Gestion de Projet Big Data & Développement d'applications Big Data

EDAH Kodjo
Consultant Systèmes d’Information, Big-Data

Objectifs

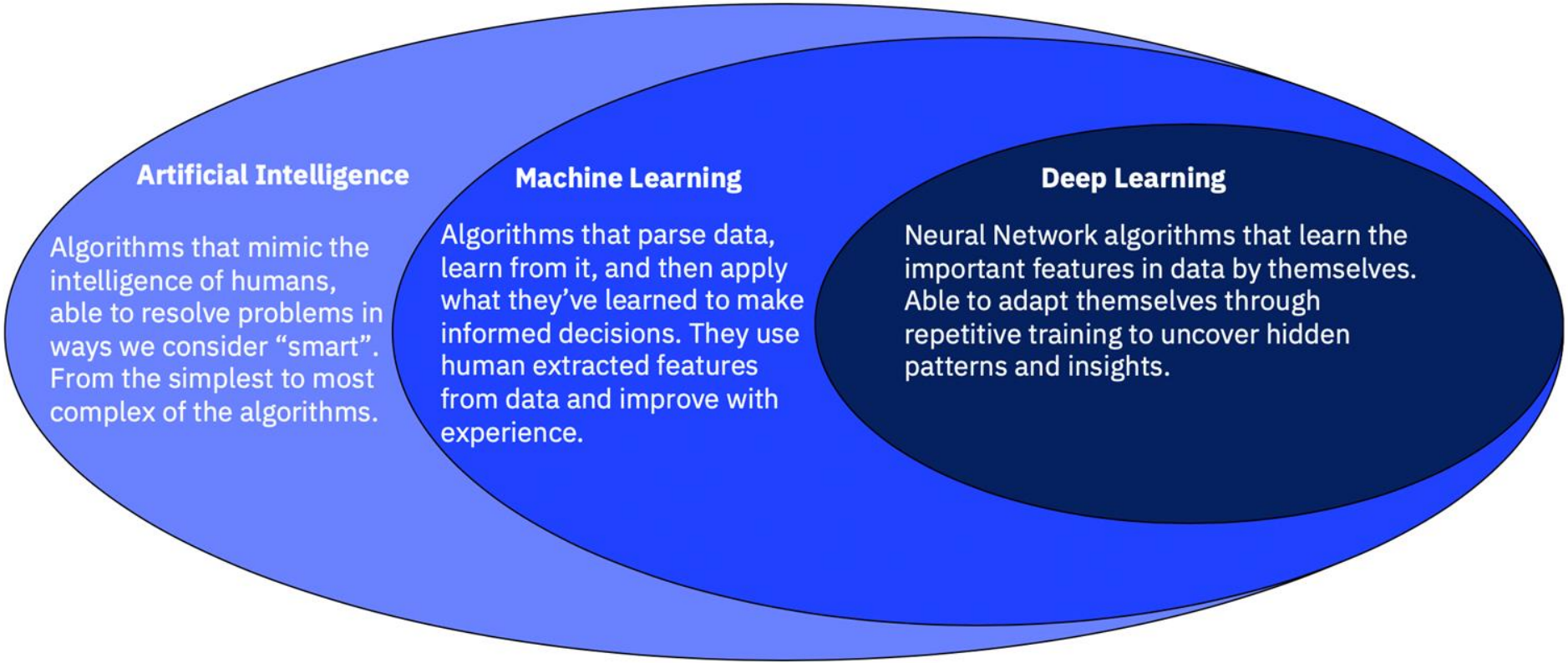
- Comprendre la notion et les spécificités du Big Data
- Connaître les technologies de l’écosystème Hadoop
- Connaître le langage python et utiliser les librairies de machine learning
- Savoir utiliser les outils de visualisation des données (Dataviz)

Partie 3 : Machine learning

Intelligence artificielle

L'intelligence artificielle (IA) est « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence »

Machine Learning et intelligence artificielle



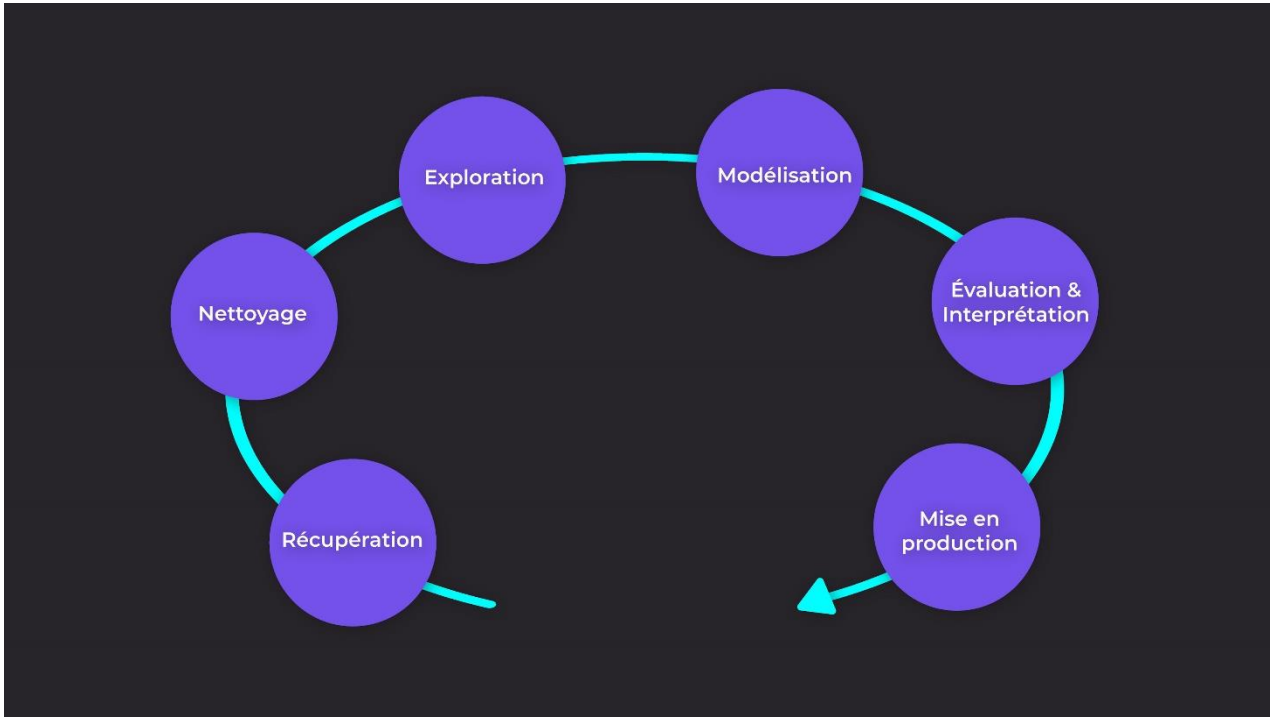
Intelligence artificielle

L'intelligence artificielle (IA) est « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence »

Exemple de problématique de machine learning

- Prédire les ventes
- Identification des objets (image, ……)
- Segmenter les utilisateurs d'un site en plusieurs groupes en fonction de leur comportement sur le site, catégoriser un produit
- Recommandation de produit

Le cycle de travail d'un data science

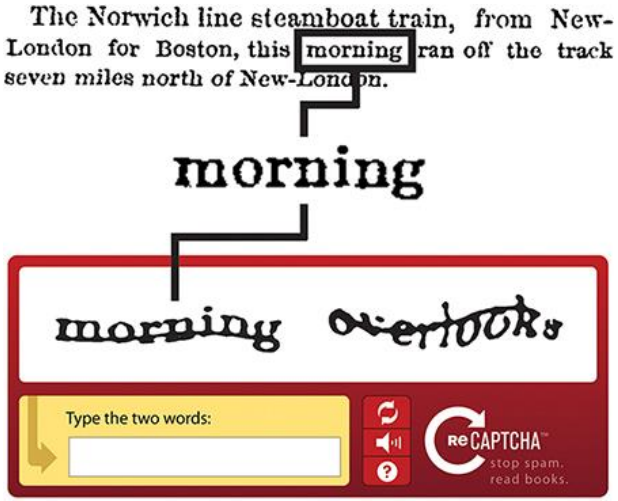


https://user.oc-static.com/upload/2019/04/23/15560369383553_2c2.jpg

Récupération des données

- Les bases de données existantes
- Les données brutes alternatives (image, son, document, pages web, etc.)
- Les réseaux sociaux
- Internet des Objets
- Création de nouveaux canaux d'acquisition de données

- Exemple : Les CAPTCHAs pour la digitalisation automatique de livres



https://user.oc-static.com/upload/2016/09/17/14741229513738_img-2.png

Nettoyage des données

- Suppression des données aberrantes et incohérentes
- Agrégation si nécessaire

- Les batchs ou job map-reduce, spark

Exploration des données

- Comprendre les différents comportements
- Détecter les schémas
- Tâche destinée au Data Analyst

- Résultats
  - Proposer plusieurs hypothèses sur les causes sous-jacentes à la génération du dataset
  - Proposer plusieurs pistes de modélisation statistique des données, qui vont permettre de résoudre la problématique de départ considérée.
  - Proposer si nécessaire de nouvelles sources de données qui aideraient à mieux comprendre le phénomène.

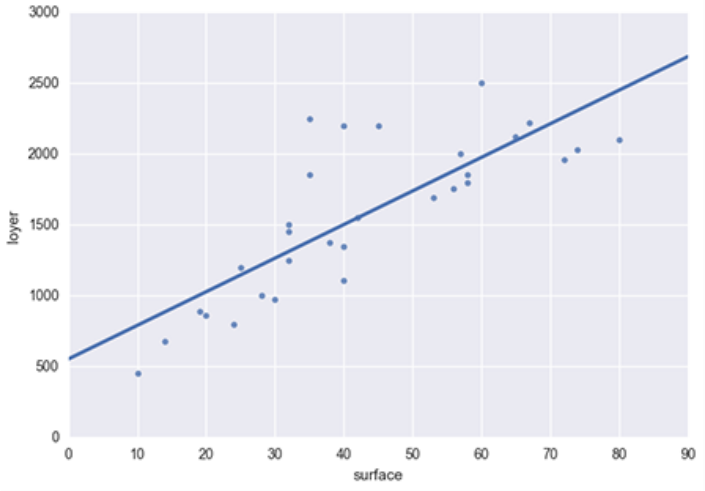
Modélisation

- C’est l’étape du machine learning ou apprentissage
- Application des algorithmes de d’apprentissage
  - la régression linéaire
  - K-nn
  - les Support Vector Machine (SVM)
  - les réseaux de neurones
  - les random forests.
  - Clustering
  - Collaborive filtering

PyTorch

scikit learn

TensorFlow

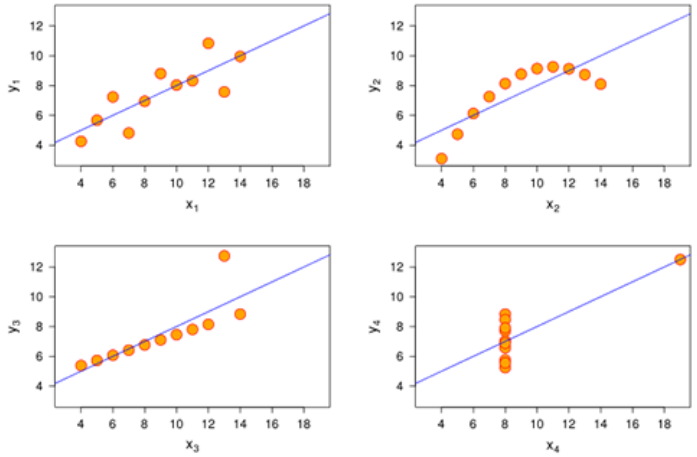


https://user.oc-static.com/upload/2016/09/17/14741406902223_download-2.png

Evaluation du modèle

- Le modèle représente t-il avec exactitude le phénomène ?
- Le modèle résout t-il le problème ?
- Quelle est la marge d'erreur ?
- Quelle est la performance du modèle

- Le quartet d'Anscombe



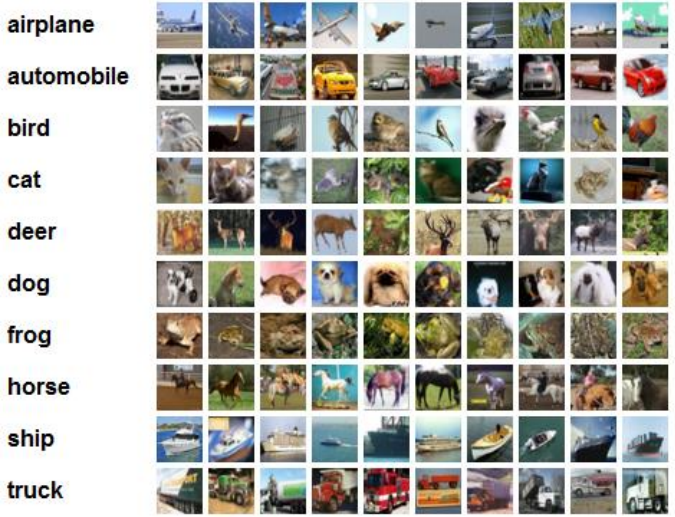
https://user.oc-static.com/upload/2016/09/17/14741418471714_640px-Anscombe.svg.png

Mise en production

- Déploiement du modèle en production
- Mise en place des supports de production
- Infrastrusture big data (Hadoop, AWS, Azure ……)

Familles d'algorithmes d'apprentissage existantes

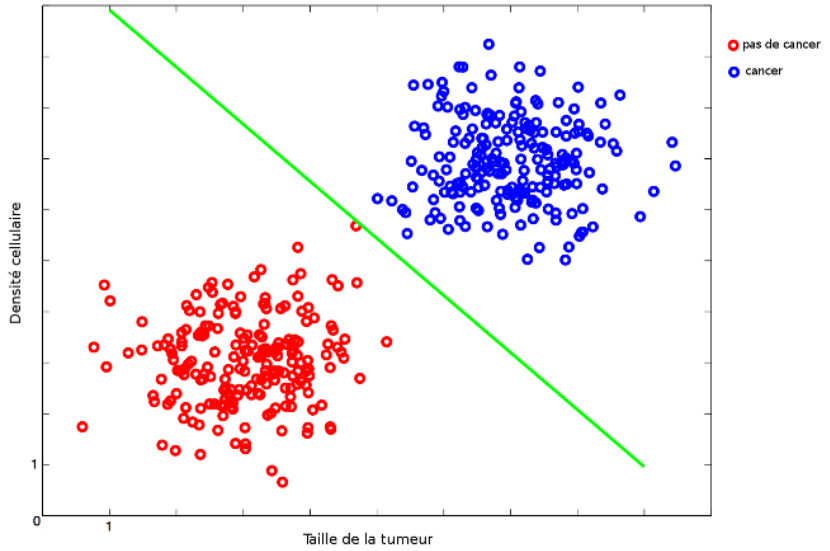
- Apprentissage « supervisé » : supervised learning
- Données sont annotées ou labélisées
- Features (x) vs Label (y) ou target
- But : Prédire y à partir des x
- Problème : comment labéliser ?



https://user.oc-static.com/upload/2016/10/24/14773158929787_cifar_preview.png

Familles d'algorithmes d'apprentissage existantes

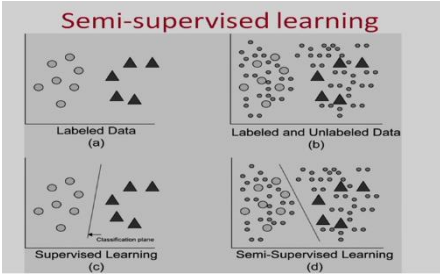
- Apprentissage « non supervisé » : unsupervised learning
- Les données ne sont pas annotées
- L'algorithme determine lui même les similarités dans le dataset



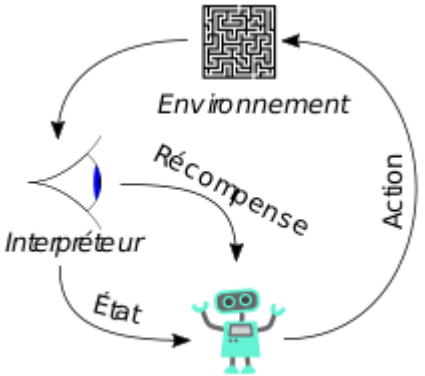
https://markdown.data-ensta.fr/uploads/upload_87ef9ad65f9163ff5a92e1691eb4d1bd.png

Familles d'algorithmes d'apprentissage existantes

- le semi-supervised learning : combine supervised et unsupervised



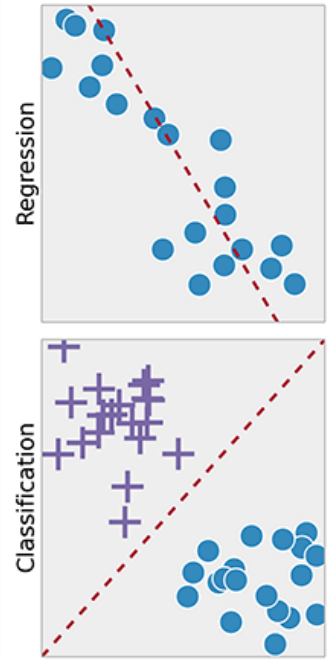
- le reinforcement learning : qui se base sur un cycle d'expérience / récompense et améliore les performances à chaque itération
  - Jeu de go, damier, échec



Source : https://www.digitalvidya.com/blog/semi-supervised-learning/ , https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Reinforcement_learning_diagram.svg/250px-Reinforcement_learning_diagram.svg.png

Familles d'algorithmes d'apprentissage existantes

- Régression
  - Recherche t-on un nombre ?
  - Valeur continue
- Classification
  - Recherche t-on une catégorie ?
  - Valeur discrète



Source : https://user.oc-static.com/upload/2016/09/18/14742103795655_ml.png

Segmentation des datasets

- Training set :
  - sous-ensemble destiné à l'apprentissage d'un modèle.
  - Exemple : Proportion 80% du dataset
- Validation set/ Test set
  - sous-ensemble destiné à l'évaluation du modèle.
  - Exemple : Proportion 20% du dataset

- N'effectuez jamais l'apprentissage sur des données d'évaluation

Cas pratique : scikit learn

- Librairie python machine learning
- Des outils simples et efficaces pour l'analyse prédictive des données
- Accessible à tous et réutilisable dans divers contextes
- Construit sur NumPy, SciPy et matplotlib
- Open source

User Guide

https://scikit-learn.org/stable/user_guide.html

Cas pratique : scikit learn

- Fonctionnement
  - Etape 1 : Sélectionner un estimateur et configurer les hyperparamètres (Créer une instance du modèle)
  - Etape 2 : Entrainer le modèle (méthode fit)
  - Etape 3 : Evaluer le modèle (méthode score)
  - Etape 4 : Utiliser le modèle (méthode predict)

```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
X = np.array([1, 4 , 5 , 6, 10, 50, 30, 15, 20, 45]).reshape(-1,1)
y = np.array([1, 2 , 3 , 7, 12, 30, 25, 10, 10, 25])

plt.scatter(X,y)
```

```
# Instance du modèle
model = LinearRegression()
# Entrainer le modèle
model.fit(X, y)
# Evaluer le modèle
model.score(X,y)
# Utiliser le modèle
model.predict([[36]])
```

Cas pratique : scikit learn

- Apprentissage supervisé

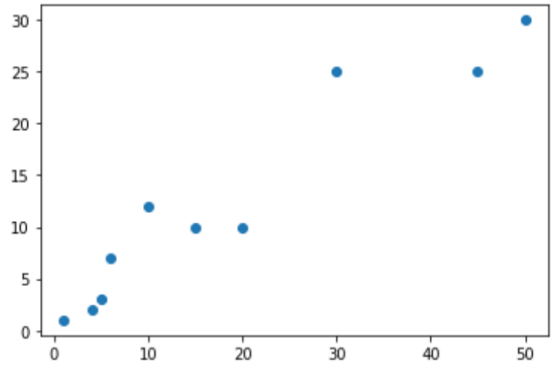
  - Apprentissage « supervisé » : supervised learning
  - Données sont annotées ou labélisées
  - Features (x) vs Label (y) ou target
  - But : Prédire y à partir des x
  - Problème : comment labéliser ?

Cas pratique : scikit learn

- Régression linéaire

```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
X = np.array([1, 4 , 5 , 6, 10, 50, 30, 15, 20, 45]).reshape(-1,1)
y = np.array([1, 2 , 3 , 7, 12, 30, 25, 10, 10, 25])

plt.scatter(X,y)
```

```
<matplotlib.collections.PathCollection at 0x7fb3e05e9990>
```



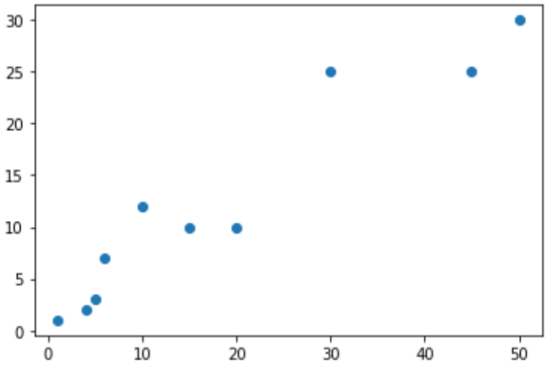

```
# Instance du modèle
model = LinearRegression()
# Entrainer le modèle
model.fit(X, y)
# Evaluer le modèle
model.score(X,y)
# Utiliser le modèle
model.predict([[36]])
```

Cas pratique : scikit learn

- Régression linéaire

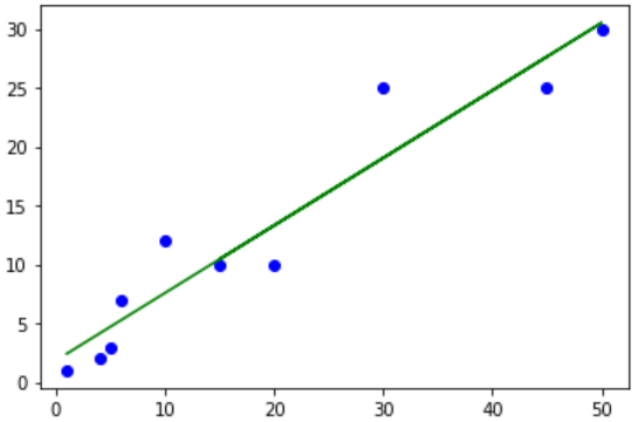
```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
X = np.array([1, 4 , 5 , 6, 10, 50, 30, 15, 20, 45]).reshape(-1,1)
y = np.array([1, 2 , 3 , 7, 12, 30, 25, 10, 10, 25])

plt.scatter(X,y)
```

```
<matplotlib.collections.PathCollection at 0x7fb3e05e9990>
```




```
model = LinearRegression()
model.fit(X, y)
model.score(X,y)
y_predict = model.predict(X)

plt.scatter(X,y, c='b')
plt.plot(X, y_predict, c='g')
```

```
[<matplotlib.lines.Line2D at 0x7fb3df9cef50>]
```



Cas pratique : scikit learn

- Classification

- Exemple : entrainer un modèle sur les données du Titanic pour déterminer si une personne pourra survivre ou non

```
# Exemple les données du titanic
# Entrainer un modèle pour savoir si quelqu'un pouvait survivre ou non

titanic_data = snb.load_dataset('titanic')

titanic_data
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_tow
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampto
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbour
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampto
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampto
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampto

Cas pratique : scikit learn

- Classification

- Exemple : entraîner un modèle sur les données du Titanic pour déterminer si une personne pourra survivre ou non

- Nous allons entraîner le modèle uniquement sur les colonnes age, sex, pclass et survived

```
titanic_data = titanic_data[['age','pclass','sex','survived']]

titanic_data
```

	age	pclass	sex	survived
0	22.0	3	male	0
1	38.0	1	female	1
2	26.0	3	female	1
3	35.0	1	female	1

Cas pratique : scikit learn

❑ Classification

- ❑ Exemple : entraîner un modèle sur les données du Titanic pour déterminer si une personne pourra survivre ou non
- ❑ Nous allons entraîner le modèle uniquement sur les colonnes age, sex, pclass et survived
- ❑ **Normaliser les données**

```
# Normaliser les données
titanic_data = titanic_data.dropna(axis='index')
# Tout convertir en valeur numérique
titanic_data['sex'].replace(['female','male'], [0,1], inplace=True)

titanic_data
```

	age	pclass	sex	survived
0	22.0	3	1	0
1	38.0	1	0	1
2	26.0	3	0	1
3	35.0	1	0	1
4	35.0	3	1	0

Cas pratique : scikit learn

❑ Classification

- ❑ Exemple : entraîner un modèle sur les données du Titanic pour déterminer si une personne pourra survivre ou non
- ❑ Nous allons entraîner le modèle uniquement sur les colonnes age, sex, pclass et survived
- ❑ Normaliser les données
- ❑ **Entrainer et évaluer le modèle KNeighborsClassifier**

```
from sklearn.neighbors import KNeighborsClassifier

X= titanic_data[['age','pclass', 'sex']]
y= titanic_data['survived']
titanic_model = KNeighborsClassifier()
titanic_model.fit(X.values,y)
score = titanic_model.score(X.values,y)

# Préfire la survie d'une personne en classe 2 ayant 23
X_to_predict = np.array([[23,2,1]])
print(X_to_predict)
predict = titanic_model.predict(X_to_predict)

print("Le score du modèle est {}".format(score))
print(predict)
```

```
[[23  2  1]]
Le score du modèle est 0.8305322128851541
[0]
```

Cas pratique : scikit learn

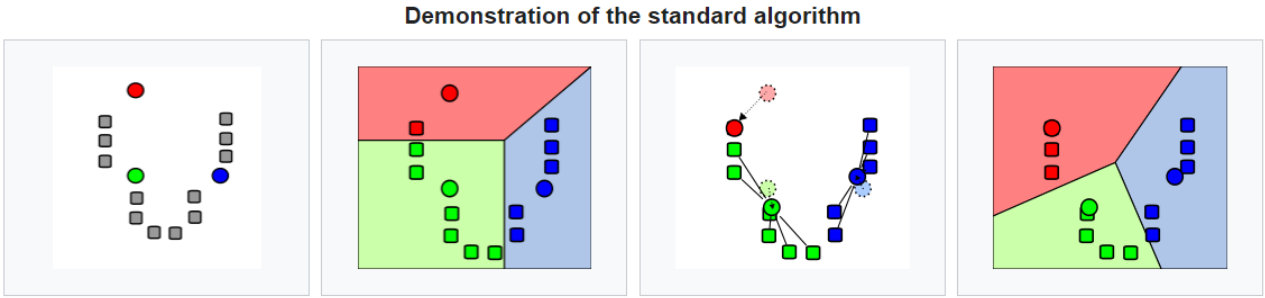
- Apprentissage non supervisé

  - Apprentissage « non supervisé » : unsupervised learning
  - Les données ne sont pas annotées
  - L’algorithme determine lui même les similarités dans le dataset

Cas pratique : scikit learn

- K-means clustering

  - regrouper les données “similaires” en groupes (ou clusters)



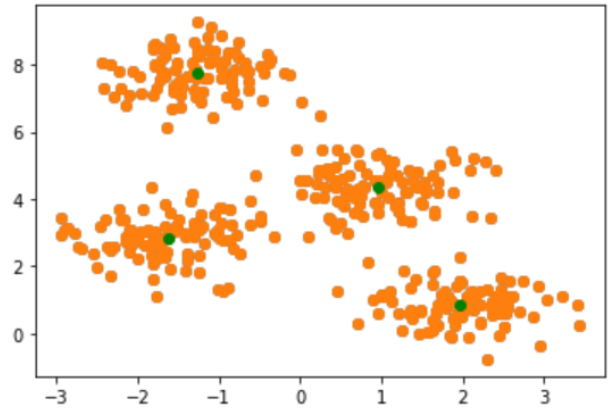
https://en.wikipedia.org/wiki/K-means_clustering

Cas pratique : scikit learn

- K-means clustering
  - regrouper les données “similaires” en groupes (ou clusters)

```
plt.scatter(X[:,0], X[:,1])
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(X)

plt.scatter(X[:,0], X[:,1])
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1], c="g")
```

```
<matplotlib.collections.PathCollection at 0x7fdcadb42d90>
```

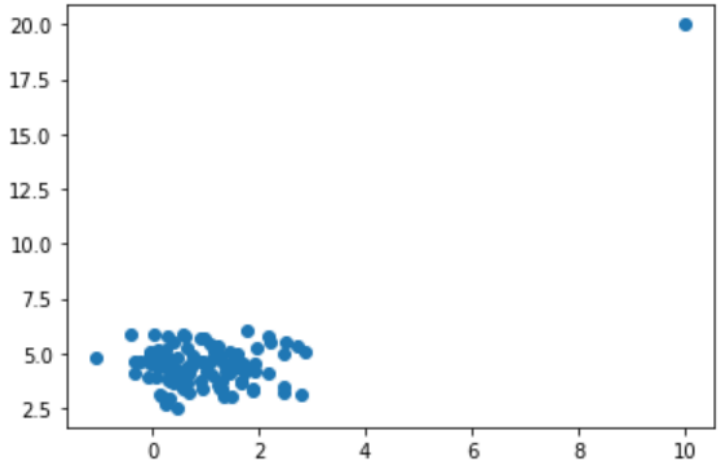


Cas pratique : scikit learn

- Isolation Forest
  - Détection d'anomalie

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
# Generate sample data
X, y = make_blobs(n_samples= 100, centers= 1,
                  cluster_std=0.80, random_state=0 )

X[-1, :] = np.array([10, 20])
plt.scatter(X[:,0],X[:,1])
```

```
<matplotlib.collections.PathCollection at 0x7feba44b5910>
```



Cas pratique : scikit learn

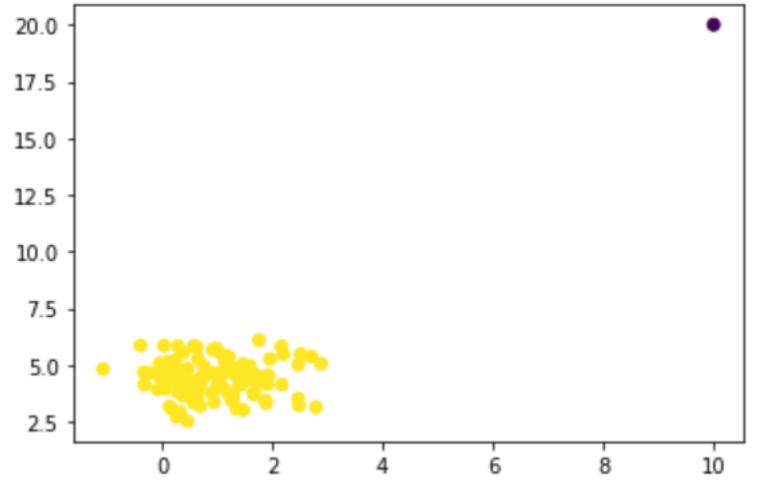
- Isolation Forest

  - Détection d'anomalie

  - Contamination : la proportion de valeurs aberrantes dans l'ensemble de données. Utilisé lors de l'ajustement pour définir le seuil sur les scores des échantillons.

```
from sklearn.ensemble import IsolationForest

model_isolation = IsolationForest(contamination=0.01)
model_isolation.fit(X)
predict = model_isolation.predict(X)
plt.scatter(X[:,0],X[:,1], c=predict)
```

```
<matplotlib.collections.PathCollection at 0x7feba3fb90d0>
```



Cas pratique : scikit learn

- Selection des données de training et de test

N'effectuez jamais l'apprentissage sur des données d'évaluation

```
# Sur les données du titanic nous allons séparer
#le Training Set et Test Set

from sklearn.model_selection import train_test_split

# Default size train_size=0.8 test_size=0.2
X_train, X_test, y_train, y_test = train_test_split(X,y)

print("Taille training set {}".format(X_train.shape))
print("Taille test set {}".format(X_test.shape))
```

```
Taille training set (535, 3)
Taille test set (179, 3)
```

Cas pratique : scikit learn

- Selection des données de training et de test

```
titanic_model.fit(X_train, y_train)

print("Score training set {}".format(titanic_model.score(X_train, y_train)))
print("Score test set {}".format(titanic_model.score(X_test, y_test)))
```

```
Score training set 0.8317757009345794
Score test set 0.7821229050279329
```

Cas pratique : scikit learn

- Selection des données de training et de test
- Que remarquez-vous ?

```
titanic_model = KNeighborsClassifier(n_neighbors=2)
titanic_model.fit(X_train, y_train)

print("Score training set {}".format(titanic_model.score(X_train, y_train)))
print("Score test set {}".format(titanic_model.score(X_test, y_test)))
```

```
Score training set 0.8448598130841122
Score test set 0.7430167597765364
```

Cas pratique : scikit learn

- Selection des données de training et de test

  - Cherchons le nombre de voisin optimal qui rend le modèle optimal

  - Problème : le test set est aussi fixe. Les réglages sont liés corrélés à la répartition

  - Solution : en plus du train set, test set, il faut une validation set

```
titanic_model = KNeighborsClassifier(n_neighbors=2)
titanic_model.fit(X_train, y_train)

print("Score training set {}".format(titanic_model.score(X_train, y_train)))
print("Score test set {}".format(titanic_model.score(X_test, y_test)))
```

```
Score training set 0.8448598130841122
Score test set 0.7430167597765364
```

```
titanic_model = KNeighborsClassifier(n_neighbors=3)
titanic_model.fit(X_train, y_train)

print("Score training set {}".format(titanic_model.score(X_train, y_train)))
print("Score test set {}".format(titanic_model.score(X_test, y_test)))
```

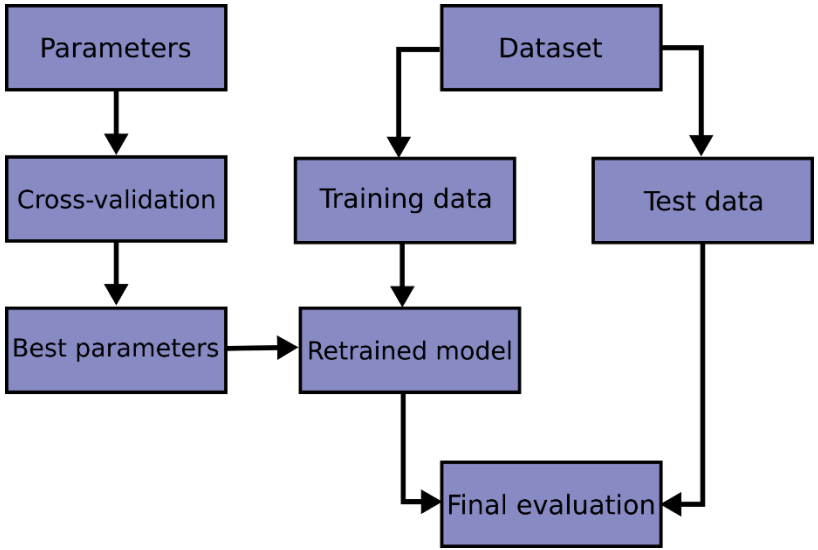
```
Score training set 0.8654205607476636
Score test set 0.7597765363128491
```

```
titanic_model = KNeighborsClassifier(n_neighbors=4)
titanic_model.fit(X_train, y_train)

print("Score training set {}".format(titanic_model.score(X_train, y_train)))
print("Score test set {}".format(titanic_model.score(X_test, y_test)))
```

```
Score training set 0.822429906542056
Score test set 0.7150837988826816
```

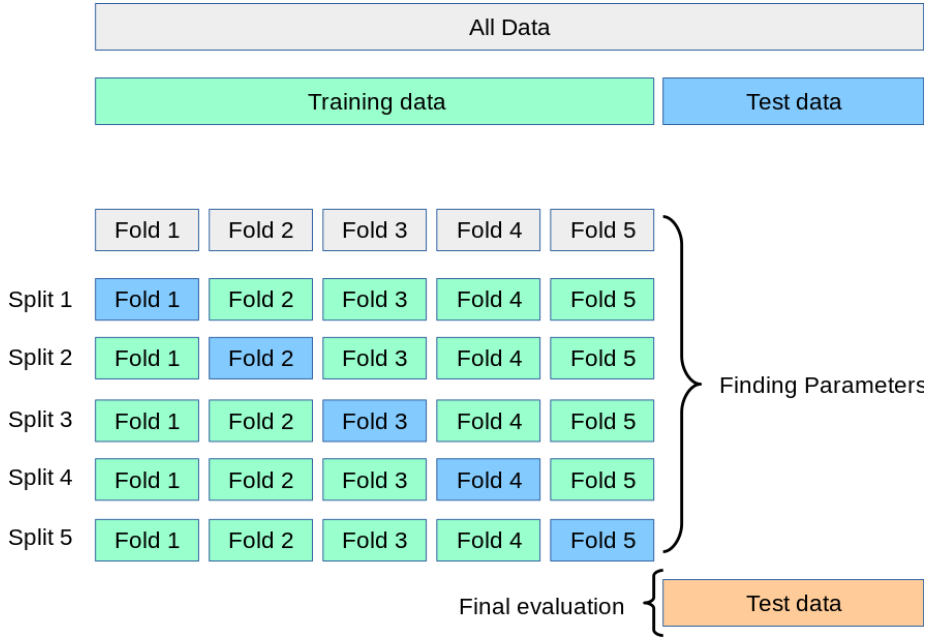
Cas pratique : scikit learn

- Cross validation test



https://scikit-learn.org/stable/modules/cross_validation.html

- K-Fold



https://scikit-learn.org/stable/_images/grid_search_workflow.png

https://scikit-learn.org/stable/modules/cross_validation.html

Cas pratique : scikit learn

- Cross validation test

```
# Cross validation test

from sklearn.model_selection import cross_val_score
print("Training avec des validation set")
for i in range(1,5):
  print(cross_val_score(KNeighborsClassifier(n_neighbors=i), X_train, y_train, cv=6))


# Choisir le modèle avec la bonne moyenne
print("Moyenne Training avec des validation set")
for i in range(1,5):
  print(cross_val_score(KNeighborsClassifier(n_neighbors=i), X_train, y_train, cv=6).mean())
```

```
Training avec des validation set
[0.76666667 0.68539326 0.6741573  0.75280899 0.62921348 0.78651685]
[0.73333333 0.70786517 0.69662921 0.7752809  0.6741573  0.76404494]
[0.75555556 0.76404494 0.6741573  0.83146067 0.69662921 0.75280899]
[0.74444444 0.78651685 0.69662921 0.80898876 0.70786517 0.79775281]
Moyenne Training avec des validation set
0.7157927590511859
0.7252184769038701
0.7457761131918436
0.7570328755722014
```

Cas pratique : scikit learn

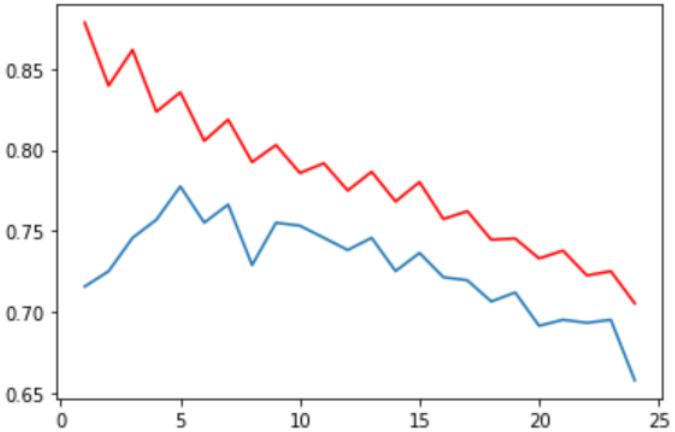
- Cross validation test

- Seul le parameter neighbors a été modifié.

- Pour évaluer un ensemble de paramètre
  - GridSearchCV

```
from sklearn.model_selection import validation_curve
neighborsParam = np.arange(1, 25)
train_score, val_score = validation_curve(KNeighborsClassifier(), X_train, y_train,
                                          param_name='n_neighbors',
                                          param_range=neighborsParam,
                                          cv=6 )
#print(val_score)
#print(train_score)
plt.plot(neighborsParam, val_score.mean(axis=1))
plt.plot(neighborsParam, train_score.mean(axis=1), c='r')
```

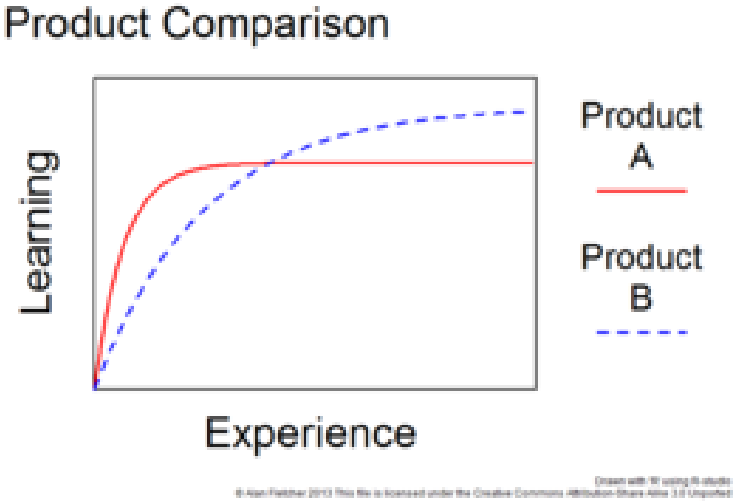
```
[<matplotlib.lines.Line2D at 0x7fb3d5547dd0>]
```



Cas pratique : scikit learn

- Learn curbe (Courbe d'apprentissage)

- Quelle quantité de données optimales pour un meilleur apprentissage



https://upload.wikimedia.org/wikipedia/commons/thumb/7/77/Alanf777_Lcd_fig09.png/405px-Alanf777_Lcd_fig09.png

Cas pratique : scikit learn

- Preprocessing
- But : transformer toutes les valeurs non numériques en valeurs numériques

- LabelEncoder
- OrdinalEncoder
- OneHotEncoder

```
from sklearn.preprocessing import LabelEncoder
# Analyse des données du football
# Partie du corps ayant marqué le but
X = np.array(["Pied","Tete","Poitrine","Pied"])
encoder = LabelEncoder()
# Analyser les dataset
encoder.fit(X)
# Appliquer la transformation
encoder.transform(X)
```

```
array([0, 2, 1, 0])
```

```
# Analyser et transformer
encoder.fit_transform(X)
```

```
array([0, 2, 1, 0])
```

Cas pratique : scikit learn

- Preprocessing
- But : transformer toutes les valeurs non numériques en valeurs numériques

- LabelEncoder
- OrdinalEncoder
- OneHotEncoder

```
from sklearn.preprocessing import OrdinalEncoder
# Analyse des données du football
# Partie du corps ayant marqué le but
X = np.array([["Pied","Gauche"],
              ["Tete","Front"],
              ["Poitrine", "Haute"],
              ["Pied","Droit"]])
# Pour encoder les tableaux mutli-dimensions
encoder = OrdinalEncoder()
# Analyser le dataset et transformer
encoder.fit_transform(X)
```

```
array([[0., 2.],
       [2., 1.],
       [1., 3.],
       [0., 0.]])
```

Cas pratique : scikit learn

- Preprocessing

- But : transformer toutes les valeurs non numériques en valeurs numériques

- LabelEncoder
- OrdinalEncoder
- OneHotEncoder

```
from sklearn.preprocessing import LabelBinarizer
# Analyse des données du football
# Partie du corps ayant marqué le but
X = np.array(["Pied","Tete","Poitrine","Pied"]).reshape(-1,1)

# Pour encoder les tableaux mutli-dimensions
encoder = LabelBinarizer()
# Analyser le dataset et transformer
encoder.fit_transform(X)
```

```
array([[1, 0, 0],
       [0, 0, 1],
       [0, 1, 0],
       [1, 0, 0]])
```

- Eviter de mettre les valeurs numériques sans réel sens arithmétique
- Matrice creuse (sparse matrix)
- Compressed Sparse Row (Csr)

Cas pratique : scikit learn

- Preprocessing

- Transformers : Transformez les features en adaptant chaque fonctionnalité à une plage donnée. Exemple : [0-1]

- MinMaxScaler (inefficace face valeurs aberrantes)
- StandardScaler (inefficace face valeur aberrantes)
- RobustScaler

```
from sklearn.preprocessing import MinMaxScaler

X = np.array([10, 20, 40, 100]).reshape(-1,1)

scaler = MinMaxScaler();
scaler.fit_transform(X)
```

```
array([[0.        ],
       [0.11111111],
       [0.33333333],
       [1.        ]])
```

Cas pratique : scikit learn

❑ Classification

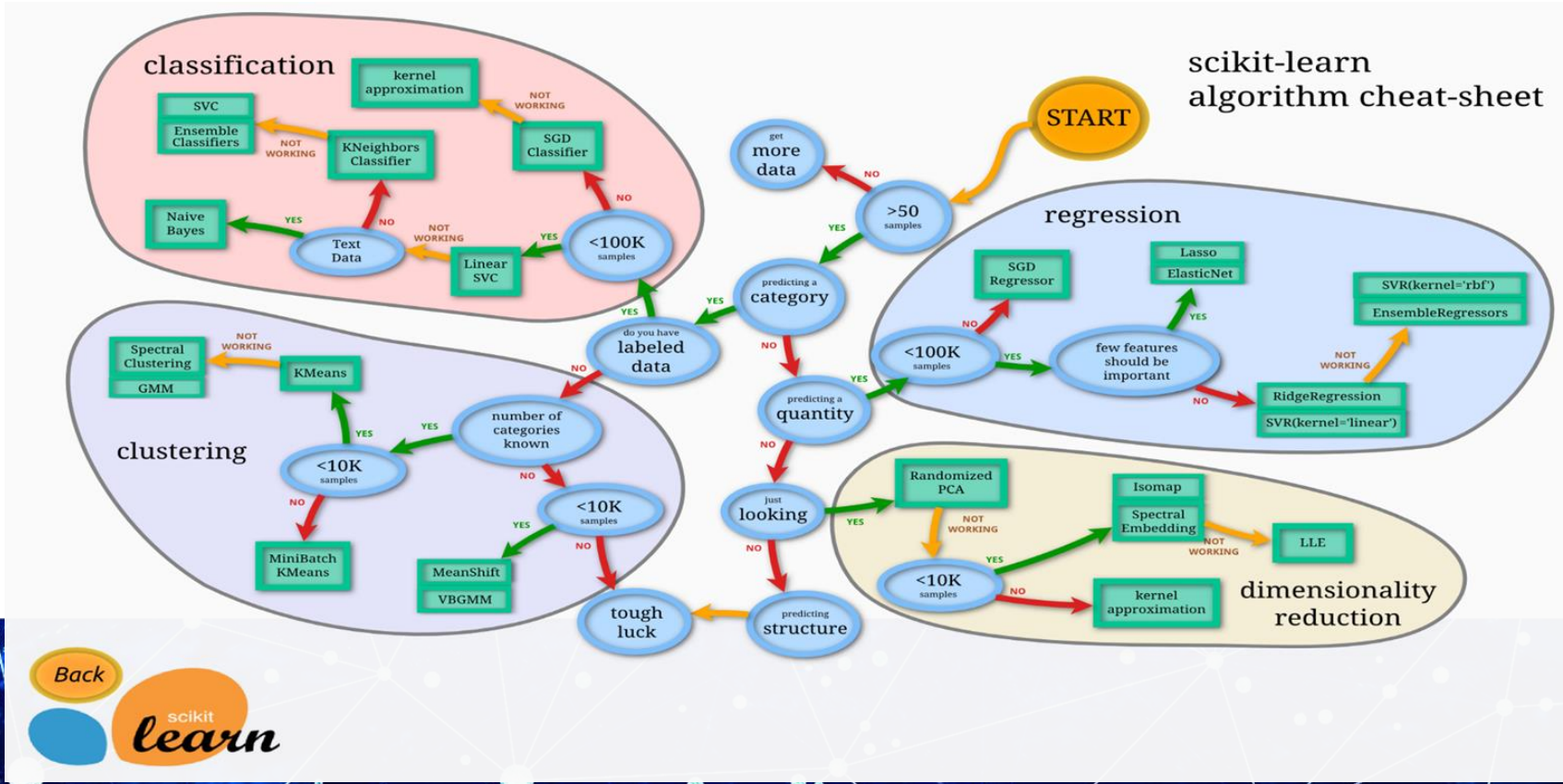
- ❑ Exemple : entrainer un modèle sur les données du Titanic pour déterminer si une personne pourra survivre ou non
- ❑ Nous allons entrainer le modèle uniquement sur les colonnes age, sex, pclass et survived
- ❑ Normaliser les données
- ❑ **Entrainer et évaluer le modèle KNeighborsClassifier**

```
from sklearn.neighbors import KNeighborsClassifier

X= titanic_data[['age','pclass', 'sex']]
y= titanic_data['survived']
titanic_model = KNeighborsClassifier()
titanic_model.fit(X.values,y)
score = titanic_model.score(X.values,y)

# Préfire la survie d'une personne en classe 2 ayant 23
X_to_predict = np.array([[23,2,1]])
print(X_to_predict)
predict = titanic_model.predict(X_to_predict)

print("Le score du modèle est {}".format(score))
print(predict)
```

```
[[23  2  1]]
Le score du modèle est 0.8305322128851541
[0]
```

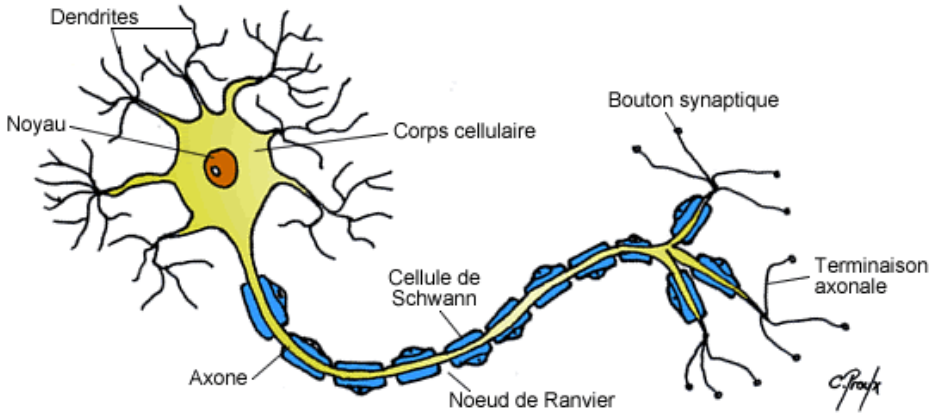
Cas pratique : scikit learn

- Comment choisir le bon modèle



Les réseaux de neurones artificiels

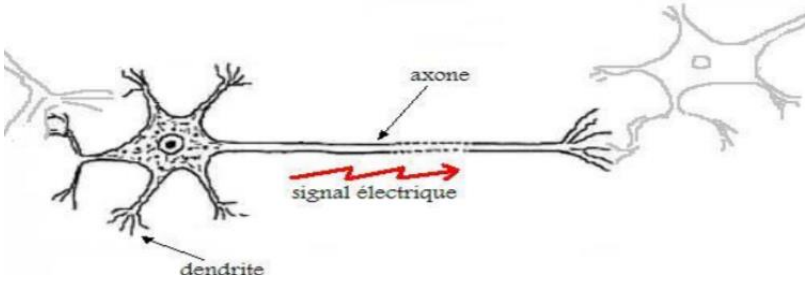
- Inspiré des réseaux de neurones humains



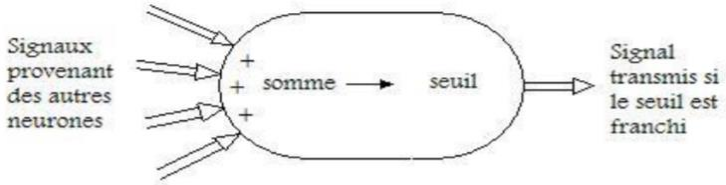
http://ressources.unisciel.fr/DAEU-biologie/P2/res/chap4_im05.png

Métaphore biologique

- Fonctionnement du cerveau Transmission de l'information et apprentissage

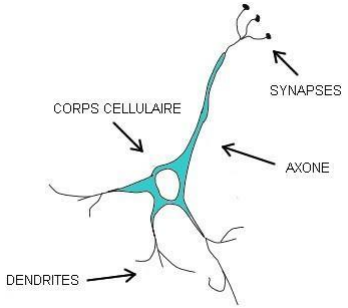


- Un perceptron !!!!

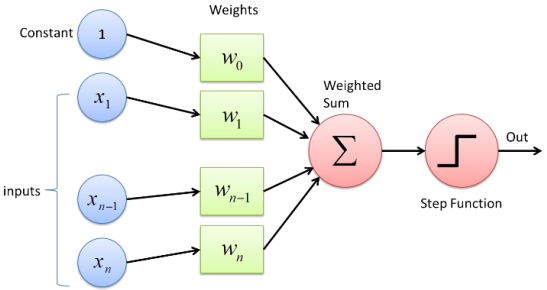


Un neurone : le perceptron

- Les neurones reçoivent des signaux (impulsions électriques) par les dendrites et envoient l ’information par les axones.
- Les contacts entre deux neurones (entre axone et dendrite) se font par l ’intermédiaire des synapses.

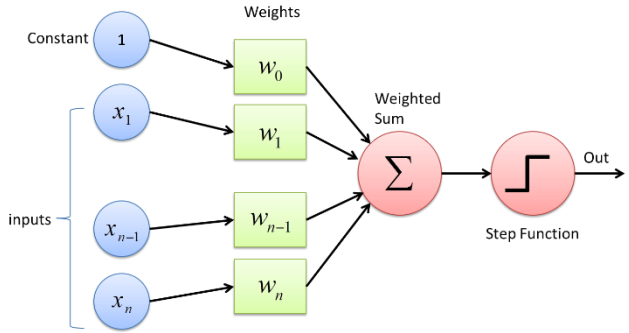


Neurone biologique	Neurone artificiel
Axones	Signal de sortie
Dendrites	Signal d'entrée
Synapses	Poids de la connexion



Un neurone : le perceptron

- Frank Rosenblatt en 1956
- Algorithme d'apprentissage supervisé de classifieurs binaires
- Inconvénient : linéarité



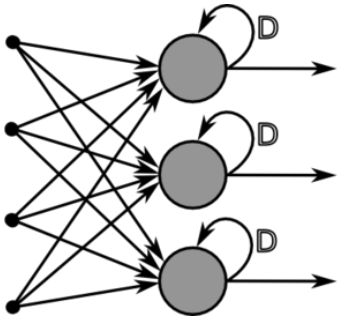
https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

Un perceptron à *n* entrées $(x_1, \ldots, x_n)$ et à une seule sortie *o* est défini par la donnée de *n* poids (ou coefficients synaptiques) $(w_1, \ldots, w_n)$ et un biais (ou seuil) θ par[2] :

$$o = f(z) = \begin{cases} 1 & \text{si} \quad \sum_{i=1}^{n} w_i x_i > \theta \\ 0 & \text{sinon} \end{cases}$$

Un neurone : le perceptron

- La règle de Hebb
- Correction du modèle (loi de Widrow-Hoff)
- Inconvénient : linéarité



$$W_i' = W_i + \alpha(Y_t - Y)X_i$$

W_i' = le poids i corrigé

Y_t = sortie attendue

Y = sortie observée

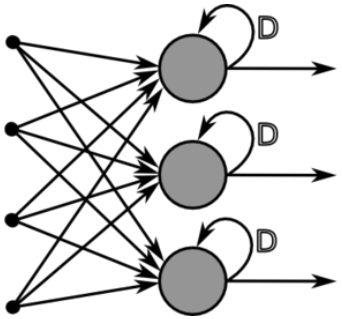
α = le taux d'apprentissage

X_i = l'entrée du poids i pour la sortie attendue Y_t

W_i = le poids i actuel

Un neurone : le perceptron

- La règle de Hebb
- Correction du modèle (loi de Widrow-Hoff)
- Inconvénient : linéarité



$$W_i' = W_i + \alpha(Y_t - Y)X_i$$

W_i' = le poids i corrigé

Y_t = sortie attendue

Y = sortie observée

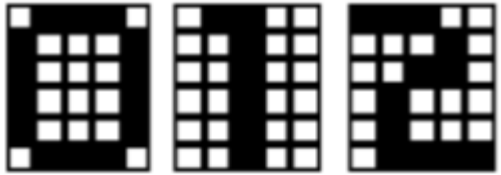
α = le taux d'apprentissage

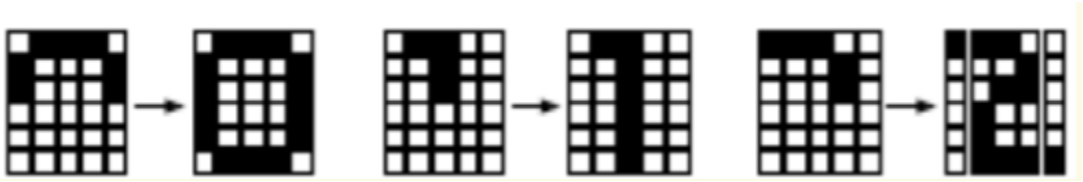
X_i = l'entrée du poids i pour la sortie attendue Y_t

W_i = le poids i actuel

Un neurone : le perceptron

- Exemple : reconstruction d'image
- Reconnaitre les chiffres 0, 1, 2

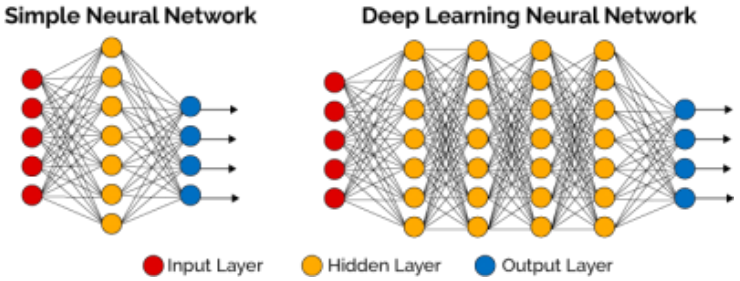


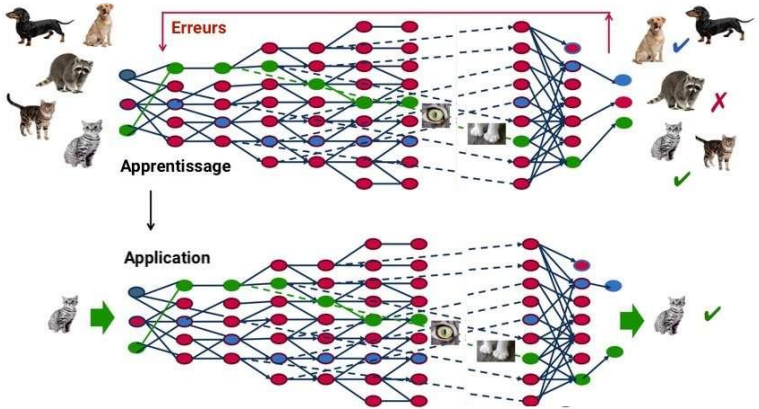


http://master-ivi.univ-lille1.fr/fichiers/Cours/rdf-semaine-8-neurones.pdf

Les réseaux de neurones : deep learning

- Réseau multi-couche

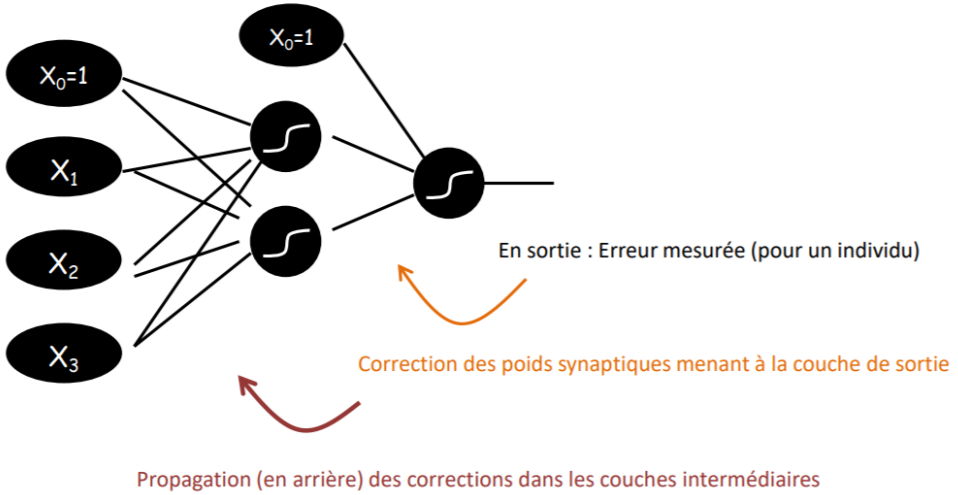




https://cdn.futura-sciences.com/buildsv6/images/mediumoriginal/d/c/d/dcdc8d74ca_125717_deep-learning.jpg

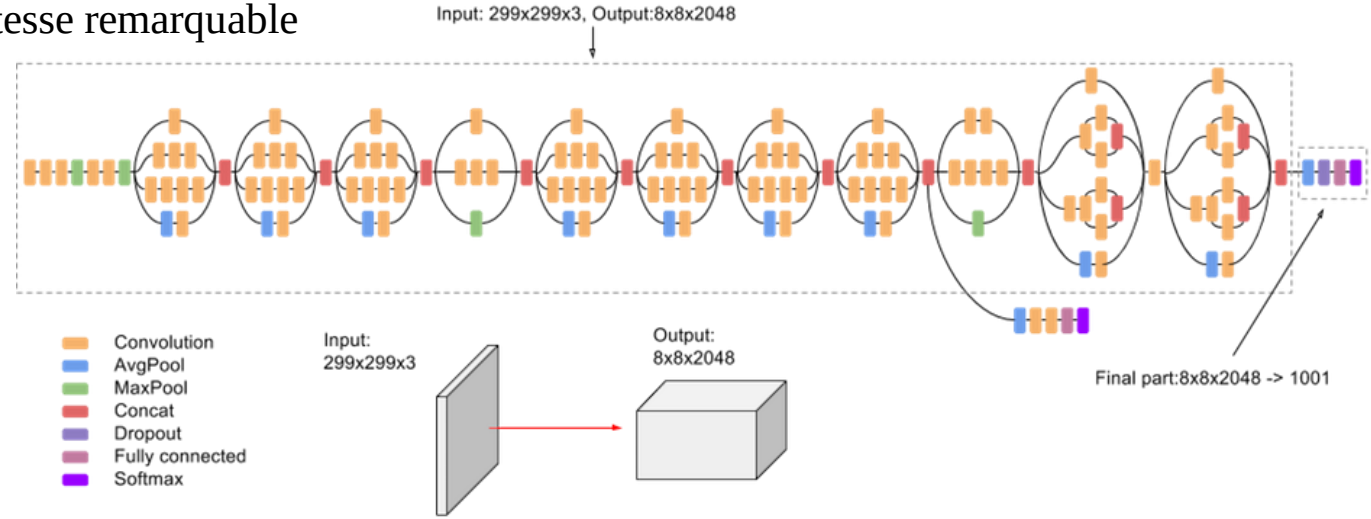
Les réseaux de neurones : deep learning

- La rétropropagation du gradient (backpropagation)
- Généraliser la règle de Widrow-Hoff – Rétropropagation



Les réseaux de neurones : deep learning

- Inception v3 sur Cloud TPU
- modèle de reconnaissance d'images
- Atteint une justesse remarquable



https://cloud.google.com/tpu/docs/images/inceptionv3onc--oview.png

TP : Python

Questions ?

Merci

fppt.com