UNIVERSITE MOULAY ISMAIL –UMI-

BACHELOR EN GENIE INFORMATIQUE -BGI-

Projet

Intitulé :

Evaluation De Plusieurs Modèle De Machine Learning et Deep Learning.

Réalisé par :

- **Ouhdidou Ayoub**

Encadré par :

- Pr, Ba-Ichou Ayoub

- Pr, Bekri ALi

Soutenu le 4 février 2022 devant le jury :

Pr. Ba-Ichou Ayoub, Professeur à la Faculté des Sciences- Meknès

Pr. Bekri Ali, Professeur à la Faculté des Sciences- Meknès

Pr. Oubelkacem Ali, Professeur à la Faculté des Sciences- Meknès

Pr. Benhlima Said, Professeur à la Faculté des Sciences- Meknès

Année Universitaire : 2021-2022

Dédicace

Je dédie ce travail à la famille OUHDIDOU, qu'elle trouve en ce travail l'expression de ma profonde gratitude pour tout son soutien et tous ses encouragements.

Remerciements

Ce travail est le fruit de la combinaison d'efforts de plusieurs personnes. Je remercie tout d'abord le tout puissant qui, par sa grâce m'a permis d'arriver au bout de mes efforts en me donnant la santé, la force, le courage et en me faisant entourer des merveilleuses personnes dont je tiens à remercier. Je remercie :

Monsieur Ayoub pour son encadrement sans faille, sa rigueur au travail, ses multiples conseils, ses orientations.

Mes Professeurs, Encadrants, Professeur Bekri Ali et Oubelkacem Ali pour leurs contrôles et leurs orientations.

Tous les enseignants de Bachelor, pour leurs enseignements de qualité et leurs conseils qui nous ont permis de poursuivre notre itinéraire académique jusqu'à présent ;

Mes Frères et sœurs pour leurs encouragements durant tout mon parcours ;

Mes camarades, amis et connaissances ;

Tous ceux qui de près ou de loin ont contribué à l'accomplissement de ce travail.

AYOUB OUHDIDOU.

Plan

Abstract

Grace à l'évolution de la technologie comme l'Intelligence Artificielle (IA), le cloud, le Big Data, la blockchain et l'Internet des Objets, notre monde de vie change continuellement et rapidement. En ce jour, ces technologies ont envahi d'une manière très significative notre vie quotidienne dans presque tous les domaines. L'IA et le Big Data sont deux technologies en plein essor avec beaucoup de promesses d'application pour les entreprises de toutes les industries. Ces deux technologies sont fortement liées, en effet les données massives sont généralement le point de départ de toute stratégie en IA.

L'IA est une discipline scientifique qui cherche à travers un ensemble de programmes informatiques à créer et à simuler l'intelligence humaine, comme la compréhension des langages naturels, le raisonnement, la perception visuelle ou auditive, etc. L'IA trouve sa place dans de plus en plus de secteurs de notre vie quotidienne: des voitures plus intelligentes, un trafic routier plus sûr, des prévisions météorologiques plus précises, de meilleurs diagnostics médicaux et dans le domaine calculatoire etc.

Introduction Général

Ce projet qui j'ai été accordé consiste à concevoir et réaliser un programme de classification et de prédiction d'occurrence de diabètes à partir des caractéristique (Age, BMI, Insulin …) des personnes à l'aide d'un ensemble de données. Cet ensemble de données provient de l'Institut national du diabète et des maladies digestives et rénales. Dans ce sens, mon projet répond proportionnellement aux besoins de notre cahier des charges, pour la première partie j'ai entrainé plusieurs classificateurs (régression logistique et Ridge, SVM, KNN, Decision Tree) afin de trouver le meilleur classificateur pour le comparer avec le modèle de Deep Learning (réseau de neurones) de la partie 2 et enfin de créer une interface graphique d'utilisateur, en utilisant les bibliothèques graphiques de Python, pour but de simplifier l'utilisateur de nos modèles.

Lors de ce projet, on va particulièrement utiliser le langage de programmation Python.

Plusieurs outils et bibliothèques ont été utilisés permettant le développement des différents modules réalisés au cours du projet.

Le résultat de ce projet est un programme capable de classifier correctement les personnes selon leurs caractéristiques.

Ce rapport se décompose en quatre parties :

- Introduction général du projet : comprenant la présentation du cahier de charges, la description du sujet et sa problématique.
- Méthodes et matériels : comprenant les différent modèles que j'ai utilisé et ses détails.
- Résultats et Discussion : comprenant les résultats et les comparaisons entre les modèles.
- Outils et interface graphique : comprenant les différents outils utilisées et l'interface graphique réalisée.
- Conclusion : comprenant une conclusion générale sur le projet.

Keywords : Régression logistique et Ridge, KNN, SVM, Decision Tree.

Méthodes et matériels

- ## Descriptions des données :

Premièrement on a utilisé une base de données des personnes avec les caractéristiques suivant : Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age et l'état médical de la personne : outcome.

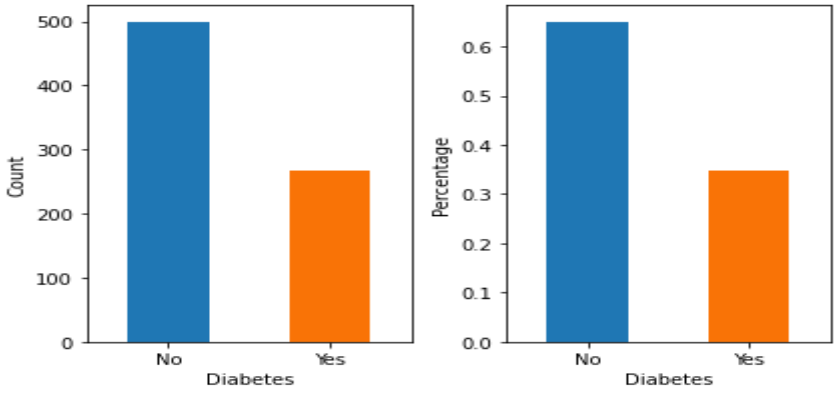
Notre Base de données a une taille de 768 personnes :

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

Alors, voici une description détailles de notre données :

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Pour bien comprendre nos données on a ploté deux histogrammes qui montrent les deux catégories (infectée ou pas).



- Il y a 268 (34.90%) personnes atteints de diabète et les 500 (65.10%) autres qui n'ont pas reçu de diagnostic de maladie.

Remarque :

Avant de détailler les différents modèles on a normalisé et standardisé les données à l'aide de (**StandardScaler()**) afin de transformer nos données pour que leur distribution ait une valeur moyenne de 0 et un écart-type de 1, en plus d'éviter les valeurs nulle et de convertir les valeurs de Int à float.

• Machine Learning :

Nous utiliserons différentes méthodes de classification telles que la régression logistique et régression Ridge, KNN, SVM, Arbres de décision, avec une taille de traitement de 80%.Nous allons effectuer tout cela avec Scikit-learn (Python).

○ La régression logistique :

- **La régression logistique** ou **modèle logit** est un modèle de régression binomiale. Comme pour tous les modèles de régression binomiale, il s'agit de modéliser au mieux un modèle mathématique simple à des observations réelles de modéliser au mieux un modèles mathématique simple à des observations réelles nombreuses. En d'autres termes d'associer à un vecteur de variables aléatoires $(x_1, \ldots, x_K)$ une variables aléatoires binomiale génériquement notée Y. La régression logistique constitue un cas particulier de modèle linéaire généralisé. Elle est largement utilisée en apprentissage automatique.
- Dans notre cas on a une classification logistique, on a la réaliser avec **SGDClassifier** avec le paramètre **loss='log',** un max d'itération égal à 1000 et un état d'erreur égal à 0.001. Pour obtenir une bonne classification et un modèle puissant.

```
SGDClassifier(eta0=0.001, loss='log')
```

○ La régression Ridge :

- **La régression Ridge (dans notre cas c'est une classification)** est une méthode d'estimation des coefficients des modèles de régression multiple dans des scénarios où les variables linéairement indépendantes sont fortement corrélées. Il a été utilisé dans de nombreux domaines, notamment l'économétrie, la chimie et l'ingénierie.
- La théorie a été introduite pour la première fois par Hoerl et Kennard en 1970 dans leurs articles Technometrics «Régressions RIDGE: estimation biaisée de problèmes non orthogonaux» et «Régressions RIDGE: applications dans des problèmes non orthogonaux».
- La régression de Ridge a été développée comme solution possible à l'imprécision des estimateurs des moindres carrés lorsque les modèles de régression linéaire ont des variables indépendantes multicolinéaires

(hautement corrélées) - en créant un estimateur de régression de Ridge (RR). Cela fournit une estimation plus précise des paramètres de Ridge, car sa variance et son estimateur carré moyen sont souvent plus petits que les estimateurs des moindres carrés précédemment dérivés.

$$\widehat{\beta}_{\mathbf{ridge}} = (X^T X + kI_p)^{-1} X^T y$$

- Alors on a réalisé notre modèle à l'aide de **RidgeClassifier**. Et pour que notre modèle soit puissant on a fait une boucle afin de trouver l'alpha correspond à l'erreur minimale.

```
#### dans les paramaitre de Ridge on aun alpha qui réalise l'inversion de la matrice utilisable par le model Ridge
#### donc on fait une boucle pour choisir notre alpha
model_RIDGE_= RidgeClassifier(alpha=1, class_weight=None, copy_X=True, fit_intercept=True,
                max_iter=None, normalize=True, random_state=None, solver='auto',
                tol=0.001)
model_RIDGE_.fit(xtrain, ytrain)
error = 1 - model_RIDGE_.score(xtest,ytest)
min=error
alp=2
print('Erreur: %f' % error)
print('E')
for i in range(2,5):
    model_RIDGE_ = RidgeClassifier(alpha=i, class_weight=None, copy_X=True, fit_intercept=True,
                max_iter=None, normalize=True, random_state=None, solver='auto',
                tol=0.001)
    model_RIDGE_.fit(xtrain, ytrain)
    error = 1 - model_RIDGE_.score(xtest,ytest)
    print('Erreur: %f' % error)
    if(min>error):
        min=error
        alp=i

min,alp
```

- <u>Le modèle KNN :</u>
  - Le k-NN est le diminutif de k Nearest Neighbors. C'est un algorithme qui peut servir autant pour la classification que pour la régression. Il est surnommé « nearest neighbors » (plus proches voisins, en français) car le principe de ce modèle consiste en effet à choisir les **k** données les plus proches du point étudié afin d'en prédire sa valeur.
  - Modélisation de notre modèle knn : afin de trouver le meilleur **k** on doit faire un boucle de 2 à 15 qui calcule les erreurs et trouve le k correspond à l'erreur minimal.

```
### boucle pour trouver le meilleur K
knn_ = neighbors.KNeighborsClassifier(n_neighbors=2)
knn_.fit(xtrain, ytrain)
error = 1 - knn_.score(xtest, ytest)
min=error
k=2
print('Erreur: %f' % error)
print('E')
for i in range(3,15):
    knn_ = neighbors.KNeighborsClassifier(n_neighbors=i)
    knn_.fit(xtrain, ytrain)
    error = 1 - knn_.score(xtest, ytest)
    print('Erreur: %f' % error)
    if(min>error):
        min=error
        k=i

min,k
```

- Le modèle SVM :
  - Les machines à vecteurs de support ou séparateurs à vaste marge sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs avec des noyaux différents (linear, Polynomial…).
  - On a travaillé **kernel=linear** parce que la précision de **kernel=linear** est la plus grands.

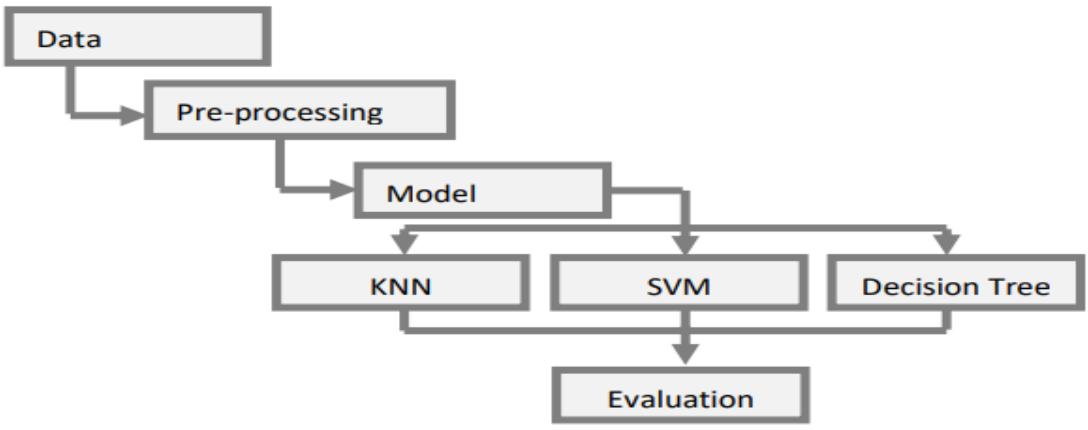
```
### la création du modèle , avec un linear kernel
classifier = svm.SVC(kernel='linear', gamma=0.01)
```

- Le modèle Arbre de décision :
  - Un arbre de décision est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches, et sont atteintes en fonction de décisions prises à chaque étape.
  - Modélisation du modèle Decision Tree: On trace l'arbre de décision à l'aide de DecisionTreeClassifier (max depth= **k**), afin de trouver le meilleur **k** on doit faire un boucle de 1 à 20 qui calcule les erreurs et trouve le k correspond à l'erreur minimal.

```
clf_ = DecisionTreeClassifier(max_depth=1)
clf_.fit(X,y)
error = 1 - clf_.score(xtest, ytest)
min=error
k=2
print('Erreur: %f' % error)
print('E')
for i in range(2,20):
    clf_ = DecisionTreeClassifier(max_depth=i)
    clf_.fit(xtrain, ytrain)
    error = 1 - clf_.score(xtest, ytest)
    print('Erreur: %f' % error)
    if(min>error):
        min=error
        k=i

min,k
```

- Conclusion :



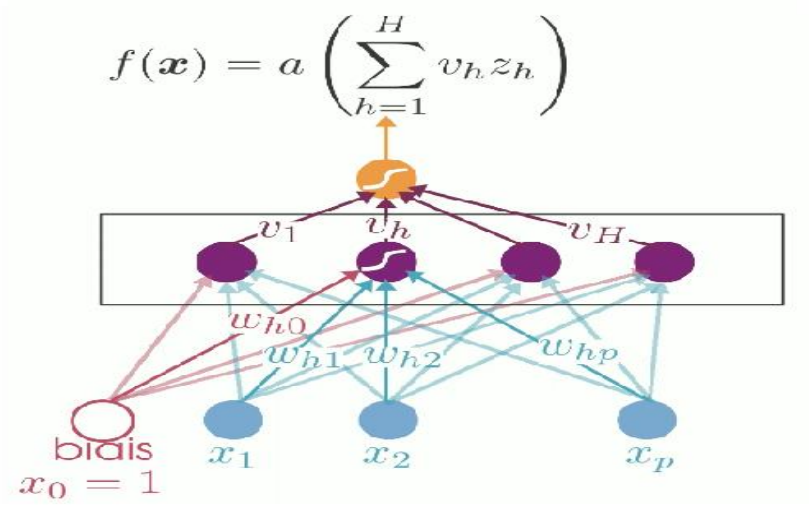
- ## Deep Learning :

  - Le modèle Réseau de neurones :

    - Un **réseau de neurones artificiels**, ou **réseau neuronal artificiel** est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.
    - Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides, et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif indépendant des idées propres de l'implémenteur, et des informations d'entrée au raisonnement logique formel .
    - En modélisation des circuits biologiques, ils permettent de tester quelques hypothèses fonctionnelles issues de la neurophysiologie, ou encore les conséquences de ces hypothèses pour les comparer au réel.
    - **Modélisation de notre réseau** :

On a utilisé Keras, C'est une des bibliothèques Python les plus puissantes et les plus faciles à utiliser pour les modèles d'apprentissage profond et qui permet l'utilisation des réseaux de neurones de manière simple. Keras englobe les bibliothèques de calcul numérique Theano et TensorFlow. Pour définir un modèle de deep Learning, on définit les caractéristiques suivantes : (nombre de couche, nombre de neurones, fonction d'activation).

➢ Choix de nombre des couches, nombre de neurones et de fonction d'activation :



- Nombre de couches :

  On a choisis 5 couche pour diminuer la valeur de loss, on n'ajouté pas une autre couche tant que la valeur de loss et d'accuracy n'est pas stagné.

- Nombre de neurones :

  On a augmenté le nombre de 40 jusqu'à loss et l'accuracy stagne, après on a ajouté une autre couche et on a augmenté le nombre de 40 jusqu'à loss et l'accuracy stagne,…mais pour la couche de sortie on a mis seulement un neurone.

- Fonction d'activation :

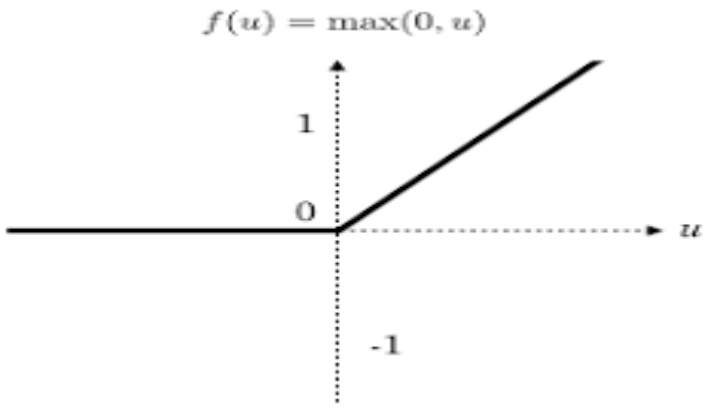
  Une fonction d'activation est une fonction mathématique utilisée sur un signal. Elle va reproduire le potentiel d'activation que l'on retrouve dans le domaine de la biologie du cerveau humain. Elle va permettre le passage d'information ou non de l'information si le seuil de stimulation est atteint. Concrètement, elle va avoir pour rôle de décider si on active ou non une réponse du neurone. Un neurone ne va faire qu'appliquer la fonction suivante :

  X = ∑ (entrée * poids) + biais

  C'est sur cette sortie que la fonction d'activation va s'appliquer.
  Alors on a choisis deux fonctions d'activation pour notre réseau :

  ✓ La fonction 'relu' : Relu = Rectified Linear Unit

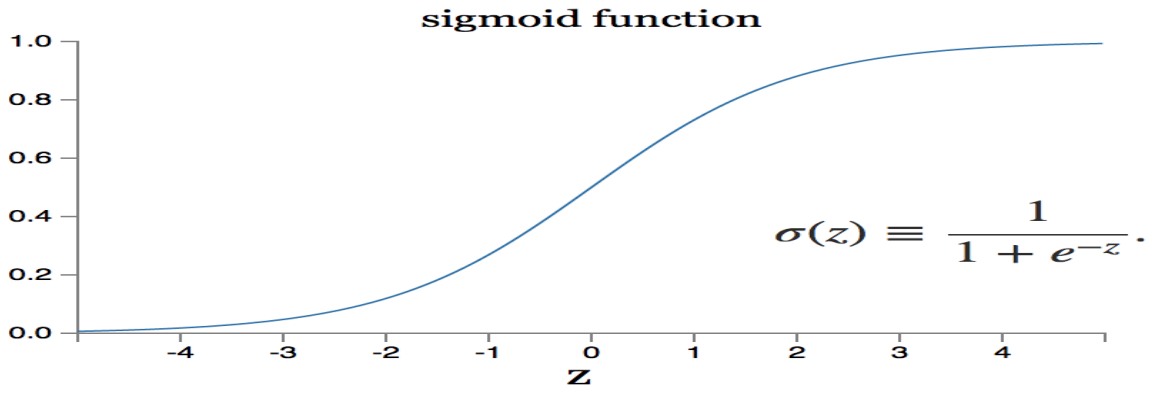
  Ce sont les fonctions les plus populaires de nos jours. Elles permettent un entrainement plus rapide comparé aux fonctions sigmoid et tanh, étant plus légères. Très utilisé pour les CNN, RBM, et les réseaux de multi perceptron. Intervalle de sortie (0;+∞).



  ✓ La fonction 'sigmoid' :

  Fonction la plus populaire depuis des décennies. Mais aujourd'hui, elle devient beaucoup moins efficace par rapport à d'autre pour une utilisation pour les couches cachées. Elle perd de l'information due à une saturation que cela soit pour la phase de feed forward ou de back propagation, en donnant des effets non linéaires au réseau due à un paramètre unique. Elle a aussi des soucis de gradient 0 avec des entrées étant très large, même si les soucis est

minimalisé avec les systèmes utilisant des batch par lots (mini batch). Utilisé en couche de sortie pour de la classification binaire. Intervalle de sortie : {0,1}



- Pour combler le manque de la fonction sigmoid, on a alterné les deux fonctions dans les couches.

```
### ici on met 5 layers pour diminuer la valeur de loss
### pour le nombre de neurons on change le change tant que les valeurs stagnent.
mod = keras.Sequential([
layers.Dense(64, activation = 'relu', input_shape = [xtrain.shape[1]]),
layers.Dropout(0.3, seed = 2),
layers.Dense(70, activation = 'sigmoid'),
layers.Dense(64, activation = 'relu'),
layers.Dense(64, activation = 'sigmoid'),
layers.Dense(64, activation = 'relu'),
layers.Dense(1)])
### pour la couche de sortie on met seulement un neuron
```

- Pour rendre notre modèle à une classification on met loss= ‘binary_crossentropy’ :

```
### notre optimiseur
optimiseur = tf.keras.optimizers.RMSprop(learning_rate = 0.001)
### dons notre optimiseur on calcule la précision (accuracy)
mod.compile(loss ='binary_crossentropy', optimizer = optimiseur,metrics = ['accuracy'])
### pour rendre notre modèle à une classification on met loss='binary_crossentropy'
```

- Cette alternation des fonctions nous donne une bonne accuracy :

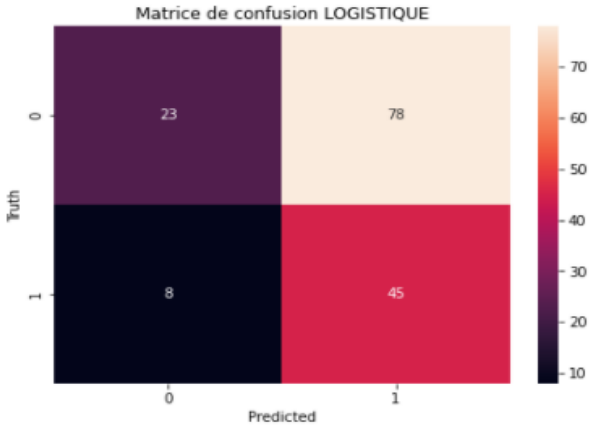
```
5/5 - 0s - loss: 0.5878 - accuracy: 0.7013
[0.5878371000289917, 0.701298713684082]
```

Résultats et discussions

Dans cette partie on va présenter et discuter les résulter des modèles de machine Learning (matrice de confusion + courbe de ROC) et on va les comparer avec le modèle de Deep Learning.

- ## Résultats :

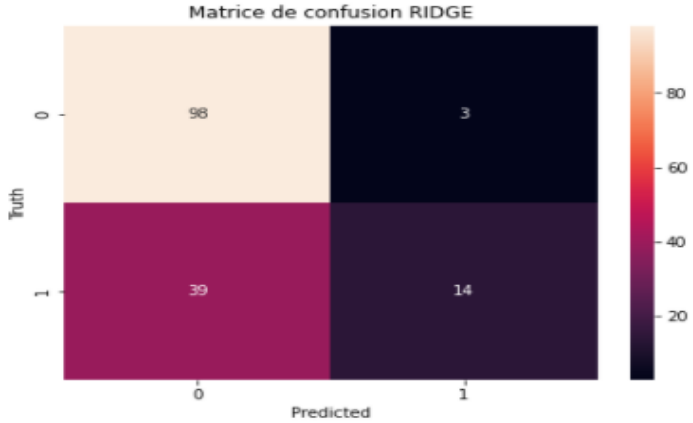
  - *Résultats de régression logistique :*
    - Ce modèle nous a donné un faible score : 0,4415
    - Et une moyenne d'accuracy à l'aide de Cross validation : 0,6465
    - Voici le plot de la matrice de confusion :



    - La traduction de cette matrice est :

TP (True Positives) (1,1) : 45 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
TN (True Negatives) (0,0) : 23 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
FP (False Positives) (0,1) : 78 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
FN (False Negatives) (1,0): 8 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.

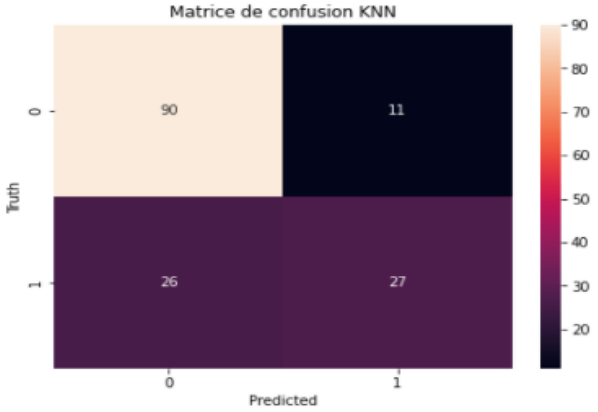
  - *Résultats de régression de Ridge :*
    - Ce modèle nous a donné un score : 0,7272
    - Et une moyenne d'accuracy à l'aide de Cross validation : 0,7116
    - Voici le plot de la matrice de confusion :



- La traduction de cette matrice est :

```
TP (True Positives) (1,1) : 14 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
TN (True Negatives) (0,0) : 98 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
FP (False Positives) (0,1) : 3 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
FN (False Negatives) (1,0): 39 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
```

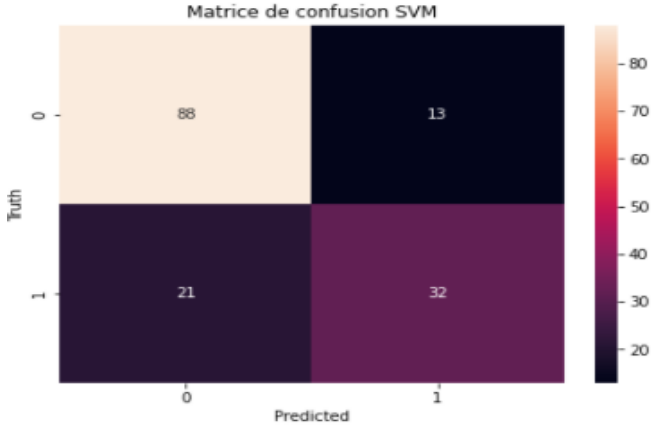
- *Résultats de modèle KNN :*
  - Ce modèle nous a donné un score : 0,7579
  - Et une moyenne d'accuracy à l'aide de Cross validation : 0,7328
  - Voici le plot de la matrice de confusion :



- La traduction de cette matrice est :

```
TP (True Positives) (1,1) : 27 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
TN (True Negatives) (0,0) : 90 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
FP (False Positives) (0,1) : 11 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
FN (False Negatives) (1,0): 26 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
```

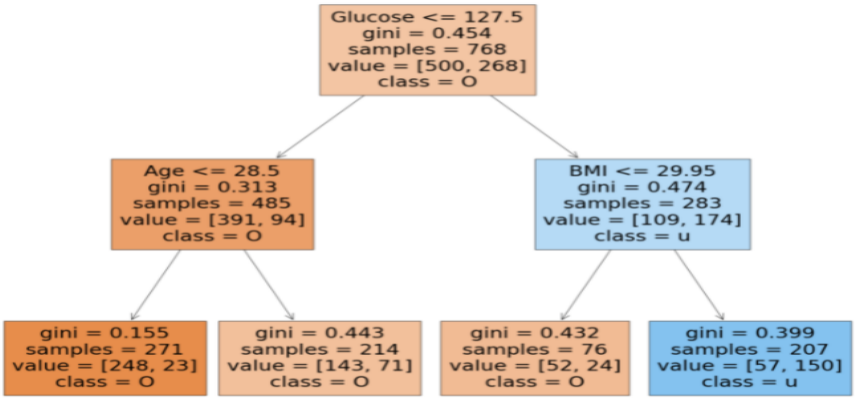
- *Résultats de modèle SVM :*
  - Ce modèle nous a donné un score : 0,78
  - Et une moyenne d'accuracy à l'aide de Cross validation : 0,76
  - Voici le plot de la matrice de confusion :



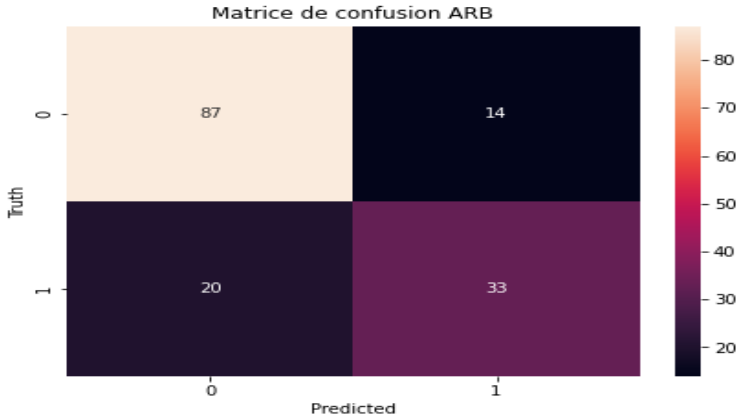
- La traduction de cette matrice est :

```
TP (True Positives) (1,1) : 32 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
TN (True Negatives) (0,0) : 88 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
FP (False Positives) (0,1) : 13 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
FN (False Negatives) (1,0): 21 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
```

- *Résultats de modèle Arbre de décision :*
  - Ce modèle nous a donné un score : 0,83
  - Et une moyenne d'accuracy à l'aide de Cross validation : 0,74
  - Voici le plot de l'arbre :



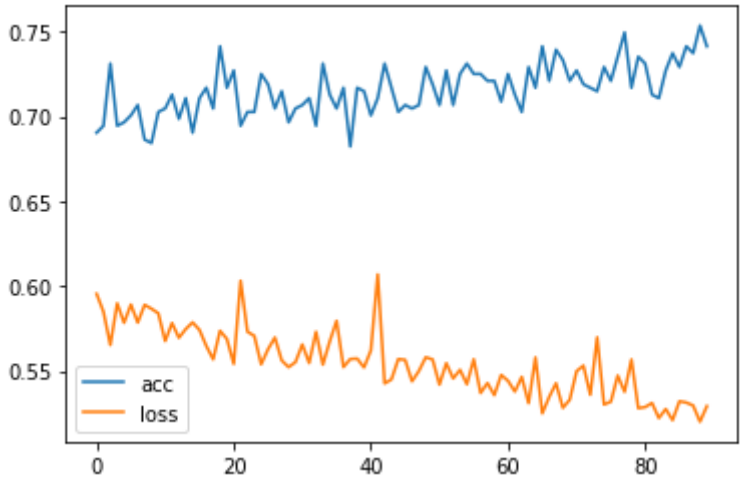
  - Voici le plot de la matrice de confusion :



  - La traduction de cette matrice est :

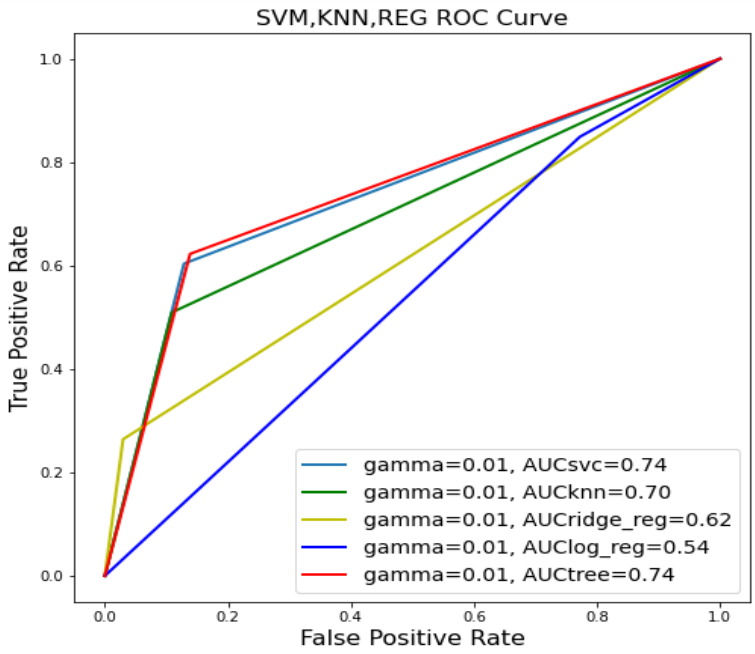
TP (True Positives) (1,1) : 33 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
TN (True Negatives) (0,0) : 87 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.
FP (False Positives) (0,1) : 14 des Femmes sans diabète et Notre prédiction de modèle a révélé qu'elles ont de diabète.
FN (False Negatives) (1,0): 20 des Femmes avec diabète et Notre prédiction de modèle a révélé qu'elles n'ont pas de diabète.

- *Résultats de modèle Réseau de neurones:*
  - Ce modèle nous a donné une accuracy : 0,70
  - Et une valeur de loss : 0.58
  - Voici le plot qui montre **le développement de la précision et de loss** :

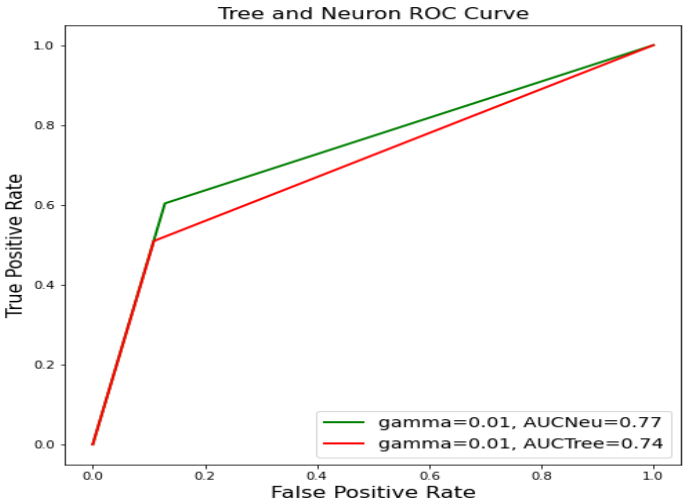


- **Discussions :**

Alors nous avons plusieurs machines modèles, mais on a besoin seulement du meilleur modèle, alors on a fait une comparaison entre les modèles avec la courbe du ROC :



- La courbe de roc nous montre que d'après les modèles de machine Learning le plus efficace est : le modèle d'arbre de décision.
- Alors dans la prochaine partie on va comparer le modèle arbre de décision avec le modèle réseau de neurones de Deep Learning.



- On observe qu'il n y a pas une grande différence entre les deux modèles alors on va utiliser les deux modèles dans notre interface graphique.

Outils et interface graphique

- ## Bibliothèques et Outils de développement :

Jupyter :

Project Jupyter est une organisation à but non lucratif crée pour "développer des logiciels open source, des standards ouverts et des services pour l'informatique interactive dans des dizaines de langages de programmation". Lancé d'IPython en 2014 par Fernando Pérez, Project Jupyter prend en charge les environnements d'exécution dans plusieurs dizaines de langues. Le nom du projet Jupyter est une référence aux trois principaux langages de programmation pris en charge par Jupyter, à savoir Julia, Python et R, ainsi qu'un hommage aux cahiers de Galileo enregistrant la découverte des lunes de Jupiter.

Matplotlib :

Matplotlib est une bibliothèque de traçage pour le langage de programmation Python et son extension de mathématiques numériques NumPy. Il fournit une API orientée objet pour incorporer des tracés dans des applications à l'aide de boîtes à outils GUI à usage général telles que Tkinter, wxPython, Qt ou GTK +. Il existe également une interface procédurale «pylab» basée sur une machine à états (comme OpenGL), conçue pour ressembler étroitement à celle de MATLAB, bien que son utilisation soit déconseillée. Scie utilise Matplotlib.

NumPy :

NumPy (NUM-pee)) est une bibliothèque pour le langage de programmation Python, ajoutant la prise en charge de grands tableaux et matrices multidimensionnels, ainsi qu'une grande collection de fonctions mathématiques de haut niveau pour opérer sur ces tableaux.L'ancêtre de NumPy, Numeric, a été créé à l'origine par Jim Hugunin avec les contributions de plusieurs autres développeurs. En 2005, Travis Oliphant a créé NumPy en incorporant des fonctionnalités du Numarray concurrent dans Numeric, avec des modifications importantes. NumPy est un logiciel open source et compte de nombreux contributeurs.

Keras :

Keras est une bibliothèque de logiciels open source qui fournissent une interface Python pour les réseaux de neurones artificiels. Keras agit comme une interface pour la bibliothèque TensorFlow.

Conçu pour permettre une expérimentation rapide avec des réseaux de neurones profonds, il se concentre sur être convivial, modulaire et extensible. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-end Neuro-Electronic Intelligent Robot Operating System), et

son principal auteur et mainteneur est François Chollet, un ingénieur de Google. Chollet est également l'auteur du modèle de réseau neuronal profond XCeption.

Scikit-learn :

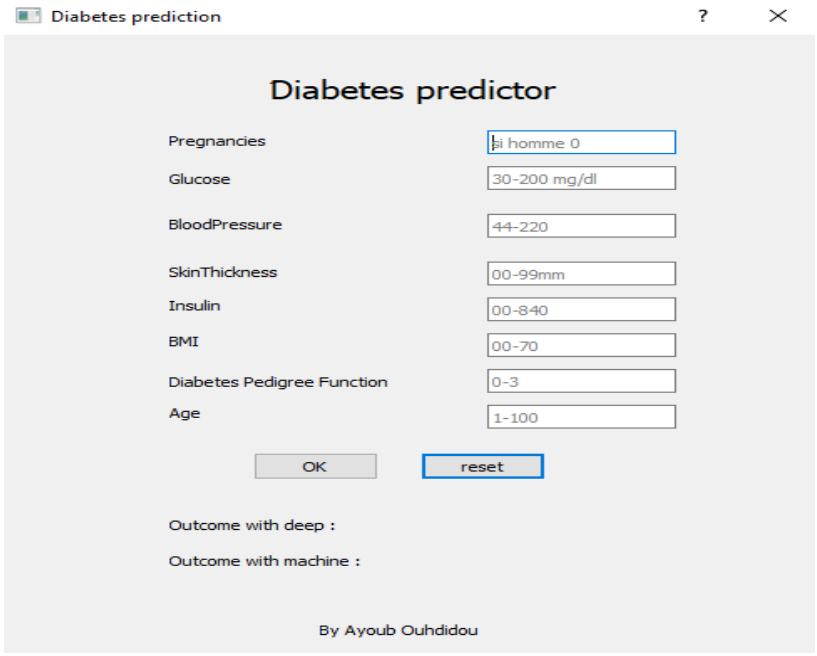
Scikit-learn (anciennement scikits.learn et également connu sous le nom de sklearn) est une bibliothèque d'apprentissage automatique de logiciels gratuits pour le langage de programmation Python.Il propose divers algorithmes de classification, de régression et de clustering, y compris les machines vectorielles de support, les forêts aléatoires, l'augmentation de gradient, k-means et DBSCAN, et est conçu pour interagir avec les bibliothèques numériques et scientifiques Python NumPy et SciPy.

- ## Interface graphique:

QT :

Qt (prononcé officiellement en anglais cute mais couramment prononcé Q.T. une API orientée objet est développée en C++, conjointement par The Qt Company et Qt Project. Qt offre des composants d'interface graphique (widgets), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, par certains aspects, elle ressemble à un framework lorsqu'on l'utilise pour concevoir des interfaces graphiques ou que l'on conçoit l'architecture de son application en utilisant les mécanismes des signaux et slots par exemple.

On a réalisé l'interface graphique à l'aide de la bibliothèque QT, dans l'interface graphique il y a des inputs pour donner les caractéristiques du personne et deux boutons (OK et reset) après que l'utilisateur clique le boutons OK, la sortie sera deux prédiction une avec machine Learning et l'autre avec Deep.



Conclusion générale

On peut conclure que la 2-ème méthode utilisant les réseaux neuronaux est la plus performante, même si elle a besoin de beaucoup de ressources et de temps, mais les résultats sont vraiment très bons.

Par contre la première méthode utilisant les modèles de machine Learning (Arbre de décision), n'est pas très précis, cela peut créer des vrais soucis lors du déploiement du modèle, car une telle marge d'erreur n'est pas tolérée sur terrain.

Alors vaut mieux consommer plus de temps et de ressource sur l'entrainement (le cas de la deuxième méthode) et avoir de bon résultat, que d'entrainer rapidement un modèle qui n'est pas très fiable.

Finalement, les résultats de ce projet ont nourri ma curiosité, pour mieux connaitre le deep learning. Car j'en suis sûre que nos sociétés vont y dépendre prochainement, et grâce à lui l'humanité va accomplir d'incroyables créations et prodigieux progrès scientifiques.

Ouhdidou Ayoub.

Bibliographie

- https://openclassrooms.com
- https://scikit-learn.org
- https://www.stackoverflow.com
- https://Wikipedia.org
- Tps de Monsieur B.Ayoub.