

# Machine Learning, méthodes et solutions

(avec Python)

---

GIRAUD François-Marie



<https://www.orsys.fr/>

# **Machine Learning, méthodes et solutions**

Introduction à la Modélisation de Données

---

**Nom** Giraud François-Marie

**Courriel** giraud.francois@gmail.com

**Activité** Consultant/Formateur indépendant

**Spécialité** Intelligence Artificielle

**Parcours** Master Intelligence Artificielle et Décision (Paris 6)

Cette formation présente les **fondamentaux** de la **Modélisation Statistique** à travers des travaux pratiques.

# Votre formation — connaissances préalables

- Connaissances de base en statistiques
- Connaissances de base en python
- Avoir un **compte Google** afin de pouvoir faire les TPs dans [Google colaboratory](#)

# Votre formation — objectifs à atteindre

- Comprendre les différents modèles d'apprentissage
- Modéliser un problème pratique par le machine learning
- Identifier les méthodes adaptées à un problème donnée
- Évaluer la performance de la solution
- Se familiariser avec les librairies scientifiques python (NumPy, seaborn, sikit-learn, ... )

- 4 jours de 9h à 12h30 et de 14h à 17h30
- Le dernier jour on finit à 15h30
- À 15h on commence à remplir les documents administratifs.

- Introduction à la modélisation
- Evaluation de modèles prédictifs
- Les algorithmes supervisés/non-supervisés
- Réduction de dimension
- Données séquentielles (temporelles)



- Je vous ferai parvenir les ressources utilisées à chaque début de cours.
- Elles sont aussi accessibles via [https ://myorsys.orsys.fr/](https://myorsys.orsys.fr/)

## Tour de table — présentez-vous

- Votre nom
- Votre métier
- Vos compétences dans les domaines liés à cette formation
- Vos objectifs et vos attentes vis-à-vis de cette formation

# **Machine Learning, méthodes et solutions**

Introduction à Python

---

# Introduction à Python

Python est créé en 1989 par Guido Van Rossum.

En 2001 création de la Python Software Foundation.

Python est sous licence GPL depuis 2001.

Python 3 depuis 2009



- Langage **interprété**
  - Compilé à la volée
- Orienté **Objet**
  - Paradigme objet (mais pas que)
- **Portable**
  - Compatible avec toutes les plateformes actuelles
- Un couteau suisse **puissant** et **populaire**
  - À chaque besoin, une librairie
  - Les librairies importés sont compilées en C/C++

# Introduction à Python

## Atouts

- Stable
- multi-plateforme
- Facile à apprendre
- Grande communauté (le plus utilisé depuis 2019)
- un besoin, un module

## Inconvénients

- Non-compilé
  - Plus lent qu'un langage bas-niveau
  - Optimiser une opération  $\Rightarrow$  pas facile à apprendre

Différents interpréteurs :

- Python/CPython  $\Rightarrow$  C
- Jython  $\Rightarrow$  Java
- IronPython  $\Rightarrow$  .Net

Domaines d'applications :

- Web (Django ,Flask, ...)
- Sciences (Data mining, Machine learning, Physique, ...)
- OS (Linux, Raspberry, Script administration système, ...)
- Éducation (Initiation à la programmation)
- CAO 3D (FreeCAD, pythonCAD, ...)
- Multimédia (Kodi, ...)



# Introduction à Python

Syntaxe à typage dynamique sans délimiteurs de blocs :

```
1  a="une chaine de caracteres"  
2  b=a  
3  a=8  
4  
5  if a > 5:  
6      print('a=',a,'; b=',b)  
7  else:  
8      print("c'est étrange")
```

```
1  a= 8 ; b= une chaine de caracteres
```

# **Machine Learning, méthodes et solutions**

Notebook Jupyter

---

# Notebook Jupyter

Jupyter est un environnement de développement via une interface web.  
Plus de 40 langages de programmation sont supportés, dont le python.



Une démonstration vaut mieux qu'un long discours :  
[Jupyter Notebook Demo](#)

# **Machine Learning, méthodes et solutions**

Installation de Anaconda

---

Anaconda est une distribution Python faite pour la “Data Science”



# Installation de Anaconda

Ce qui sera installé après ce tutoriel :

- Python
- Jupyter Notebook
- Des librairies de “Data Science” :
  - SciPy, Numpy
  - Pandas, seaborn
  - scikit-learn, statsmodels
  - Matplotlib

[Installation d'Anaconda](#)



# Installation de Anaconda

Une fois installé, lancez le navigateur Anaconda puis cliquez sur “Jupyter”.

Chargez le fichier “Anaconda.ipynb” que vous trouverez dans le dossier “ressources/”.

Editez et Exécutez les cellules pour prendre en main cet environnement de développement.

# **Machine Learning, méthodes et solutions**

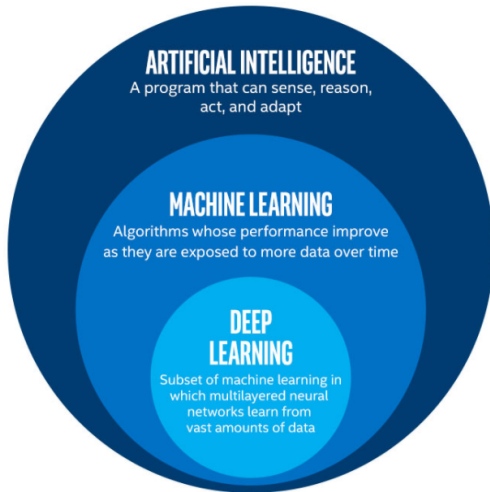
Machine Learning

---

# Machine Learning



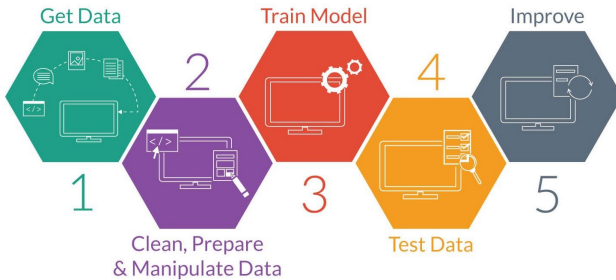
# Machine Learning



Nouvelle manière d'aborder la **conception logicielle**.

*Programmation Implicite  $\neq$  Programmation Explicite*

# Machine Learning



Définition du besoin :

Apprentissage **supervisé** ou **non-supervisé** ?

## **Apprentissage non-supervisé**

Faire émerger des profils, des groupes

Ex : groupes de clients pour adapter sa stratégie marketing



## Apprentissage supervisé

**Prédire** une valeur numérique (**régression**) ou l'appartenance à une classe (**Classification**)

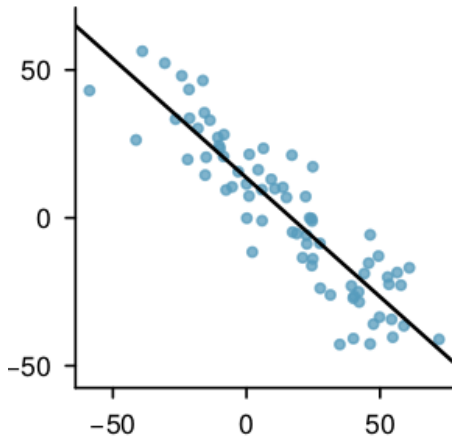
Ex (Régression) : Prédire le poids d'un individu en fonction de l'âge, la taille et le sexe.

Ex (Classification) : Prédire si une image est un chat ou un chien.

# Machine Learning

Régression Linéaire :

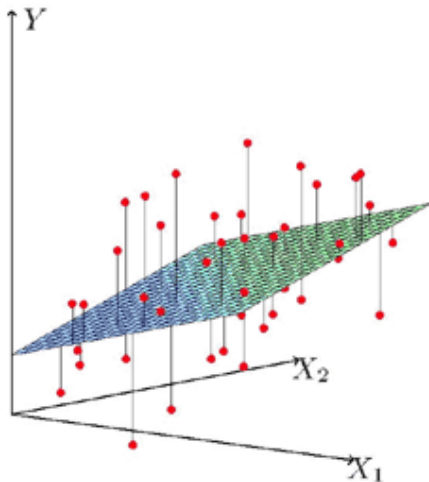
$$Y = a * X + b$$

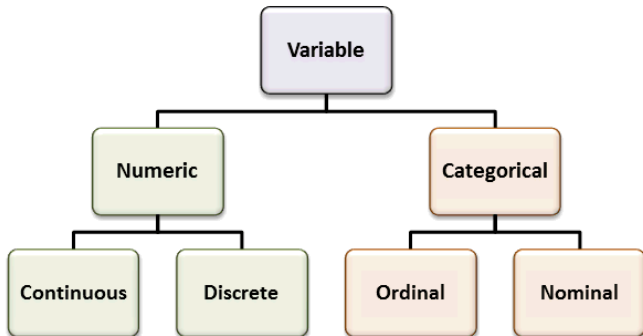


# Machine Learning

Régression Linéaire pour des données en plusieurs dimensions :

$$Y = a * X_1 + b * X_2 + c$$





## Régression Logistique (Classification)

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	1	4	1	3	1	8	0	8
0	1	0	5	5	5	7	8	4	3
0	1	0	4	3	3	1	9	1	8
0	0	6	8	5	4	1	8	1	2
0	1	2	9	5	0	2	8	8	5

# Machine Learning

## Régression Logistique (Classification)



[001.ak47](#)



[002.american-flag](#)



[003.backpack](#)



[004.baseball-bat](#)



[005.baseball-glove](#)



[006.basketball-hoop](#)



[007.bat](#)



[008.bathtub](#)



[009.bear](#)



[010.beer-mug](#)



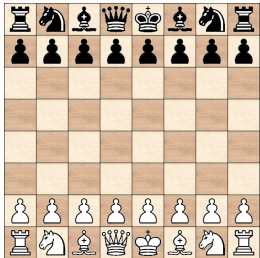
[011.billiards](#)



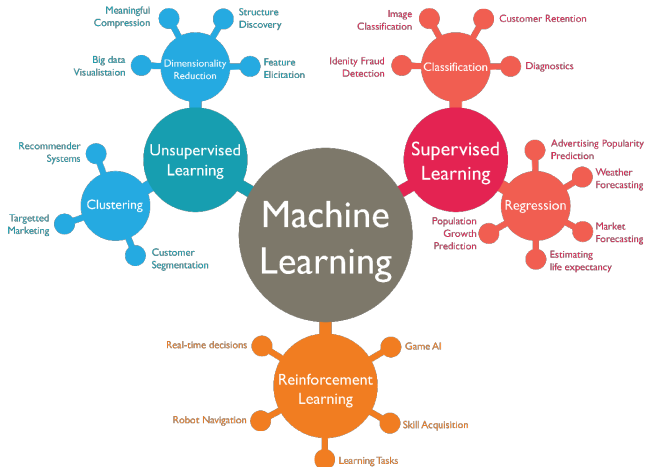
[012.binoculars](#)

## Apprentissage par Renforcement

Apprendre une **stratégie** efficace dans un **univers** où les **actions** fournissent des **récompenses** (possiblement négatives)



# Machine Learning





# **Machine Learning, méthodes et solutions**

Data Mining

---

# Data Mining



Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection
  - mesure

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection
  - mesure
  - attrition

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection
  - mesure
  - attrition
  - ...



Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection
  - mesure
  - attrition
  - ...
- trouver de fausses variables explicatives

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection
  - mesure
  - attrition
  - ...
- trouver de fausses variables explicatives

Attention aux différents biais de vos données !

- **variables confondantes** (Ex : “obésité” dans la corrélation entre “conso. viande” et “cancer colon”)
- **biais statistiques**
  - sélection, autosélection
  - mesure
  - attrition
  - ...
- trouver de fausses variables explicatives

→ Le garder en tête pendant toute l'étude.

Meilleures données > Meilleurs modèles  
(trash-in, trash-out)

→ À garder en tête pendant toute l'étude, en particulier durant l'entraînement de modèles

- valeurs manquantes
- preprocessing (texte, image)
- standardisation
- transformation

Gênant pour certains modèles. Plusieurs options :

- supprimer les enregistrements

Gênant pour certains modèles. Plusieurs options :

- supprimer les enregistrements
- remplacer par une valeur (imputation) :

Gênant pour certains modèles. Plusieurs options :

- supprimer les enregistrements
- remplacer par une valeur (imputation) :
  - constante



Gênant pour certains modèles. Plusieurs options :

- supprimer les enregistrements
- remplacer par une valeur (imputation) :
  - constante
  - moyenne de la colonne

Gênant pour certains modèles. Plusieurs options :

- supprimer les enregistrements
- remplacer par une valeur (imputation) :
  - constante
  - moyenne de la colonne
  - prédiction d'un autre modèle

- tokenizer, POS-tagger le texte (<https://spacy.io/>)
- utiliser un réseau de neurones préentraîné sur les images (<https://keras.io/applications/>)
- appliquer une transformée de fourier sur le son
- ...

Beaucoup de modèles travaillent mieux avec des données normales et sont plus efficaces autour de  $[-5, 5]$  :

- centrer sur la moyenne puis diviser par l'écart-type
- transformation de Box-Cox en cas d'asymétrie
- transformations spécifiques en fonction de la distribution

Quand un modèle n'accepte pas de données catégorielles :

- label encoding si ordinal
- one-hot encoding sinon

Si les données sont ordinales :

Ordinal :

Température
Froid
Froid
Tiède
Chaud
Tiède

Label encoding :

Température
1
1
2
3
2

## Préparation des données — one-hot encoding

Remplacer une feature par  $n$  features avec  $n$  le nombre de catégories.

Catégoriel :

Couleur
Rouge
Rouge
Jaune
Vert
Jaune

One-hot :

Rouge	Jaune	Vert
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

But :

- se rendre compte des prétraitements à effectuer (Box-Cox, imputations, etc)



But :

- se rendre compte des prétraitements à effectuer (Box-Cox, imputations, etc)
- comprendre la variable de sortie : distribution, équilibre des classes, features les plus corrélées, ...

But :

- se rendre compte des prétraitements à effectuer (Box-Cox, imputations, etc)
- comprendre la variable de sortie : distribution, équilibre des classes, features les plus corrélées, ...
- détecter les corrélations

But :

- se rendre compte des prétraitements à effectuer (Box-Cox, imputations, etc)
- comprendre la variable de sortie : distribution, équilibre des classes, features les plus corrélées, ...
- détecter les corrélations
- appréhender la complexité nécessaire du modèle

But :

- se rendre compte des prétraitements à effectuer (Box-Cox, imputations, etc)
- comprendre la variable de sortie : distribution, équilibre des classes, features les plus corrélées, ...
- détecter les corrélations
- appréhender la complexité nécessaire du modèle

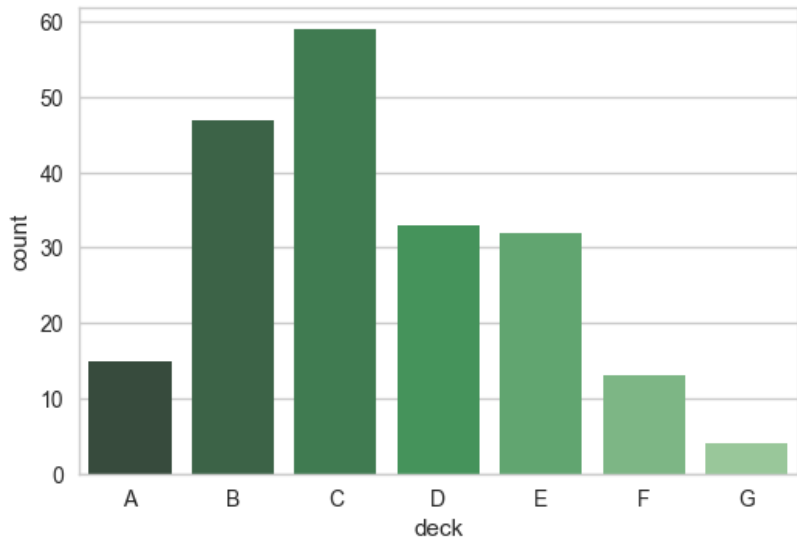
Attention : garder des données de côté (test set) et ne pas les regarder.  
**Sinon biais statistique énorme.**

Plusieurs outils sont disponibles pour explorer des données. On utilise principalement des plots pour :

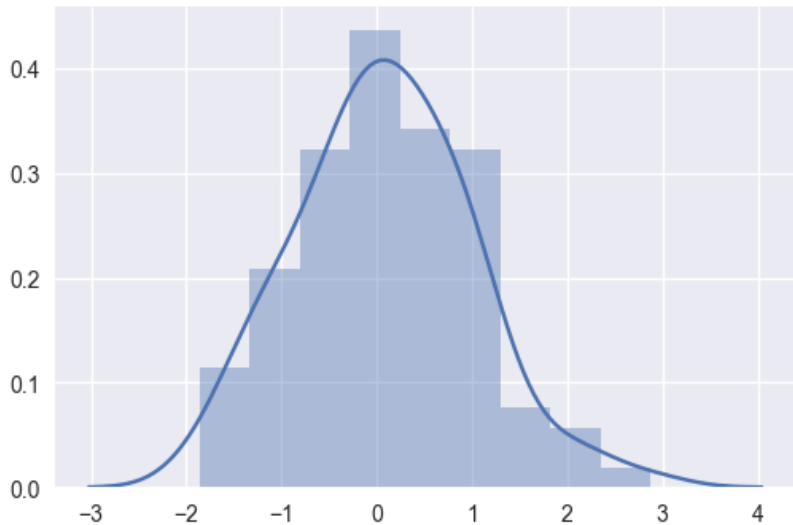
- se renseigner sur une distribution
- se renseigner sur la corrélation de deux distributions
- visualiser des corrélations linéaires

Les outils suivants sont sauf mention contraire présents dans seaborn.

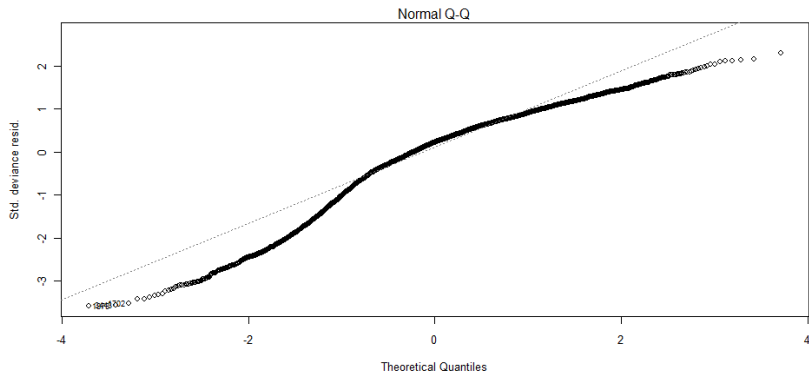
## Outils — count plot



## Outils — dist plot



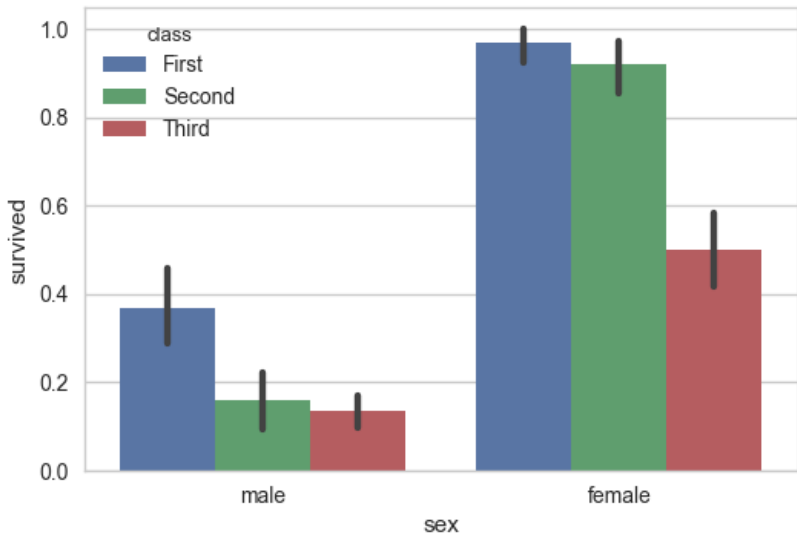
## Outils — qq plot



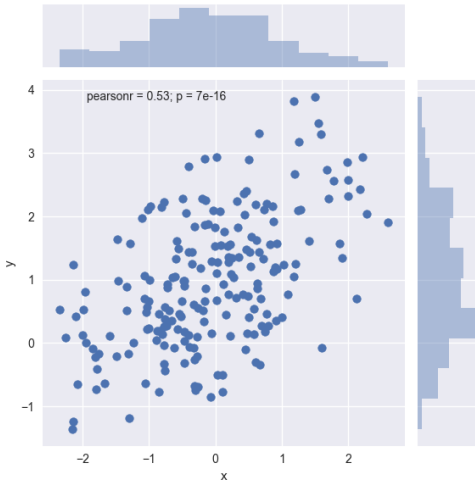
Attention, pas seaborn mais statsmodel ou scipy.stats.



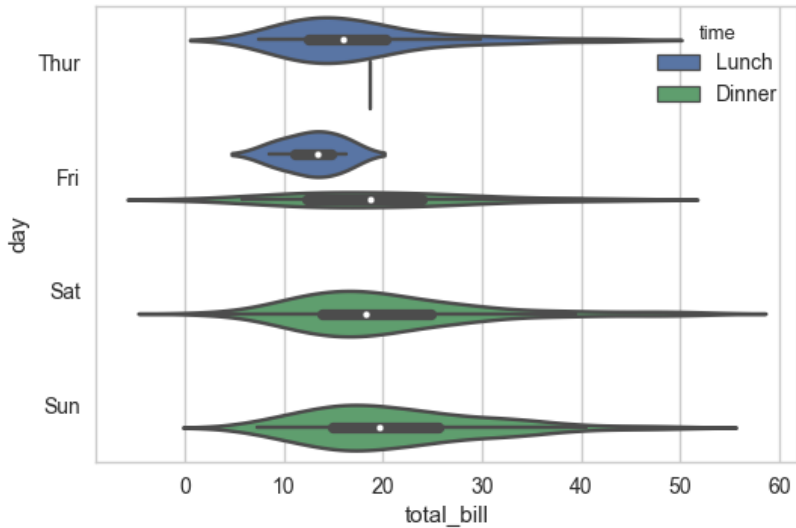
## Outils — bar plot



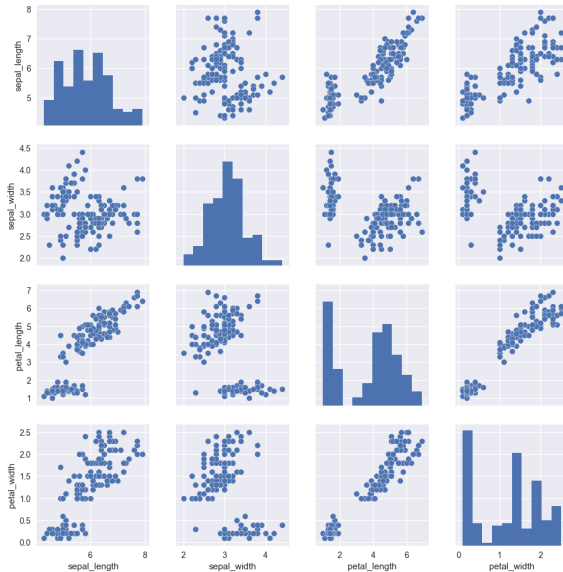
# Outils — scatter plot



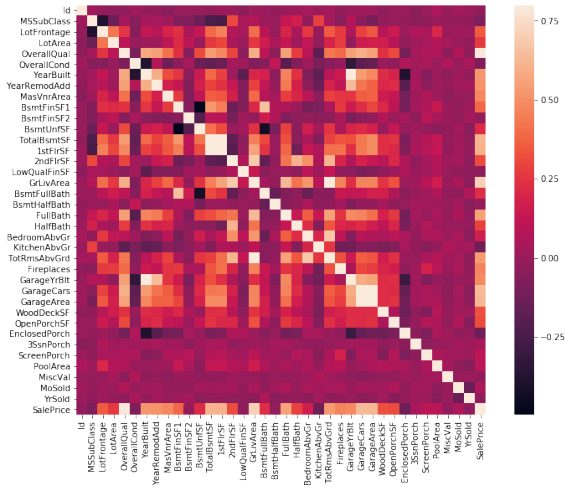
## Outils — violin plot



# Outils — pair plot



# Outils — correlation matrix



Bonnes pratiques pour explorer un dataset :

- analyser la(es) variable(s) de sortie (countplot/distplot)

Bonnes pratiques pour explorer un dataset :

- analyser la(es) variable(s) de sortie (countplot/distplot)
- trouver les corrélations linéaires les plus fortes

Bonnes pratiques pour explorer un dataset :

- analyser la(es) variable(s) de sortie (countplot/distplot)
- trouver les corrélations linéaires les plus fortes
- analyser les variables correspondantes



Bonnes pratiques pour explorer un dataset :

- analyser la(es) variable(s) de sortie (countplot/distplot)
- trouver les corrélations linéaires les plus fortes
- analyser les variables correspondantes
- regarder s'il y a des outliers évidents dans ces variables

# **Machine Learning, méthodes et solutions**

Évaluation et Apprentissage de modèles

---

En classification :

**Précision**

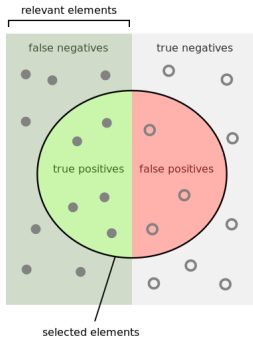
$$\frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux positifs}}$$

**Rappel**

$$\frac{\text{vrais positifs}}{\text{vrais positifs} + \text{faux négatifs}}$$

**F-mesure** moyenne harmonique entre précision et rappel (aussi appelée F1 score)

# Évaluation — outils — précision, rappel



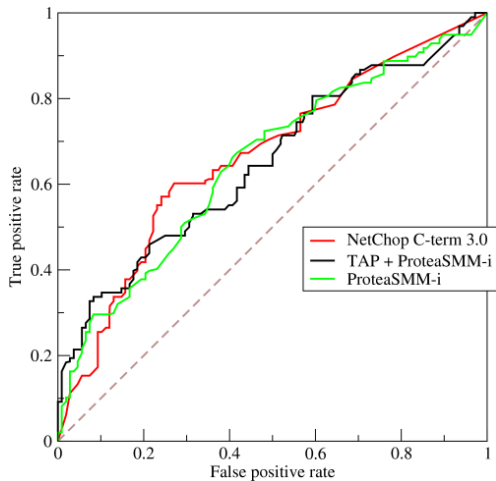
How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

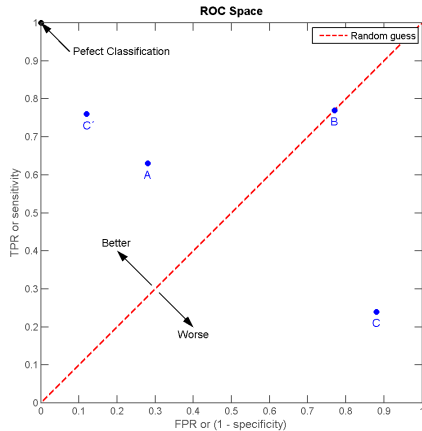
How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

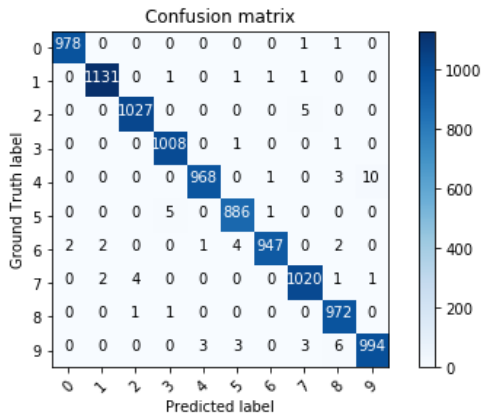
## Outils — courbe ROC



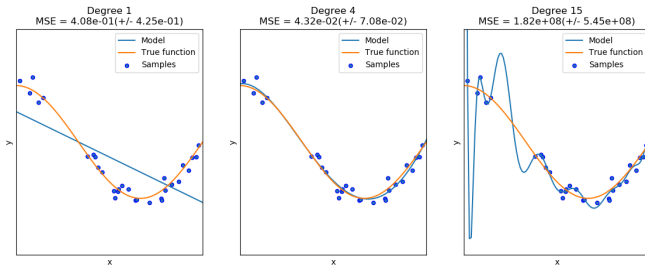
# Outils — courbe ROC



# Évaluation — outils — matrice de confusion



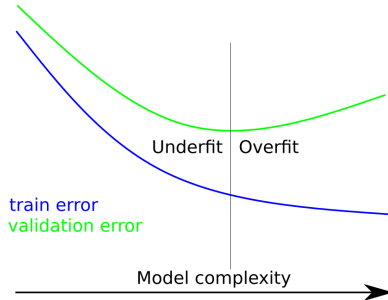
## Entraînement supervisé d'un modèle — overfit



Problème : trop minimiser la perte n'est pas bon !



# Apprentissage de modèles



→ Minimiser la perte sur un ensemble de validation

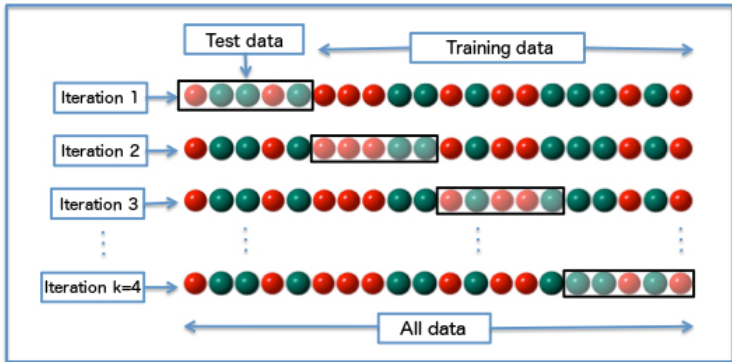
Séparation des données :

- ensemble d'entraînement
- ensemble de validation pour mesurer la généralisation
- ensemble de test (pour éviter le biais statistique)

→ Split 60/20/20 habituel.

# Apprentissage de modèles

Idéalement : Cross Validation Pour « perdre » moins de données et mieux tester la généralisation, cross-validation :



Ici, 4-fold cross-validation.

# Apprentissage de modèles

1 - initialisation aléatoire du modèle

2 - Tant que(critère arrêt == 0)

- Selection aléatoire d'un **batch** de données
- **Forward** : Passe avant du **batch** dans le modèle
- Calcul de l'erreur par rapport aux sorties attendues
- **Backward** : Rétropropagation du gradient de l'erreur en fonction des paramètres dans le modèle (mise à jour du modèle)
- Calcul critère arrêt

3 - Calcul de l'erreur sur un échantillon de données **qui n'ont JAMAIS été vues par le modèle pendant l'apprentissage !**

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon
  - recommandations :  $y_i = \text{liste d'items } x_{k \neq i}$



Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon
  - recommandations :  $y_i = \text{liste d'items } x_{k \neq i}$
  - réduction de dimensionnalité :  $y_i = x_i$  projeté dans moins de features

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon
  - recommandations :  $y_i = \text{liste d'items } x_{k \neq i}$
  - réduction de dimensionnalité :  $y_i = x_i$  projeté dans moins de features
- on définit quand même une perte (loss)

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon
  - recommandations :  $y_i = \text{liste d'items } x_{k \neq i}$
  - réduction de dimensionnalité :  $y_i = x_i$  projeté dans moins de features
- on définit quand même une perte (loss)

Densité intra- et inter-clusters en clustering

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon
  - recommandations :  $y_i = \text{liste d'items } x_{k \neq i}$
  - réduction de dimensionnalité :  $y_i = x_i$  projeté dans moins de features
- on définit quand même une perte (loss)

Densité intra- et inter-clusters en clustering

Ou utiliser des données supervisées...

Étant donné des exemples  $x_i$ , trouver un modèle  $h$  :

- but moins défini qu'en supervisé :
  - clustering :  $h(x_i) = \hat{y}_i = \text{cluster de } x_i$
  - détection d'anomalies :  $y_i = 1$  si anomalie, 0 sinon
  - recommandations :  $y_i = \text{liste d'items } x_{k \neq i}$
  - réduction de dimensionnalité :  $y_i = x_i$  projeté dans moins de features
- on définit quand même une perte (loss)

Densité intra- et inter-clusters en clustering

Ou utiliser des données supervisées... Pourcentage d'info perdue pour la réduction de dimensionnalité

- extrêmement importante pour compléter les analyses après les retours business
- ensemble de bonnes pratiques

- garder une trace exacte du preprocessing

- garder une trace exacte du preprocessing
- de préférence utiliser des notebooks



- garder une trace exacte du preprocessing
- de préférence utiliser des notebooks
- faire attention au random (utiliser des seeds)

- garder une trace exacte du preprocessing
- de préférence utiliser des notebooks
- faire attention au random (utiliser des seeds)
- définir les datasets utilisés, dates comprises

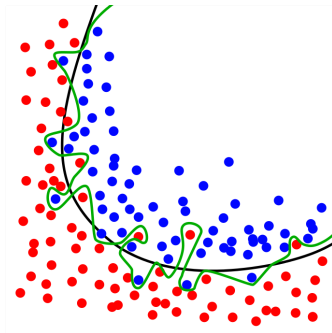
- garder une trace exacte du preprocessing
- de préférence utiliser des notebooks
- faire attention au random (utiliser des seeds)
- définir les datasets utilisés, dates comprises
- garder une trace de l'environnement

# Méta-paramètres et Régularisation

Les méta-paramètres forment l'ensemble des prétraitements, la forme et les contraintes appliquées au modèle **AVANT** son apprentissage.

- Forme : Nombre de couches?, de quelle taille? ...
- L'algorithme d'optimisation (SGD, adaboost, adam,...)
- Méthodes de régularisation (norme des paramètres dans la loss, bruitage, dropout, ...)

Régularisation  
 $\approx$   
empêcher le surapprentissage



Avez-vous des questions ?

# **Machine Learning, méthodes et solutions**

Exploration de données

---

## Quelques fonctions utiles pour les Notebook dans Colaboratory

Après avoir ouvert le lien dans Colaboratory :

Fichier > Enregistrer une copie dans Drive...

Sinon vous ne pourrez pas éditer le notebook.

## Exploration de données-TP

[python-help](#)

[pandas-help](#)

[matplotlib-help](#)

(Mais on n'oublie pas la doc ;- )