Angélique Delevingne
M1 Informatique – option Big Data
19000955

TP1 - Prima Indians Diabète
Cours : Fouilles de données textuelles
mars – 2023

1. Objectif du TP

Le TP a pour objectif de réaliser un réseau de neurones capable de classer les patients atteints ou non de diabète.
Le jeu de données est « Prima Indians Diabète » provenant de Kaggle.
Cet exercice est réalisé en Python à l'aide de la bibliothèque Keras.

2. Comment utiliser le programme

Le fichier ***requirements.txt*** contient toutes les bibliothèques à installer pour tester le code.
Pour installer les dépendances du projet, exécuter la requête suivante :

- ***pip install -r requirements.txt***

Le fichier ***tp1.py*** permet de faire de la classification des patients atteints ou non de diabète à l'aide du fichier Pima_Indians_Diabetes.csv.

Exemple d'utilisation du script:
python3 tp1.py

3. IA : Utilisation des données et Tests

A) Les données

Le CSV contient 769 patients représentant tous des femmes,. Les ensembles de données se compose de 8 caractéristiques prédictives médicales et d'une variable cible, indiquant si la personne est atteinte ou non du diabète.
Les 8 caractéristiques sont :

1. Pregnancies (grossesses) : Le nombre de grossesses que la patiente a eues.
2. Glucose : La concentration de glucose dans le sang de la patiente.
3. BloodPressure (pression artérielle) : La pression artérielle de la patiente (en mmHg).
4. SkinThickness (épaisseur de la peau) : L'épaisseur du pli cutané du triceps de la patiente (en mm).
5. Insulin : La concentration d'insuline dans le sang de la patiente (en mU/L).
6. BMI (indice de masse corporelle) : L'indice de masse corporelle de la patiente, qui est calculé en divisant le poids de la patiente (en kg) par le carré de sa taille (en mètres).
7. DiabetesPedigreeFunction : Une mesure de la probabilité de diabète chez les parents et les ancêtres de la patiente, qui est basée sur les antécédents familiaux.
8. Age : L'âge de la patiente (en années).

9. Outcome (résultat) : Une variable binaire qui indique si la patiente a développé un diabète de type 2 (1) ou non (0) dans les 5 ans suivant les observations.

B) IA

a) Les étapes

Les différentes étapes de l'algorithme sont les suivantes :

- Lecture du fichier Pima_Indians_Diabetes.csv,
- Pré-traitement des données : Transformation des valeurs à 0 en valeur NULL,
  Supprime les lignes avec des valeurs manquantes,
  Standardiser les caractéristiques
- Séparation des données d'entraînements et données tests,
- Application du réseau de neurone,
- Diagrammes

b) Le réseau de neurone

Tout d'abord il faut configurer le modèle Keras. Le constructeur est le Sequential. Nous utilisons le modèle séquentiel car notre réseau est constitué d'un empilement linéaire de couches.

Le modèle est constitué de 3 couches. La couche d'entrée qui spécifie le nombre d'entrée, qui dans notre cas est de 8 caractéristiques. La fonction d'activation utilisée est une unité linéaire redressée, ou ReLU. ReLU est la fonction d'activation la plus largement utilisée car elle est non linéaire et a la capacité de ne pas activer tous les neurones en même temps.

J'ai utilisé le loss binary_crossentropy car il existe deux classes cibles.

C) Tests

Les différents tests exécutées :

- Avec et sans pré-traitement sur les données
- La modification du réseau de neurone
- Le temps d'apprentissage

4. Résultats

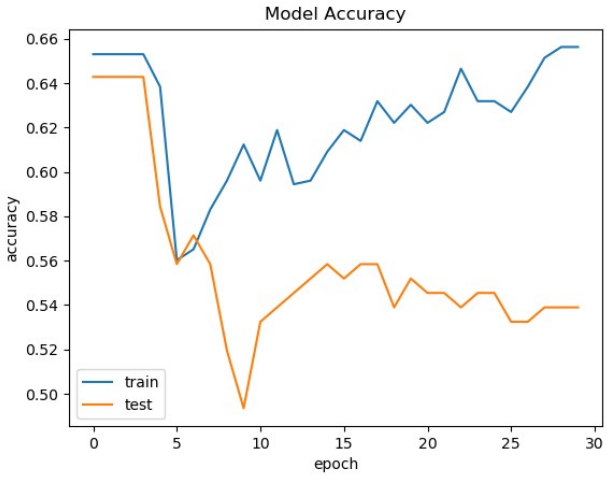
Dans cette partie nous allons voir les divers résultats des tests abordés dans la partie précédente.

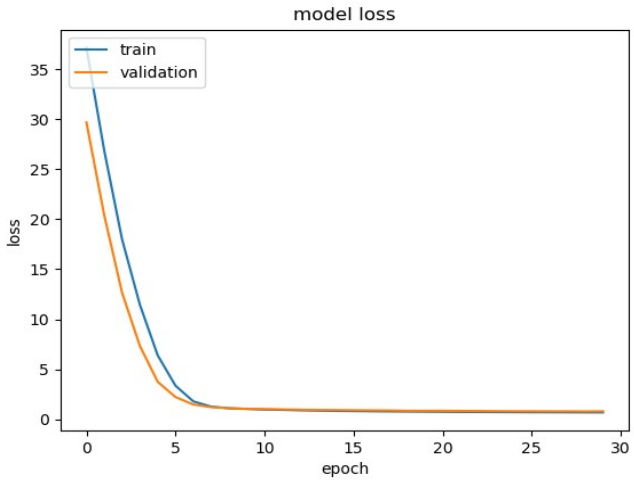
Test numéro 1 : Comparaison des résultats avec et sans prétraitement des données

Le prétraitement des données est le fait de transformer les valeurs 0 par des valeurs NULL, pour les colonnes suivantes : Glucose Pression artérielle Skinthickness Insuline IMC.
Il est préférable de remplacer Zeros par Nan car après cela, les compter serait plus facile et les zéros doivent être remplacés par des valeurs appropriées.
J'ai supprimé les lignes où il y avait ces valeurs manquantes pour voir si ça influençait les résultats.
J'ai également standardiser les données.

Résultat sans prétraitements des données

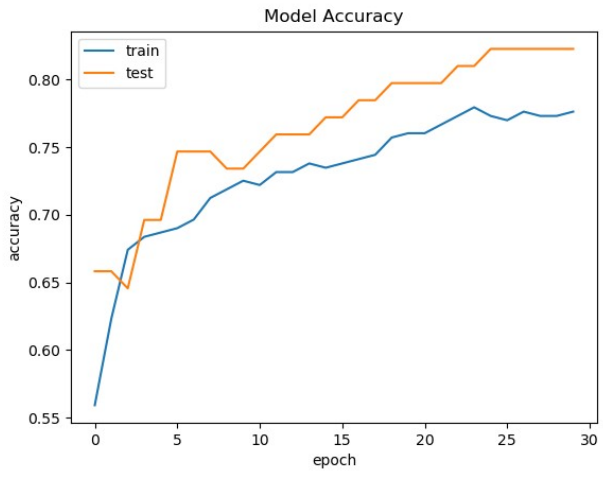
Accuracy	Precision	Recall	F1 score
53.89	41.8	37.0	39.31

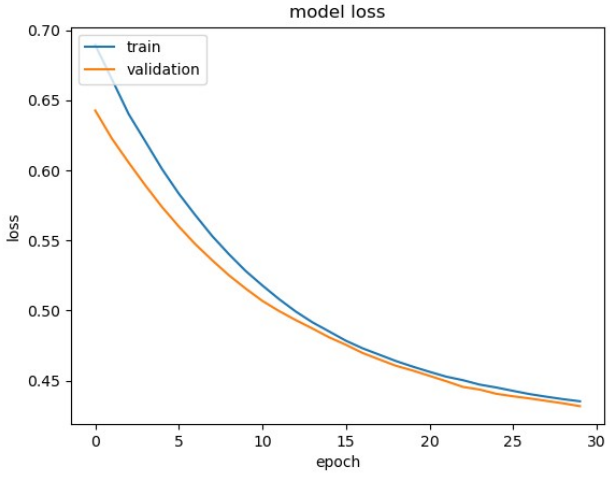




Résultat avec prétraitements des données

Accuracy	Precision	Recall	F1 score
82.27	64.0	76.19	69.56





On observe pour le test n°1 un meilleur taux de bonne classification pour l'algorithme avec le prétraitement des données soit 82 % face à 53% sans prétraitements.
On peut en conclure que le programme est plus performant en appliquant le prétraitement au dataset.

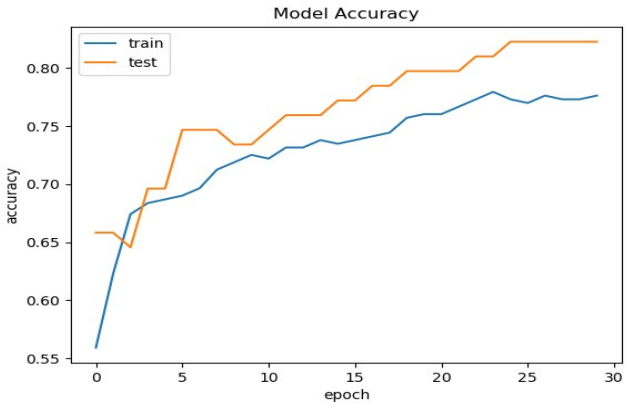
Test numéro 2 : La modification du réseau de neurone

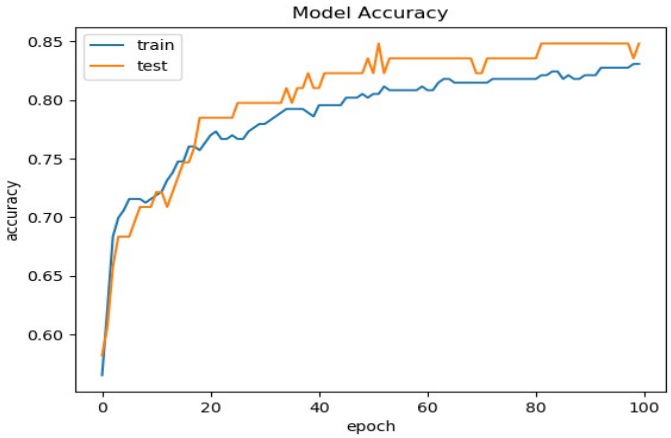
Pour le réseau de neurone, j'ai voulu tester si le nombre de neurone influence le résultat.
Pour les deux couches j'ai ainsi modifié le nombre de neurone

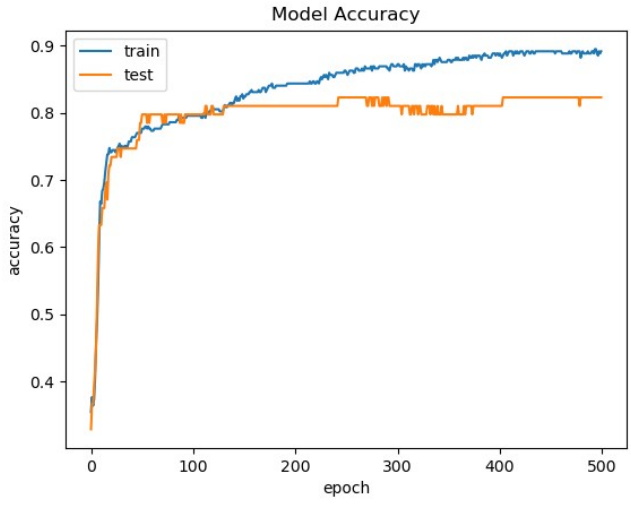
Nombre de neurone par couche	Accuracy	Precision	Recall	F1 score
2 & 6	77.21	52.0	68.42	59.09
12 & 8	81.01	68.0	70.83	69.38
200 & 40	82.27	68.0,	73.91	70.83
1000 & 400	84.81	80.0	74.07	76.92

On observe que plus il y a de neurone, meilleures sont les résultats. A l'inverse moins il y a de neurone moins les résultats sont bons.
On en conclus que le nombre de neurone influence les résultats.

Test numéro 3 : Le temps d'apprentissage







On observe 3 diagramme qui représente le taux de bonne classification en pourcentages en fonction du nombre de répétions de l'algorithme.

On remarque sur le diagramme 1 pour 30 répétions, le réseau de neurone est encore entrain d'apprendre sur les données. On a envion 85% de bonne classification.

On remarque sur le diagramme 2 pour 100 répétions, les données d'entraînements se stabilise avec les données tests.

Enfin on remarque sur le diagramme 3 pour 500 répétions de l'algorithme, les données d'entraînements font du sur-apprentissage au bout de 120 répétions.

On peut en conclure qu'il faut faire environ 100 répétions de l'algorithme pour avoir le meilleur taux de classification.

Bilan des tests :

Pour avoir un bon taux de classification il faut :

- Faire un pré-traitement des données : Supprimer les patients qui ont des caractéristiques non remplies et standardiser les données,
- Avoir beaucoup de neurones
- Faire 100 répétions de l'algorithme pour éviter le sous ou le sur apprentissage

5. Conclusion

C'est avec énormément d'intérêt que j'ai réalisé ce TP.
J'ai pu découvrir la librairie Keras en python.
Mon projet m'a permis de revoir et de manipuler les réseau de neurone en utilisant une librairie.
J'ai également appris l'importance de revoir et optimiser le jeu de données pour avoir une bonne classification.