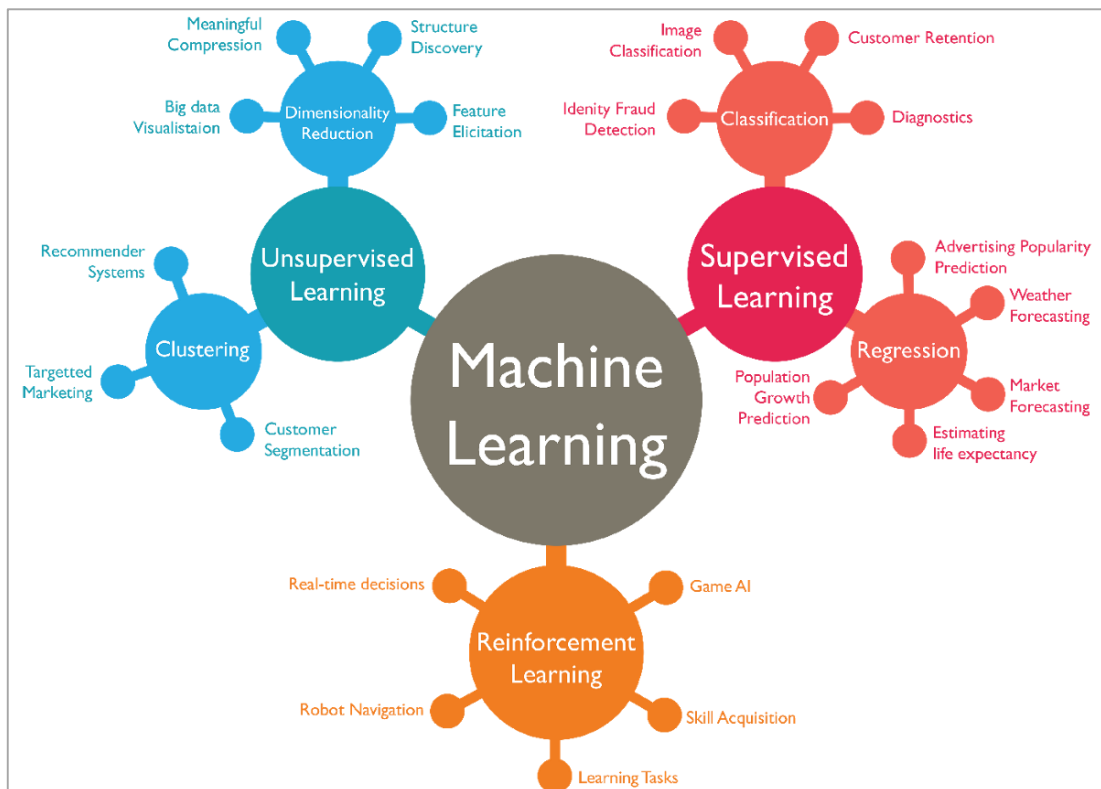


## 1<sup>ère</sup> Année Master Informatique, Semestre 2

### Module : TER

#### *Étude d'un algorithme de Machine Learning*



#### Rapport réalisé par :

SARAH OUHOCINE (AMIS)

#### Sujet proposé et encadré par :

M. LUC BOUGANIM

M. LUDOVIC JAVET

# SOMMAIRE

<b>INTRODUCTION GENERALE .....</b>	<b>1</b>
<b>ETAPE 1 .....</b>	<b>1</b>
1. Liens vers les pages qui décrivent ALGO .....	
2. Motivation personnelle pour ALGO .....	
3. Scénarios pertinents pour ALGO .....	
<b>ETAPE 2 (<i>Supervisé avec paramètres par défaut</i>) .....</b>	<b>2</b>
1. Datasets Choisis .....	
2. Justification des choix des datasets .....	
3. Implémentation d'un programme de test de base (paramètres par défauts) .....	
3.1 Classification .....	
3.1.1 Lien vers le code .....	
3.1.2 Métriques choisies et résultats obtenus .....	
3.2 Régression .....	
3.2.1 Lien vers le code .....	
3.2.2 Métriques choisies et résultats obtenus .....	
4. Observations et conclusions .....	
<b>ETAPE 3 (<i>Supervisé avec paramètres optimaux</i>) .....</b>	<b>5</b>
1. Description et explications des paramètres de MLP .....	
2. Discussion des variations .....	
3. Classification .....	
3.1 Lien vers le code .....	
3.2 Résultats obtenus .....	
4. Régression .....	
4.1 Lien vers le code .....	
4.2 Résultats obtenus .....	
5. Observations et conclusions .....	
<b>ETAPE 4 .....</b>	<b>6</b>
1. Classification .....	
1.1 Lien vers le dataset .....	
1.2 Lien vers l'analyse du dataset .....	
2. Régression .....	
2.1 Lien vers le dataset .....	
2.2 Lien vers l'analyse du dataset .....	
3. Justification des choix des datasets .....	
<b>ETAPE 5 (<i>Supervisé avec paramètres par défauts et optimaux</i>) .....</b>	<b>8</b>
1. Liens vers les codes .....	
2. Discussion des résultats obtenus, observations et conclusions .....	
<b>ETAPE 6 (<i>Semi-Supervisé</i>) .....</b>	<b>9</b>
1. Lien vers les codes .....	
2. Discussion des résultats obtenus .....	
3. Comparaison des résultats du semi-supervisé et supervisé .....	
4. Observations et conclusions .....	
<b>CONCLUSION GENERALE .....</b>	<b>10</b>

## INTRODUCTION GENERALE

Il existe plusieurs algorithmes de Machine Learning, selon le mode d'apprentissage qu'ils emploient (supervisé, non supervisé, semi-supervisé, par renforcement, par transfert etc.)

Dans le cadre du projet TER, nous avons opté pour l'algorithme « **les Réseaux de Neurones Artificiels** », (ci-après nommé **ALGO**), qui sont en fait, des systèmes informatiques inspirés des réseaux de neurones biologiques (Circuits neuronaux) qui constituent le cerveau des animaux.

### *Étape 1*

#### 1- Liens vers les pages qui décrivent **ALGO**



**Scikit-learn :**

- [Modèles de réseaux de neurones \(supervisés\).](#)
- [Modèles de réseaux de neurones \(non supervisés\).](#)



**Wikipédia :**

- [Réseaux de neurones artificiels](#)

#### 2- Motivation personnelle pour **ALGO**

Notre choix s'est porté sur **ALGO** pour les raisons suivantes :

- ✓ **ALGO** respecte rigoureusement les contraintes posées sur l'algorithme à choisir (feuille de route). En effet, **ALGO** est un algorithme d'apprentissage supervisé applicable aux problèmes de régression et de classification, et adapté à l'apprentissage semi-supervisé.
- ✓ **ALGO** est un algorithme indépendant des données qui nous donne la liberté de choisir n'importe quel ensemble de données.
- ✓ Les réseaux de neurones sont très populaires, grâce à leur performance sur les datasets volumineux.

#### 3- Scénarios pertinents pour **ALGO** :

Les cas d'usages des réseaux de neurones sont nombreux. Parmi les exemples d'usages, on peut citer :

- ✓ La classification de textes et d'images.
- ✓ La reconnaissance d'images, exemple : identifier et délimiter des tumeurs dans des images médicales.
- ✓ La reconnaissance d'écriture manuscrite.
- ✓ La reconnaissance faciale.
- ✓ La prévision des marchés financiers, aussi appelée « trading algorithmique »
- ✓ Entraînement des chatbots et des algorithmes de traitement naturel du langage (NLP).
- ✓ La prédiction météo.
- ✓ L'analyse prédictive en entreprises.

### *Étape 2 (Supervisé avec paramètres par défaut)*

#### 1- Datasets Choisis

scikit-learn présente plusieurs datasets adaptés à la régression et à la classification.

Nous avons choisi les 2 datasets suivants :

✓ **Pour le problème de régression** : nous avons choisi de tester notre ALGO sur le dataset « *Diabetes* ».

Il s'agit des données relatives à 442 patients diabétiques, tels qu'un patient est caractérisé par 11 attributs :

- 10 attributs (features / inputs) : qui sont des valeurs prédictives numériques / réelles (âge, sexe, indice de masse corporelle, pression artérielle moyenne, et six mesures différentes de sérum sanguin).
- 1 attribut (target / output) : qui est une valeur entière entre 25 et 346, représentant une mesure quantitative de la progression de la maladie un an après la ligne de base.

✓ **Pour le problème de Classification** : Nous avons choisi le dataset « *Breast cancer* ».

Il s'agit des données relatives à 569 images médicales de tumeurs / cancer, tels que chaque tumeur est caractérisée par 31 attributs :

- 30 attributs (features / inputs) qui sont des valeurs numériques prédictives / réelles / positives (rayon, texture, périmètre, zone, douceur, compacité, concavité, points concaves, symétrie, dimension fractale, etc...).
- 1 attribut (target / output) qui est une valeur discrète représentant la classe dont la tumeur appartient (**B**énigne ou **M**aligne).

## 2- Justification des choix des datasets :

- ✓ Les données des deux datasets ne nécessitent pas une étape de nettoyage (aucune valeur d'attribut manquante), ce qui va nous permettre d'accélérer (ou de sauter) l'étape de prétraitement des données.
- ✓ Les deux datasets sont d'une bonne taille (pas trop grands). De ce fait, ils sont utiles pour illustrer rapidement le comportement des différents algorithmes implémentés dans scikit-learn et ainsi, obtenir rapidement les résultats.
- ✓ Les deux datasets sont parfaits et optimisés pour scikit-learn. De ce fait, même les modèles de base donnent de bons résultats.
- ✓ Le thème des données des deux datasets sont approximativement proches « domaine médical », qui est un domaine passionnant, très intéressant et donne de plus en plus des résultats brillants avec l'évolution de l'Intelligence Artificielle et notamment, de la Machine Learning.
- ✓ Nous voulons travailler sur un même thème pour la régression ainsi que pour la classification. Cela va donner un sens à notre travail, mais aussi, va nous faciliter le choix des métriques et l'interprétation des résultats.  
Nous sommes curieux de savoir s'il y a des (faibles / moyennes / fortes) corrélations entre le diabète et le cancer du sein.

## 3- Implémentation d'un programme de test de base (paramètres par défaut) :

### 3.1 Classification :

Notre programme de test est basé sur les réseaux de neurones artificiels de type Perceptron Multicouchee (MLP). En effet, nous avons utilisé la fonction `MLPClassifier` définie dans le module `sklearn.neural_network`, qui fournit une implémentation du Perceptron Multicouche pour la classification.

`MLPClassifier` comprend 23 paramètres à ajuster tels que le nombre de neurones pour chaque couche cachée (`hidden_layer_sizes`), la fonction d'activation (`activation`) et l'algorithme d'optimisation (`solver`) ...etc.

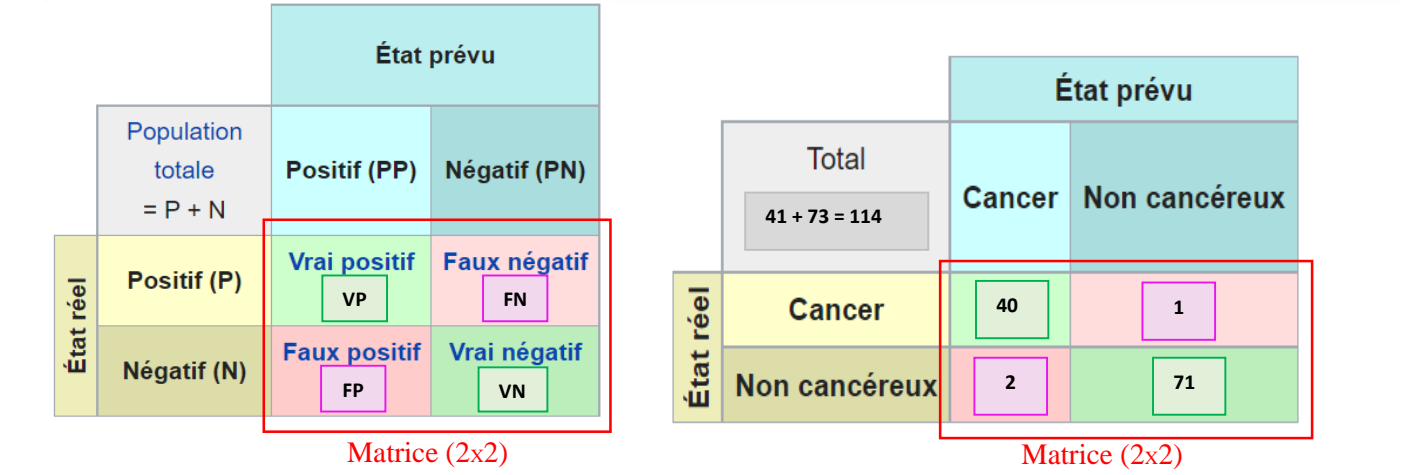
Notre Implémentation est une implémentation **de base**, i.e. les 23 paramètres ont été pris avec leur valeur par défaut dans `scikitlearn` (**paramètres par défaut**).

### 3.1.1- Lien vers le code : [GitHub](#)

### 3.1.2- Métriques Choiesies et résultats obtenus :

Nous avons choisi 5 métriques adaptées aux problèmes de classification, mais nous allons présenter les 2 métriques les plus pertinentes :

#### ✓ Matrice de confusion



- Sur les 42 échantillons avec cancer (40 + 2), le système a prédit que 2 étaient sans cancer.
- Sur les 72 échantillons sans cancer (2 + 71), il a prédit que 1 avait un cancer.
- Toutes les **prédictions correctes** sont situées dans la diagonale du tableau (surlignées en vert) → 40 + 71 = 111.
- Toutes les **prédictions incorrectes** sont situées dans la diagonale du tableau (surlignées en rouges) → 1 + 2 = 3. Il est donc facile d'inspecter visuellement le tableau pour les erreurs de prédiction, car les valeurs en dehors de la diagonale les représenteront. En additionnant les 2 lignes de la matrice de confusion,
- Le nombre total d'échantillons positifs (P) est donné par :  $P = VP + FN = 40 + 1 = 41$ .
- Le nombre total d'échantillons négatifs (N) est donné par :  $N = FP + VN = 2 + 71 = 73$ .
- Le nombre total d'échantillons est donné par :  $P + N = 41 + 73 = 114$ .

#### ✓ Accuracy ([accuracy\\_score](#))

Ce paramètre fait la somme de tous les vrais positifs et vrais négatifs qu'il divise par le nombre total d'instances. Il permet d'apporter une réponse à la question suivante : de toutes les classes positives et négatives, combien parmi elles ont été prédites correctement ?

VP + VN

VP + FN + FP + VN

Plus il est élevé, mieux c'est. La meilleure accuracy possible est de 1. **Notre modèle a atteint une accuracy assez élevée (0.974), on constate que notre modèle de base est bon et performant.**


En résumé, les paramètres par défaut nous ont permis de construire un bon modèle et d'obtenir de bons résultats malgré le peu de données que nous avons (petit dataset). Notre interprétation est que, les paramètres par défaut construisent de bons modèles et donnent de bons résultats lorsque nous utilisons les jeux de données de scikitlearn pour les problèmes de classification.

### 3.2 Régression :

Notre programme de test est basé sur les réseaux de neurones artificiels de type Perceptron Multicouchee (MLP). En effet, nous avons utilisé la fonction MLPRegressor définie dans le module sklearn.neural\_network, qui fournit une implémentation du Perceptron Multicouche pour la régression.

MLPRegressor comprend 23 paramètres à ajuster tels que le nombre de neurones pour chaque couche cachée (hidden\_layer\_sizes), la fonction d'activation (activation) et l'algorithme d'optimisation (solver) ...etc.

Notre Implémentation est une implémentation **de base**, i.e. les 23 paramètres ont été pris avec leur valeur par défaut dans scikitlearn (**paramètres par défaut**).

3.2.1- Lien vers le code : [GitHub](#) 

#### 3.2.2- Métriques Choisies et résultats obtenus :

Nous avons choisi 3 métriques adaptées aux problèmes de régression, mais nous allons présenter la plus métrique la plus pertinente :

##### ✓ Coefficient de Détermination ([r2\\_score](#))

Le coefficient de détermination est le carré de la corrélation de Pearson (R) entre les valeurs prédites et vraies valeurs. Il se calcule en complétant la RSE à 1

$$R^2 = 1 - RSE$$
$$RSE = \frac{\sum_{i=1}^n (y_i - f(x_i))^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$
$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

Le coefficient de détermination est un indicateur qui mesure la qualité d'une prédiction. Plus cet indicateur est proche de 1, plus le modèle est proche de la réalité et les valeurs prédites sont corrélées aux vraies valeurs. Le meilleur coefficient de détermination possible est de 1, mais il est rarement atteint par un modèle dans la pratique.

Il peut être **négatif** (car le modèle peut être arbitrairement pire, lorsqu'il est très loin de la réalité).

**Notre modèle a un coefficient de détermination ( $R^2$ ) négatif = -0,80. En effet, notre modèle de base est très loin du modèle réel, et les valeurs prédites ne sont pas corrélées aux vraies valeurs. Nous constatons que notre modèle de base n'est pas bon et n'est pas performant.**

Nous pouvons dire que notre modèle de régression de base (avec les paramètres par défauts) ne s'insère pas bien dans le nuage de point pour les différents attributs du dataset. Ce qui veut dire qu'il y a de grandes marges d'erreurs entre les prédictions  $f(X_i)$  et les vraies valeurs ( $Y_i$ ) du dataset. De ce fait, nous constatons que notre modèle est très loin de la réalité (loin des vraies valeurs et représente mal les données).

#### 4. Observations et résultats

En résumé, contrairement à la classification. Dans la régression, les paramètres par défaut ne nous ont pas permis de construire un bon modèle et d'obtenir de bons résultats malgré que le jeu de données utilisé est un jeu de données de Sickitlearn (Diabetes). Notre interprétation est que, la régression nécessite beaucoup plus de données pour avoir un bon modèle et obtenir de bons résultats. Pour cette raison-là, que nous allons nous focaliser dans les étapes qui suivent sur d'autres jeux de données (plus grands), mais aussi, sur l'ajustement des paramètres des deux modèles pour essayer de les améliorer, notamment celui de la régression.

### *Étape 3 (Supervisé avec paramètres optimaux)*

#### 1- Description et explication des paramètres de MLP

Nous allons décrire uniquement 4 paramètres sur 23 que nous avons jugé comme pertinents : activation, solver, learning\_rate, max\_iter).

La pertinence a été définie en visualisant les effets de variation de chaque paramètre de MLP indépendamment des paramètres restants sur la fonction de perte (erreur) tels que plus le résultat de cette fonction est minimisé, plus le réseau de neurones est performant.

#### [VISUALISATION DES GRAPHES](#)

Remarque : Nous avons gardé que les graphes où nous avons une remarquable différence sur la courbe de la fonction d'erreur (et donc les graphes correspondants aux paramètres pertinents). Et cela pour chacun des deux datasets : de classification (*breast\_cancer*) et de la régression (*diabetes*).

- ✓ **Activation** : La fonction d'activation est spécifique à chaque couche cachée. Elle sert à introduire une complexité non-linéaire au modèle. elle modifie de manière **non-linéaire** les données. Cette non-linéarité permet de modifier leur représentation.  
Un modèle étant composé de multiples couches cachées, et donc de multiples fonctions d'activation ; rend la représentation des données complexe et changeable successivement. Ce qui nous permet d'avoir **un nouveau point de vue sur nos données**.
- ✓ **Solveur** : ou l'optimiseur, il joue le rôle de corriger / optimiser l'algorithme (corriger les poids d'un perceptron), pour que la fois suivante, il prédise une valeur plus proche de la valeur réelle. Autrement dit, le solveur diminue / optimise la fonction d'erreur (loss function) ; dans le but d'améliorer les prédictions et la stabilité du modèle construit auparavant.
- ✓ **Learning\_rate** : Il représente la vitesse à laquelle un modèle d'apprentissage automatique "apprend", i.e il indique la vitesse à laquelle les poids de neurones évoluent (mis à jour).
- ✓ **Max\_iter** : C'est le nombre maximal d'itérations de l'algorithme. En effet, le solveur itère jusqu'à

La convergence ou ce nombre d'itérations défini.

## 2- Discussion des variations

Il existe maintenant de nombreux algorithmes d'optimisation et de réglage des paramètres, nous avons choisi la méthode GridSearchCV qui trouve facilement les valeurs des paramètres optimaux parmi les valeurs données en faisant une recherche exhaustive sur la grille. Nous avons donc exploité cette méthode pour faire un programme automatique variant différentes valeurs pour les 4 paramètres d'ALGO (notre espace de recherche). Et donner ainsi, la meilleure combinaison en fonction du score de prédiction et observer les effets (ou les non-effets) de la modification de ces paramètres.

Nous avons choisi cette méthode, car elle est très répandue, efficace, et donne des résultats rapidement.


## 3- Classification

3.1 - Lien vers le code : [GitHub](#) 

3.2 Résultats obtenus :

Le nouveau modèle déterminé par GridSearchCV Nous a permis d'améliorer le modèle de « classification avec paramètres par défaut » de :  $1 - 0,974 = 0.026$

## 4- Régression

4.1 - Lien vers le code : [GitHub](#) 

4.2 Résultats obtenus :

Le nouveau modèle déterminé par GridSearchCV Nous a permis d'améliorer le modèle de « régression avec paramètres par défaut » de :  $0.61 - (-0,80) = 1.41$

## 5- Observations et conclusions

Nous avons constaté que l'implémentation d'un algorithme de **Deep Learning** (augmentation du nombre de couches cachées et des neurones dans chaque couche cachée) dans notre cas n'améliore pas les 2 modèles, au contraire il les empire. Selon nous, cela est dû à la taille des datasets (petits datasets).

Les modèles de régression restent plus difficiles à améliorer par rapport aux modèles de classification. Et cela car le nombre des classes à prédire n'est pas déterminé, étant donné que les valeurs sont continues

Les deux modèles Ci-dessus sont les deux meilleurs modèles obtenus jusqu'à maintenant.

## *Étape 4 (Supervisé)*

### 1. Classification :

1.1- Lien vers le dataset : [Dataset de la classification](#)

1.2- Lien vers l'analyse du dataset : [Analyse du dataset de la classification](#)

Nous avons décidé de travailler sur le dataset proposé par le site de [« SuperAnnotate »](#). Le dataset choisi est : [« The Fetal Heath Classification »](#). Il est créé le 07 septembre 2010, et il vise à classer la santé d'un fœtus comme normale, suspecte ou pathologique à partir des données d'exams de cardiocotogrammes (CTG).



Il s'agit des données relatives à 2126 examens de CTG, tels que chaque examen est caractérisée par 22 attributs

- 21 attributs (features / inputs), citons les 3 premiers :
  - ✓ **'Baseline value'** Valeurs de fréquence cardiaque fœtale de base.
  - ✓ **'accelerations'** Nombre d'accélération par seconde.
  - ✓ **'fetal\_movement'** Nombre de mouvements du fœtus par seconde.
- 1 attribut (target / output) :
  - ✓ **'fetal\_health'** : 1 (Normal), 2 (Suspect) et 3 (Pathologique).

D'après l'analyse du dataset, tous les attributs sont pertinents (i.e. les valeurs de tous les attributs sont significatives), pour la classification ce qui fait dans l'étape 5 on va prendre en compte tous les attributs du dataset pour faire l'apprentissage.

## 2. Régression :

2.1- Lien vers le dataset : [Dataset de la regression](#)

2.2- Lien vers l'analyse du dataset : [Analyse du dataset de la régression](#)

Nous avons décidé de travailler sur le dataset proposé par le site de [« kaggle »](#). Le dataset choisi est : [« Medical Cost Personal »](#) qui consiste à prédire le coût de l'assurance maladie via des données collectés par des experts.

Il s'agit des données relatives à 1338 patients, tels que chaque patient est caractérisée par 7 attributs (colonnes)

- 6 attributs (features / inputs), citons les 3 premiers:
  - ✓ **Âge** : âge du principal bénéficiaire
  - ✓ **Sexe** : sexe de l'assureur, féminin, masculin
  - ✓ **BMI** : indice de masse corporelle, permettant de comprendre le corps, les poids relativement élevés ou faibles par rapport à la taille, indice objectif de poids corporel ( $\text{kg/m}^2$ ) utilisant le rapport taille sur poids, idéalement 18,5 à 24,9.
- 1 attribut (target / output) :
  - ✓ **Charges** : Frais médicaux individuels facturés par l'assurance maladie.

D'après l'analyse du dataset, ce n'est pas tous les attributs qui sont pertinents (i.e. les valeurs de certains attributs tels que sex, smoker, region ne sont pas significatives), ce qui fait dans l'étape 5 on ne va pas prendre en compte ces 3 attributs cités du dataset pour ne pas fausser le résultat de l'apprentissage.

## 3- Justification des choix des datasets :

- ✓ Nous voulions rester dans le même thème médical que celui des datasets précédents (étape 2 et 3).
- ✓ Les datasets traitent des sujets intéressants.
- ✓ La taille du dataset est importante : le nombre d'instances est suffisant pour faire un bon apprentissage, que ce soit pour la classification (2126 instances) ou la régression (1338 instances).
- ✓ Les données des datasets ne nécessitent pas une étape de nettoyage (aucune valeur d'attribut manquante d'après l'analyse).

### *Étape 5 (Supervisé avec paramètres par défauts et optimaux)*

Dans cette étape nous avons refait les :

- ✓ **Étapes 2** : en faisant l'apprentissage avec les paramètres par défaut de Sickitlearn sur les deux datasets décrits dans l'étape 4.
- ✓ **Étape 3** : en faisant l'apprentissage avec les paramètres optimaux (meilleur modèle trouvé grâce à GridSearchCV de Sickitlearn) sur les deux datasets décrits dans l'étape 4.

Dans cette étape :

Nous avons exploité le TPU de « google colab » pour faire l'apprentissage (notamment pour trouver les paramètres optimaux), vu la taille des deux datasets décrits dans l'étape 4.

## 1. Liens vers le code

Pour chaque type de problème (Classification/ Régression), nous avons rassemblé l'apprentissage avec les paramètres par défaut et celui avec les paramètres optimaux dans un seul code.

- ✓ **1.1. Classification** : [GitHub](#) 
- ✓ **1.2. Régression** : [GitHub](#) 

## 2. Discussion des Résultats Obtenus, Observations et Conclusions :

CLASSIFCATION				RÉGRESSION				
Étape 2/4		Étape 3/5		Étape 2/4		Étape 3/5		
« paramètres par défaut »		« paramètres optimaux »		« paramètres par défaut »		« paramètres optimaux »		
Dataset de Sickit Learn	Dataset de Super Annonate	Dataset de Sickit Learn	Dataset de Super Annonate	Dataset de Sickit Learn	Dataset de Kaggle	Dataset de Sickit Learn	Dataset de Kaggle	
«breast_cancer»	«featl_health»	«breast_cancer»	«featl_health»	«diabetes »	«insurance»	«diabetes »	«insurance»	
Score de prédiction	0.97	0.87	1	0.9	-0.80	-1.10	0.61	0.12
	569	2126	569	2126	442	1338	442	1338
Taille du dataset								

- ✓ La classification donne toujours des résultats beaucoup mieux que la régression quel que soit le dataset utilisé Sickit-Learn ou autre (scores : 0.97, 0.87, 1, 0.9 pour la classification) contre (scores : -0.80, -1.10, 0.61, 0.12 pour la régression).

Nous concluons que la régression nécessite beaucoup plus de données (données massives) pour avoir un bon modèle étant donné que le nombre de classes à prédire dans la classification est déterminé alors que dans la régression est indéterminé.

La régression donne souvent un score négatif lorsque le modèle est très loin du modèle réel.

- ✓ La classification et la régression sur les datasets de Scikit-Learn donne de bons résultats par rapport aux autres datasets hors Sickit-Learn, malgré le peu d'instances qu'ils contiennent comparé aux nombre d'instances de ces derniers (569 instances contre 2126 pour la classification) et (442 instances contre 1338 pour la régression). Et cela que ce soit l'apprentissage fait avec paramètres par défaut ou avec paramètres optimaux.

Nous concluons que quel que soit le dataset de Sickit-Learn utilisé (taille, thème, type de problème : Classification ou Régression). Ce dernier sera parfait et optimisé pour Sickit-Learn. Du genre que, même les modèles de base construits avec les paramètres par défaut donnent de très bons résultats.

- ✓ L'apprentissage avec les paramètres optimaux (utilisation de GridSearchCV) donnent de meilleurs résultats par rapport à l'apprentissage avec les paramètres par défaut. Et cela quel que soit le type de problème (classification ou régression) et la source du dataset (Sickit-Learn ou autre) : (scores : 1 contre 0.97, 0.9 contre 0.87, 0.61 contre -0.80, 0.12 contre -1.10).

Nous concluons, qu'il est important d'utiliser une méthode qui calibre les paramètres d'une façon automatique au lieu de les calibrer manuellement pour obtenir de meilleurs résultats. En procédant ainsi, nous serons sûrs que nous ressortirons tous les modèles qui puissent exister (d'après les valeurs introduites pour chaque paramètre) et ainsi, nous sélectionnerons le meilleur modèle.

Le seul inconvénient de cette méthode est que plus la taille du dataset augmente, plus elle consomme de mémoire, mais de nombreuses solutions sont présentes telles que faire l'apprentissage avec *google colab* qui fournit à l'utilisateur des GPU et TPU gratuitement (cette solution reste meilleure par rapport aux méthodes de calibrage manuelles).

## *Étape 6 (Semi-Supervisé)*

C'est une technique d'apprentissage automatique qui consiste à apprendre une fonction de prédiction à partir d'un ensemble de données étiquetées (annotées) et non étiquetées. Ainsi, il se situe entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non supervisé qui n'utilise que des données non étiquetées.

### 1- Lien vers les codes

Nous avons découpé les 2 datasets de classification en LAB, non-LAB (de 10% à 80%) et TEST (10 %). Puis nous avons entraîné le modèle en supervisé sur LAB. ensuite, nous avons amélioré le modèle en semi-supervisé sur non-LAB et enfin, nous avons testé le modèle sur TEST.

- ✓ [breast\\_cancer de Sk-learn](#) 
- ✓ [fetal\\_health de Kaggle](#) 

### 2- Discussion des Résultats Obtenus :

Pour bien observer l'effet de la technique semi supervisé sur la qualité de l'apprentissage, nous avons varié la proportion LAB/non-LAB et recalculé la métrique de qualité (score de prédiction) sur TEST.

Remarque : Les graphes sont à la fin des codes, ainsi que dans [le fichier Excel](#).

Pour les deux datasets de classification, nous remarquons que plus la quantité des données non étiquetées diminue (en // la quantité des données étiquetées augmente), plus le score de prédiction diminue (i.e la qualité de l'apprentissage diminue).

### 3- Comparaison des résultats du semi-supervisé et supervisé

Nous avons défini pour le semi-supervisé un modèle de base (avec paramètres par défaut), ce qui fait, nous avons comparé les résultats avec les résultats de l'étape 2 du supervisé (modèle de base) car il a donné déjà de très bons résultats.

		Classification			
		Supervisé		Semi-supervisé	
		« paramètres par défaut »		« paramètres par défaut »	
Score de prédiction	Taille du dataset	Dataset de Sickit Learn	Dataset de Super Annonate	Dataset de Sickit Learn	Dataset de Super Annonate
		«breast_canc er»	«featl_health »	«breast_canc er»	«featl_health »
		0.97	0.87	0.989	0.88
		569	2126	569	2126

- ✓ La classification dans le semi supervisé donne des résultats mieux que le supervisé quel que soit le dataset utilisé Sickit-Learn ou autre et quel que soit sa taille (score : 0.989 contre 0.97 pour le dataset breast\_cancer de sk-learn) et (score : 0.88 contre 0.87 pour le dataset fetal-health de kaggle).

Nous concluons que la classification en semi-supervisé est plus efficace qu'en supervisé.

#### 4- Observations et conclusions

Les données non étiquetées, en combinaison avec **peu** de données étiquetées, permettent de produire une amélioration considérable de la précision de l'apprentissage. Ainsi, ceci améliore significativement la qualité de l'apprentissage.

Pour un problème d'apprentissage donné, l'étiquetage de données nécessite souvent l'intervention d'un utilisateur humain. En revanche, les estimateurs semi-supervisés **sklearn.semi\_supervised** (grâce à la fonction predict\_proba pour les RN) sont capables d'utiliser ces données supplémentaires non étiquetées pour mieux saisir la forme de la distribution des données sous-jacentes et mieux les généraliser à de nouveaux échantillons.

Ces algorithmes peuvent bien fonctionner lorsque nous avons une très petite quantité de données étiquetées et une grande quantité de points non étiquetés, exemple de (LAB 10% non-LAB 80%).

#### CONCLUSION GENERALE

Pour conclure, nous témoignons que ce projet de TER nous a été d'une très grande aide pour améliorer collectivement nos connaissances en apprentissage automatique supervisé et semi-supervisé pour les problèmes de classification et de régression, notamment en réseaux de neurones (un domaine qui nous intéresse beaucoup).

Nous tenons ainsi, à exprimer notre gratitude à nos deux encadreurs, M. Ludovic JAVET et M. Luc BOUGANIM pour leur encadrement, leur patience et leurs constantes orientations tout au long de ce projet. En y accordant une méticuleuse attention, pour leurs judicieux conseils et leurs précieuses explications qui nous ont été indispensables à la conduite de ce projet. Nous les remercions ainsi pour leur extrême amabilité et leur totale disponibilité.

