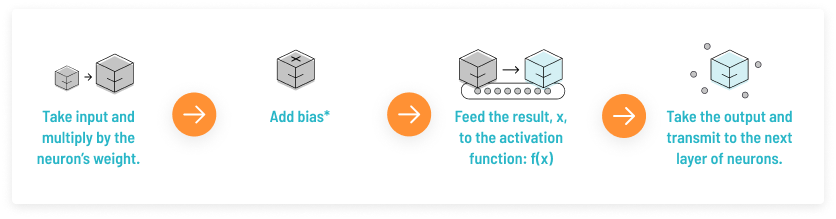
# Introduction

**Deep Learning** fait partie d'une famille plus large de méthodes d'apprentissage automatique basées sur des réseaux de neurones artificiels avec apprentissage par représentation.

L'apprentissage peut être supervisé, semi-supervisé ou non supervisé. Deep-learning architectures tels que les réseaux de neurones profonds, les réseaux de croyances profondes, les réseaux de neurones récurrents et les réseaux de neurones convolutifs ont été appliqués à des domaines tels que la vision par ordinateur, la vision industrielle, la reconnaissance vocale, le traitement du langage naturel, la reconnaissance audio, le filtrage des réseaux sociaux, la traduction automatique, la bioinformatique, la conception de médicaments , des programmes d'analyse d'images médicales, d'inspection de matériaux et de jeux de société, où ils ont produit des résultats comparables et, dans certains cas, surpassant les performances des experts humains.

Or, les réseaux de neurones artificiels (ANN) sont constitués d'un grand nombre d'éléments simples, appelés neurones, dont chacun prend des décisions simples. Ensemble, les neurones peuvent fournir des réponses précises à certains problèmes complexes, tels que le traitement du langage naturel, la vision par ordinateur et l'IA.

Un réseau de neurones peut être «superficiel», ce qui signifie qu'il a une couche d'entrée de neurones, une seule «couche cachée» qui traite les entrées et une couche de sortie qui fournit la sortie finale du modèle. Un réseau neuronal profond (DNN) a généralement entre 2 à 8 couches supplémentaires de neurones. Les recherches de Goodfellow, Bengio et Courville et d'autres experts suggèrent que les réseaux de neurones augmentent en précision avec le nombre de couches cachées.



# Problématique :

L'objectif de ce projet est de construire un modèle d'apprentissage automatique basé sur les réseaux du neurone artificielle ANN, pour la détection de breast cancer en utilisant Wisconsin Diagnostic Breast Cancer (WDBC) dataset, et la Bibliothèque TonsorFlow.

# Déroulement Projet

## Dataset

Cette data représente Wisconsin Diagnostic Breast Cancer (WDBC) dataset, publié par Center for Machine Learning and Intelligent Systems.



Les données contiennent des mesures sur des cellules dans des bosses suspectes dans le sein d'une femme. Les caractéristiques sont calculées à partir d'une image numérisée d'un aspirant à l'aiguille fine (FNA) d'une masse mammaire. Ils décrivent les caractéristiques des noyaux cellulaires présents dans l'image.

| Data Set Characteristics: | Multivariate | Number of Instances: | 569 | Area: | Life |
| --- | --- | --- | --- | --- | --- |
| Attribute Characteristics: | Real | Number of Attributes: | 32 | Date Donated | 1995-11-01 |
| Associated Tasks: | Classification | Missing Values? | No | Number of Web Hits: | 1443274 |

## Attribut Information :

la dataset contient ID number, Diagnosis (M = malignant, B = benign), et Dix caractéristiques à valeur réelle sont calculées pour chaque noyau cellulaire :

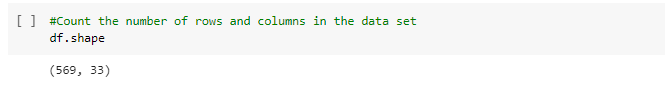
* radius (mean of distances from center to points on the perimeter)
* texture (standard deviation of gray-scale values)
* perimeter
* area
* smoothness (local variation in radius lengths)
* compactness (perimeter^2 / area - 1.0)
* concavity (severity of concave portions of the contour)
* concave points (number of concave portions of the contour)
* symmetry
* fractal dimension ("coastline approximation" - 1)

provide The mean, standard error, and "worst" or largest (mean of the three largest values) de ces caractéristiques ont été calculées pour chaque image,résultant en 30 fonctionnalités. Par exemple, le champ 3 est le rayon moyen, le champ 13 est Radius SE, le 23 est Worst Radius. Toutes les valeurs des caractéristiques sont recodées avec quatre chiffres significatifs.

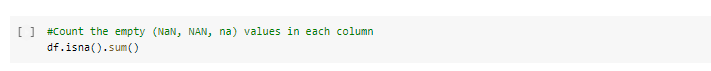
## Pré Processing Data:

Cette partie concerne l’exploration et le nettoyage de datasets et la division de data en quatre parties pour le training et le test.

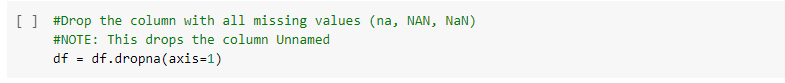
commençons par découvrir notre dataset, pour savoir les champs, nombre de ligne et de colonne :



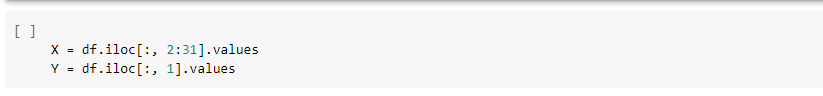
maintenant suppression des valeur null et les Nan value :



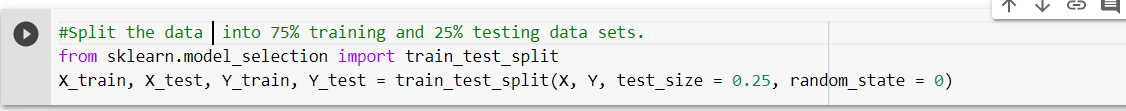
suppression des Nan value :



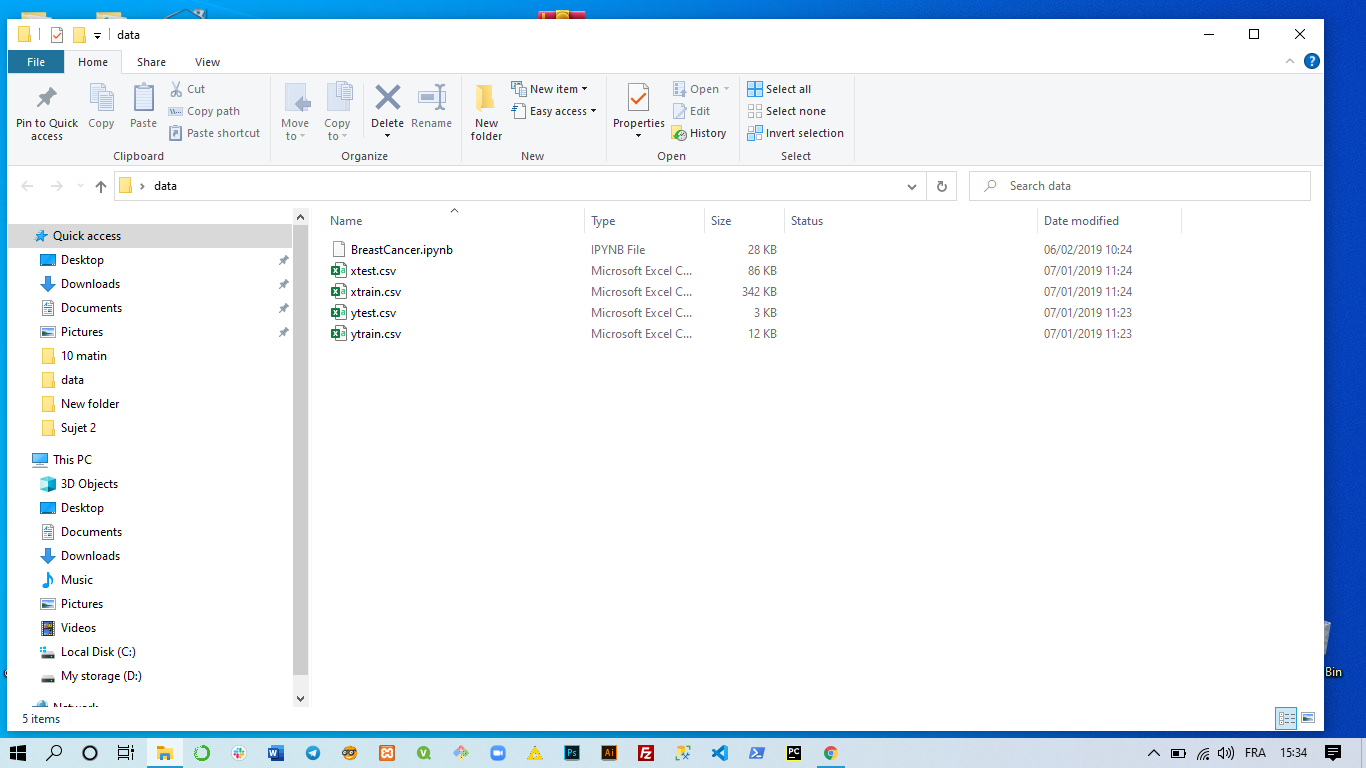
Maintenant, l’étape d'exploration et de nettoyage des données est fini. Je vais configurer mes données pour le modèle en divisant d'abord l'ensemble de données en un ensemble de données d'entités également appelé ensemble de données **indépendant (X)** et ensemble de données cible également appelé ensemble de données **dépendant (Y).**

****

Divisez à nouveau les données, mais cette fois en 75% de training et 25% de jeux de données de test.



après exportation de nouveaux répartition de data sous format csv, on obtient donc quatre fichier csv :



# Outils

## TensorFlow

TensorFlow est une bibliothèque open source de Machine Learning, créée par Google, permettant de développer et d’exécuter des applications de Machine Learning et de Deep Learning

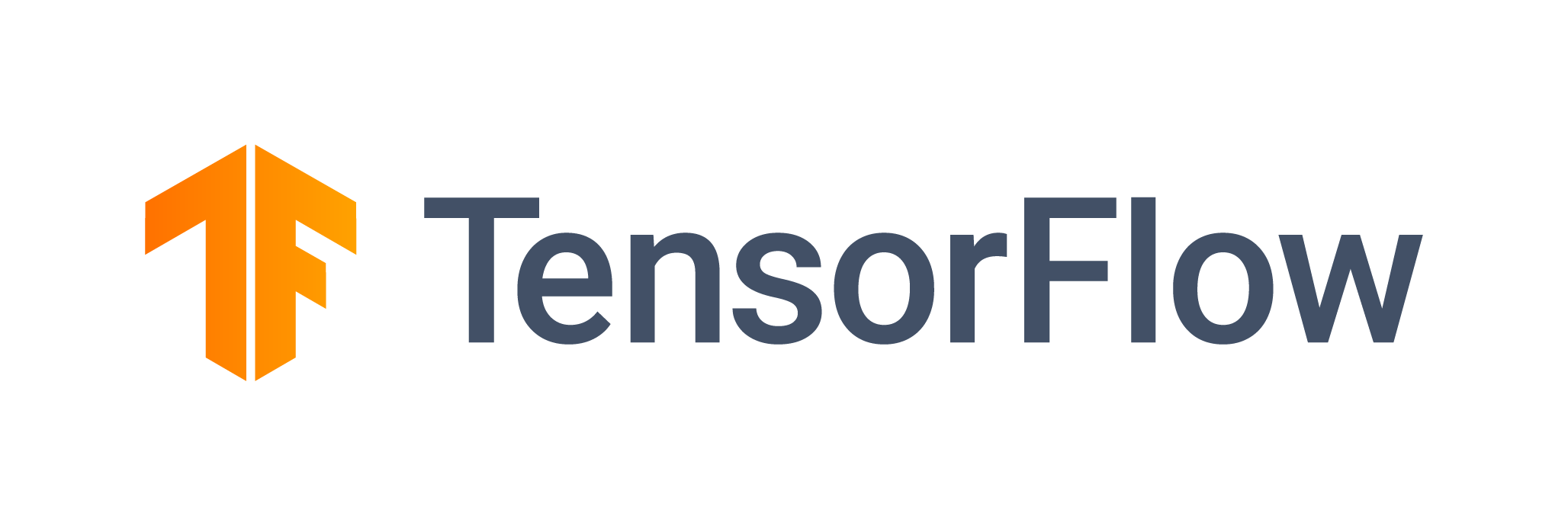
Créé par l’équipe Google Brain en 2011, sous la forme d’un système propriétaire dédié au réseau de neurones de Deep Learning.

TensorFlow s’appelait à l’origine DistBelief. Par la suite, le code source de DisBelief a été modifié et cet outil est devenu une bibliothèque basée application. En 2015, il a été renommé TensorFlow et Google l’a rendu open source.

Depuis lors, il a subi plus de 21000 modifications par la communication et est passé en version 1.0 en février 2017. Pour faire simple, TensorFlow est une bibliothèque de Machine Learning, il s’agit d’une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes avec aisance.

Elle permet aux chercheurs de développer des architectures d’apprentissage expérimentales et de les transformer en logiciels. On peut le concevoir comme un système de programmation dans lequel les calculs sont représentés sous forme de graphiques.

Les nœuds du graphique représentent les opérations mathématiques, et les bordures représentent des flèches de données multidimensionnelles communiquées entre elles : les tensors.



# Keras

tf.keras est l'API de haut niveau de TensorFlow permettant de créer et d'entraîner des modèles de deep learning. Elle est utilisée dans le cadre du prototypage rapide, de la recherche de pointe et du passage en production. Elle présente trois avantages majeurs :

* **Convivialité**

Keras dispose d'une interface simple et cohérente, optimisée pour les cas d'utilisation courants. Elle fournit des informations claires et concrètes concernant les erreurs des utilisateurs.

* **Modularité et facilité de composition**

Les modèles Keras sont créés en connectant des composants configurables, avec quelques restrictions.

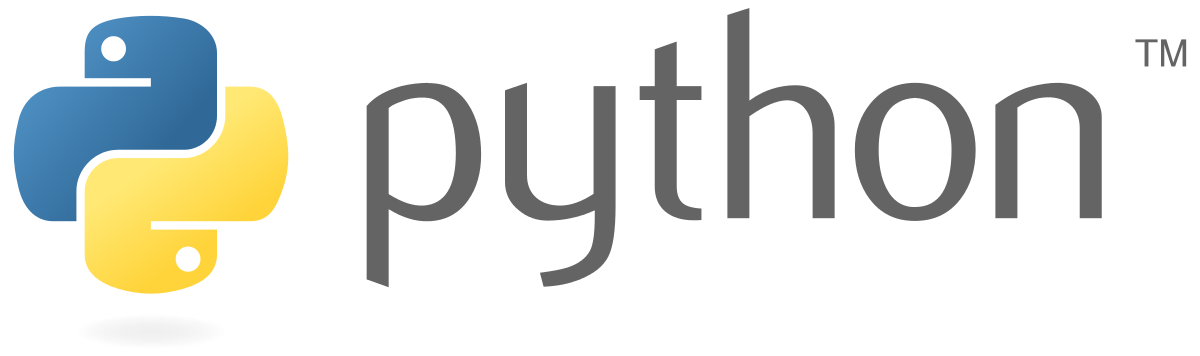
* **Facilité d'extension**

Composez des éléments de base personnalisés pour exprimer de nouvelles idées de recherche. Créez des calques, des métriques et des fonctions de perte, et développez des modèles de pointe.



PYTHON

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions .



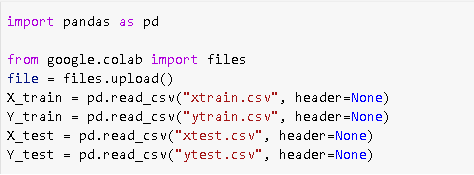
## Google Colab

Google Colab ou Collaboratory est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l’apprentissage automatique. Cette plateforme permet d’entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d’installer quoi que ce soit sur notre ordinateur à l’exception d’un navigateur.



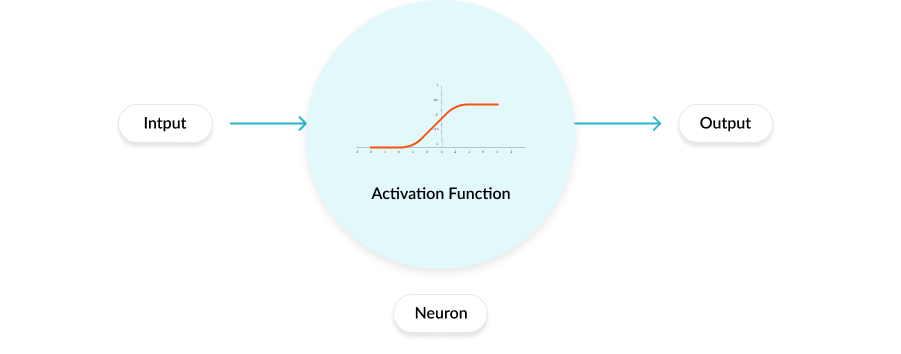
# Traitement

Dans cette première cellule, nous téléchargeons les fichiers CSV contenant les données pré-triées et pré-normalisées. Pour ceux-ci,nous avons utilisé l'ensemble de données sur le cancer du sein du Wisconsin, nous l'avons trié en ensembles d'entraînement et de test avec les valeurs `` d'entrée '' du réseau neuronal en tant que valeurs `` X '' et la `` sortie '' attendue (un 0 si bénin et un 1 si malin) comme les valeurs «Y».



Maintenant, nous allons utiliser Keras pour définir un réseau neuronal qui sera formé à partir de ces données. Ce réseau neuronal peut ensuite être utilisé pour prédire les valeurs futures de la présence de cancer du sein.

**Un modèle Sequential** est approprié pour une pile simple de couches où chaque couche a exactement un tenseur d'entrée et un tenseur de sortie .



La fonction dense prend comme argument :

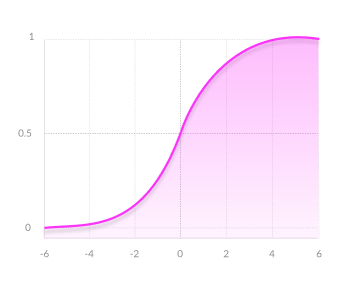
Dense(input\_layer\_neurons=16, hidden\_layer\_neurons=32, kernel\_initializer='normal', activation='relu')

on utilise “relu” et “sigmoid” comme fonction d'activation , or :

**Sigmoïde / Logistique**

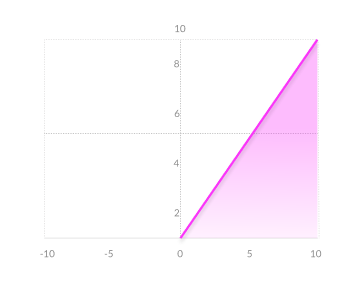
La fonction sigmoïde fourni un Gradient lisse, empêchant les «sauts» dans les valeurs de sortie.

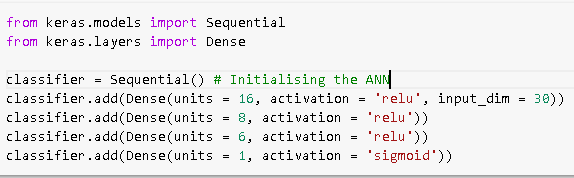
* Les valeurs de sortie sont liées entre 0 et 1, normalisant la sortie de chaque neurone.
* Prédictions claires - Pour X supérieur à 2 ou inférieur à -2, tend à amener la valeur Y (la prédiction) au bord de la courbe, très proche de 1 ou 0. Cela permet des prédictions claires.
* Dégradé de fuite - pour des valeurs de X très élevées ou très faibles, il n'y a pratiquement pas de changement dans la prédiction, ce qui entraîne un problème de gradient de disparition. Cela peut conduire le réseau à refuser d'apprendre davantage ou à être trop lent pour atteindre une prédiction précise.
* Sorties non centrées sur zéro.
* Calcul coûteux



ReLU (unité linéaire rectifiée)

* Efficacité informatique - permet au réseau de converger très rapidement
* Non linéaire - bien qu'il ressemble à une fonction linéaire, ReLU a une fonction dérivée et permet la rétropropagation
* Le problème Dying ReLU - lorsque les entrées approchent de zéro ou sont négatives, le gradient de la fonction devient zéro, le réseau ne peut pas effectuer de rétropropagation et ne peut pas apprendre.





Une fois le réseau de neurones défini, je dois maintenant spécifier la fonction **d'optimisation (optimiser)** et de **perte (loss) .**

1. Les fonctions de perte : Loss

Les fonctions de pertesont utiles pour former un réseau neuronal. Étant donné une entrée et une cible, ils calculent la perte, c'est-à-dire la différence entre la sortie et la variable cible. Dans notre cas on a utilisé **Binary Cross entropy class** qui appartient à la catégorie de **Probabilistic losses** de Keras .

* **Le Binary Cross entropy**  calcule la perte d'entropie croisée entre les étiquettes vraies et les étiquettes prévues.utilisez cette perte d'entropie croisée lorsqu'il n'y a que deux classes d'étiquettes (supposées être 0 et 1). Pour chaque exemple, il doit y avoir une seule valeur à virgule flottante par prédiction.

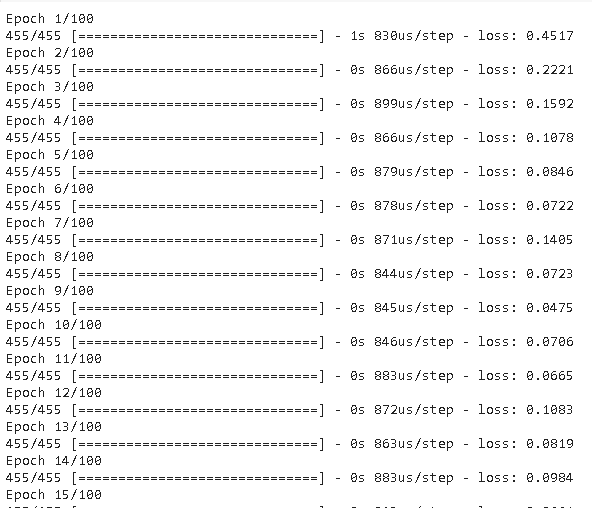
1. Les fonctions d'optimisation : optimizer

sont utilisés pour mettre à jour les pondérations et biaiser les paramètres internes d'un modèle afin de réduire l'erreur. Dans notre cas on a utiliser la catégorie **Adaptive Learning Algorithms**

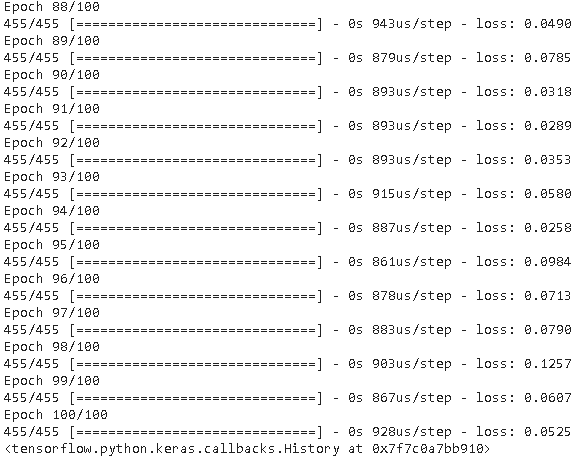


On entraîne maintenant le réseau de neurones en utilisant Classifier.fit, en lui passant les données d'entraînement - c'est-à-dire pour cet ensemble de X, voici à quoi devrait ressembler le Y. Le NN repère ensuite les modèles dans les données et construit un réseau neuronal qui pourrait reproduire cela.





Quand le training est terminé le loss prend 0.525 comme valeur finale.

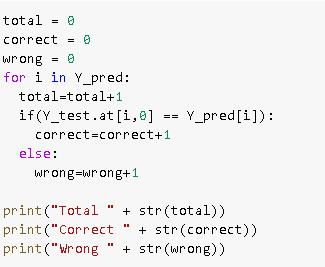


# Résultat

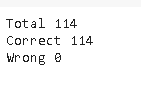
Pour prédire de nouvelles valeurs, le réseau neuronal utilisé classifier.predict. Nous allons lui passer les valeurs de test pour X (que le réseau neuronal n'a pas vues auparavant) et cela me rendra un ensemble de prédictions. Ces prédictions seront des probabilités, donc je vais les nettoyer en disant que si elles sont supérieures à 0,5, je les ferai 1, sinon je les rendrai 0.



Maintenant, nous pouvons parcourir l'ensemble des prédictions pour l'ensemble de test (appelé Y\_pred) et les valeurs réelles pour l'ensemble de test (cellule Y Test), et voir à quel point ils se ressemblent - s'ils sont identiques:



Voila, les resultats sont identiques avec pourcentage de 100%



# 

# Conclusion

# Références

<https://en.wikipedia.org/wiki/Deep_learning>

<https://www.tensorflow.org/guide/keras/sequential_model?hl=fr>

<https://keras.io/api/models/sequential/>

<https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c>

<https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/>

<https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)>