Rapport du rojet d'Intelligence Artificielle (IA)
Le meilleur modèle

Godwin AMEGAH
Mariam DIAKITE

Mai 2021

Table des matières

1 Réponses aux questions

- **Combien d'attributs comportent ces données ?**
  les donnée sont composées de 14 attributs
- **En combien de classes différentes les instances sontelles catégorisées ?**
  Les instances sont catégorisés en 4 class (0,1,2,3)
- **Combien d'instances chaque classe compte-elle ?**
  - Class 0 : 674 instances
  - Class 1 : 908 instances
  - Class 2 : 472 instances
  - Class 3 : 244 instances
- **Les données sont-elles linéairement séparables ?**
  les données sont pas linéairement séparables. En effet une visualisation des données dans notre code permets de voir que les données sont entreméler. De ce fait nous ne pouvons pas séparer les données tel qu'aucune donnée ne soit sur la frontière de décision.
- **Rappelez l'intérêt de séparer les données en un jeu d'entrainement et un jeu de test**
  L'intérêt de faire cette répartion est que les *jeu de données d'entrainement* vont nous permettre d'ajuster les paramètres (par exemple, les poids associés aux attributs) et ensuites les *jeu de données de test* vont à leur tour nous permettre d'évaluer les performances de notre modèle.
- **Aurez-vous besoin, pour l'un des types de modèle ou les deux, d'utiliser un encodage en one-hot ? de normaliser les donnée ?**
  Pour le modèle des **réseaux de neuronne** on aura besoins d'un encodage one-hot pour encoder les étiquettes. En effet les étiquettes sont sont muni d'une relation d'ordre et de ce fait le modèle peut apprendre cette relation et produire des prédictions moins précis. D'autre part nous avons besoins de normaliser les données. En effet on observe que par les valeurs de l' `attribut C` ou l' `attribut F` tourne autour du milénaire alors que par exemple la valeur de l'`attribut A` sont de l'ordre de la dizaine. Nous devons

donc ramener les données à une même échelle (*normaliser les données* ou centrer les données autour d'une valeur).
Pour le modèle des **arbres de décision** nos n'auront pas à priori besoins de normaliser les données ni d'encoder les étiquettes car l'arbre n'est pas sensible à ces changements.

2 Organisation des fichiers

Nous avons un gros ***ia_meilleur_modele*** dossier qui contient tous nos fichiers repartient dans des sous dossiers.

— **`scripts :`** Ce dossier contient tous les fichiers sources en python et Jupiter.
  — ***ArbreDeDecision.ipynb*** : Ce fichier contient l'implemantation de l'arbre de decision
  — ***ReseauDeNeurone.ipynb*** : ce fichier contient l'implementation du reseau de neurone
  — ***Metrics.ipynb*** : ce fichier contion les toutes les fonctions qu'il faut pour le calcul des Métrique
  — ***scale.py*** : fichier contenant l'implémentation des différents type de normalisation (*z-score, min-max* ...).
  — ***utility*** : fichier contenant les fonctions d'activation ainsi que celle du calcul des erreurs.

— **`/prediction`** : ce répertoire contient les de prédiction fournies par **Mme Anne**.

— **`scripts/predictions`** : Ce dossier contient toutes nos predictions.
  — ***all_preds*** : Ce fichier contient les prédiction de Mme Anne sous formme de classe.
  — ***best_preds*** : Ce fichier contiont nos differentes prédiction de l'arbre de decision et réseau de neurone

— **donnees :** Ce dossier contient les données que l'on étudie. (*synthetic.csv*)

3 Mise en oeuvre des modèles

3.1 Les arbres de décision

3.1.1 Préparation des données

Nous avons diviser le data frame en 80% pour **les données d'entrainement** et 15% pour **les données de test** qui nous a servi à tester notre model. Ensuite nous avons diviser en les données d'entrainement en 20% pour **les données de validation** et 80% pour **les données d'entrainement**. Nous avons utiliser les données de validation pour trouver la meilleur profondeur pour ce model.

3.1.2 la mise en oeuvre de l'arbre de decision

L'élaboration des arbres de décisions passe à travers différents étapes :
Pour discrétiser les attributs, nous devions cette fois-ci utiliser la méthode des *quartiles*. Les quartiles sont les valeurs qui partagent notre distribution suivant certains critères. Elles conrrepondent donc aux *valeurs de split* ie ceux nous permettant de partitionner nos données. Pour déterminer la bonne valeur de split, nous calculons respectivement pour chaque quartile ($q_{0.25}, q_{0.5}, q_{0.75}$) le **gain d'information**. La valeur de split correspond donc au quatile qui offre le gain le plus élevé suivant un attribut donné.
Nous avions entraîner le modèle avec des profondeurs maximales différentes (entre 3 et 8.), testé les performances de nos modèles avec notre jeu de test. Pour déterminer l'arbre optimal, nous avions adopté une approche pragmatique, celle de calculer le taux d'erreur générer au niveau de chaque profondeur différente. Pour calculer le taux d'erreur nous utilisons nos prédiction générées dans le répertoire `scripts/predictions/`, en comparant la sortie *y_pred_DTX* (avec *X* la profondeur) produite par les différents arbres. Nous obtenons ainsi le graphe de la figure 1. Au vu de ce résultat nous considérons les arbres de profondeur 3,4 et 5 comme étant meilleur profondeur.

3.1.3 Prédiction

Nous avons recupérer les predictions des differents profondeurs 3,4,5 de et les avons stoquer dans des fichiers dans les dossiers `predictions/best_preds/`

— *y_pred_DT3.csv = profondeur 3*
— *y_pred_DT4.csv = profondeur 4*
— *y_pred_DT5.csv = profondeur 5*

3.2 Les réseaux de neuronnes

3.2.1 Préparation des données

Pour la préparation des données,nous avons recupérer le data frame de base pour ensuite la normaliser avec une fonction qui se trouve dans le fichier **scale.ipynb** dans le dossier **scribts**. Nous avons diviser le data frame en 80% pour **les données d'entrainement** et 20% pour **les données de test**. Ensuite nous avons diviser en les données d'entrainement en 15% pour **les données de validation** et 85% pour **les données d'entrainement**.

3.2.2 Construction du Réseau de neurone

Pour la construction de notre modele de Reseau de Neurone,nous avons mis en place 3 fonctions principales. La fonction **forward_promagation** est la passe avant ,elle utilise une fonction d'activation et donne la prediction avec la fonction **softmax**.Ensuite la fonction **backward_pass** qui fait la mise ajour des paramètres **W et B**.Pour finir la fonction **earling_stoping** qui fait appel aux deux fonctions précedentes pour entainer le model avec la methode du **earling stoping**.

3.2.3 Prédictions

Nous avons recupérer les predictions des differents models de **tanh** et **relu** et stoquer dans des fichiers. `predictions/best_preds/`

- *y_pred_DT_NN_relu6_4.csv = relu(6,4)*
- *y_pred_DT_NN_relu10_8_4.csv = relu(10,8,4)*
- *y_pred_DT_NN_relu10_8_6.csv= relu(10,8,6)*
- *y_pred_DT_NN_tanh6_4.csv= tanh(6,4)*
- *y_pred_DT_NN_tanh10_8_4.csv = tanh(10,8,4)*
- *y_pred_DT_NN_tanh10_8_6.csv= tanh(10,8,6)*

4 Analyse des modèles

> *Dans cette partie, nous utilisons nos prédictions contenu dans le répertoire `scripts/predictions/best_preds` pour effectuer le calcul des différentes métrics de nos modèles.*

Nous avons implémenté via le fichier `scripts/Metrics` le calcul des différents métriques (***accuracy, Precision, Recall, F1-score etc...***) ainsi que la ***matrice de confusion*** sans librairies extérieur à notre programme à l'exeption de celui de base (**pandas**). Vous retrouverez en ***Annexe*** les différents résultats obtenus pour chaque modèle mise en place. Par exemple la figure 2b nous renseigne sur le calcul des métrics concernant nos arbres de décisions

5 Le meilleur modèle

- En supposant par exemple que nos données représentes des anomalies de type par exemple `0` `1`, `2` et `3` et en supposant aussi que ce sont des cas grâves qui doivent être traités sous peine de sonséquences néfaste (*mort*) :), nous allons préférer les modèles qui offre plus un bon rappel pour les différentes classes. Cependant ces modèles peuvent perdre en précision mais ... celà n'est pas aussi grâve car même si un patient n'est pas atteint de l'anomalie dans la réalité, il se fera soigné tout de même est donc restera peut être en vie par contre le pas détecter par exemple une anomalie chez un patient qui est réellement atteint est dans dout problématique. les modèles idéales ddans ces cas sont par exemple ceux de **NN_relu_6_4**, **NN_tanh_10_8_6** et **DT3**.
  Dans la cas par exemple l'on checher à prédire les type de produits achetés par exemple par les clients d'un magazin en fonction des caractéristique comme le *temps qu'il fait, leur budgets ... etc*, on pourrait peut être s'autorité à perdre certaines informations surtout si par exemple un type de produit (class 3) qui est peu vendu... Dans ce cas on peut privilégier les modèles comme **NN_tanh_6_4**, **NN_tanh_10_8_6** qui offre les meilleurs *precision* ainsi que des bon *recall*.
- Les réseaux de neuronne pouvant devenir rapidement très complexe surtout avec un grand nombre de neuronnes et

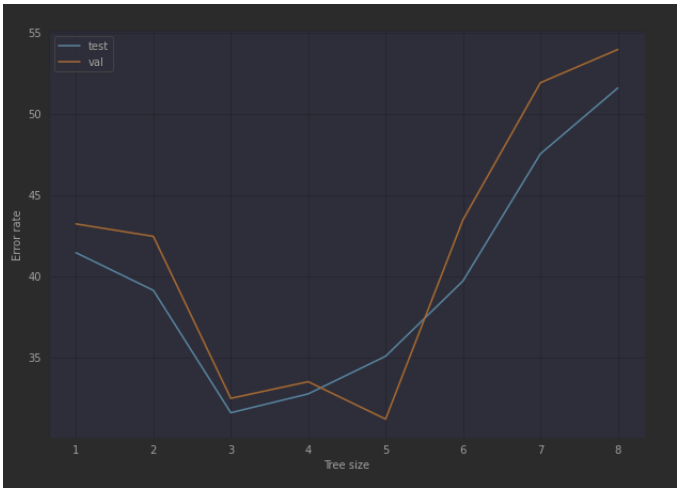

FIGURE 1 – Comparaison de différents arbres de décision

	DT3			
Classes	0	1	2	3
Accuracy	0.84	0.81	0.84	0.88
Precision	0.77	0.71	0.56	0.40
Recall	0.78	0.80	0.56	0.21
F1-Score	0.77	0.75	0.56	0.29

(a) Tree size 3

	DT4			
Classes	0	1	2	3
Accuracy	0.76	0.83	0.84	0.90
Precision	0.69	0.69	0.60	0.64
Recall	0.62	0.95	0.48	0.24
F1-Score	0.65	0.80	0.53	0.35

(b) Tree size 4

	DT5			
Classes	0	1	2	3
Accuracy	0.77	0.80	0.86	0.87
Precision	0.70	0.67	0.66	0.40
Recall	0.63	0.82	0.52	0.37
F1-Score	0.66	0.74	0.58	0.39

(c) Tree size 5

FIGURE 2 – Calcul des métrics pour les **arbres de décisions**

	NN_relu_6_4			
Classes	0	1	2	3
Accuracy	0.94	0.95	0.93	0.91
Precision	0.95	0.91	0.77	0.59
Recall	0.86	0.95	0.94	0.46
F1-Score	0.90	0.93	0.84	0.51

(a) NN relu 6-4

	NN_relu_10_8_4			
Classes	0	1	2	3
Accuracy	0.94	0.94	0.94	0.92
Precision	0.98	0.87	0.81	0.69
Recall	0.86	0.98	0.92	0.54
F1-Score	0.91	0.91	0.87	0.60

(b) NN relu 10-8-4

	NN_relu_10_8_6			
Classes	0	1	2	3
Accuracy	0.94	0.94	0.94	0.92
Precision	0.98	0.89	0.80	0.62
Recall	0.84	0.94	0.94	0.62
F1-Score	0.91	0.92	0.86	0.62

(c) NN relu 10-8-6

FIGURE 3 – Calcul des métrics des modèles de réseaux- **fonction Relu**

	NN_tanh_6_4			
Classes	0	1	2	3
Accuracy	0.90	0.95	0.92	0.89
Precision	0.86	0.88	0.72	0.0
Recall	0.88	0.99	0.92	0.0
F1-Score	0.87	0.93	0.81	0.0

(a) NN tanh 6-4

	NN_tanh_10_8_4			
Classes	0	1	2	3
Accuracy	0.94	0.94	0.92	0.89
Precision	0.91	0.88	0.72	0.0
Recall	0.91	0.98	0.98	0.0
F1-Score	0.91	0.83	0.83	0.0

(b) NN tanh 10-8-4

	NN_tanh_10_8_6			
Classes	0	1	2	3
Accuracy	0.97	0.96	0.97	0.94
Precision	0.97	0.92	0.89	0.75
Recall	0.93	0.98	0.94	0.65
F1-Score	0.95	0.94	0.91	0.70

(c) NN tanh 10-8-6

FIGURE 4 – Calcul des métrics des modèles de réseaux- **fonction Tangente**

Modèle DT3					
		0	1	2	3
True label	**0**	96	17	10	0
	1	12	97	7	6
	2	15	7	35	6
	3	2	16	11	8
	Predicted label				

(a) Tree size 3

Modèle DT4					
		0	1	2	3
True label	**0**	77	34	9	3
	1	1	116	4	1
	2	24	8	30	1
	3	10	11	7	9
	Predicted label				

(b) Tree size 4

Modèle DT5					
		0	1	2	3
True label	**0**	77	33	5	8
	1	8	100	5	9
	2	18	8	33	4
	3	7	9	7	14
	Predicted label				

(c) Tree size 5

FIGURE 5 – Matrice de confusion des modèle d'**arbre de décision**

Modèle relu_6-4					
		0	1	2	3
True label	**0**	106	5	4	8
	1	1	116	4	1
	2	1	0	59	3
	3	3	7	10	17
	Predicted label				

(a) Tree size 4

Modèle relu_10_8_4					
		0	1	2	3
True label	**0**	106	7	4	6
	1	0	119	2	1
	2	1	2	58	2
	3	1	9	7	20
	Predicted label				

(b) relu 10-8-4

Modèle relu_10_8_6					
		0	**1**	**2**	**3**
True label	**0**	104	8	4	7
	1	0	115	3	4
	2	1	0	59	3
	3	1	5	8	23
	Predicted label				

(c) relu 10-8-6

FIGURE 6 – Matrice de confusion des modèles de **réseaux de neuronne fonction relu**

Modèle tanh_6_4					
		0	**1**	**2**	**3**
True label	**0**	108	8	7	0
	1	1	121	0	0
	2	3	2	58	0
	3	14	7	16	0
	Predicted label				

(a) Tree size 4

Modèle tanh_10_8_4					
		0	**1**	**2**	**3**
True label	**0**	112	7	4	0
	1	0	119	3	0
	2	1	0	62	0
	3	10	10	17	0
	Predicted label				

(b) Tree size 5

Modèle tanh_10_8_6					
		0	**1**	**2**	**3**
True label	**0**	114	3	0	6
	1	0	119	2	1
	2	2	1	59	1
	3	1	7	5	24
	Predicted label				

(c) Tree size 6

FIGURE 7 – Matrice de confusion des modèles de **réseaux de neuronne fonction tanh**