split ont été appliqués. (On aurait pu le faire assez simplement dans ce projet). Par contre, pour un réseau de neurone, il est beaucoup plus dur d'expliquer le résultat. Un arbre de décision est un modèle excellent quand il s'agit de justifier le résultat.