

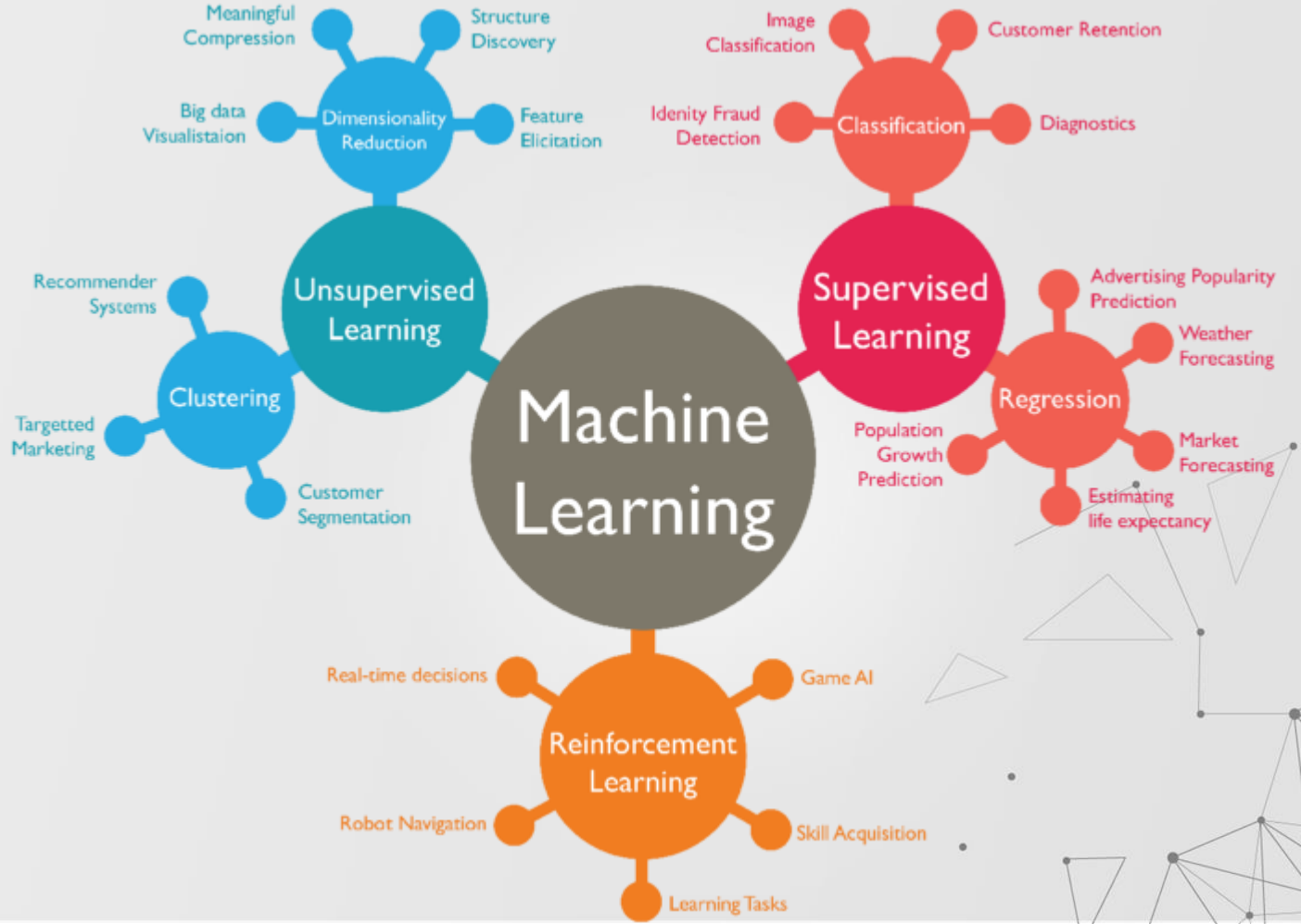
Machine Learning Pipelines with Tidymodels

Fei GAO
Mars 2024



01

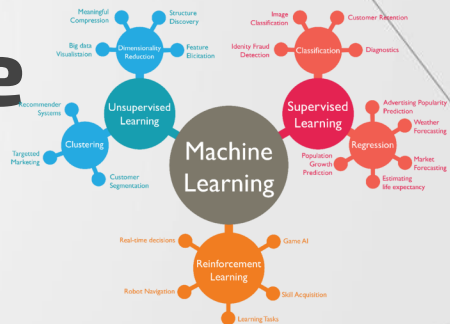
Machine Learning predictive data analysis



Données historique



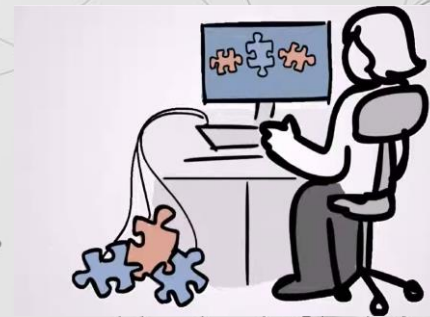
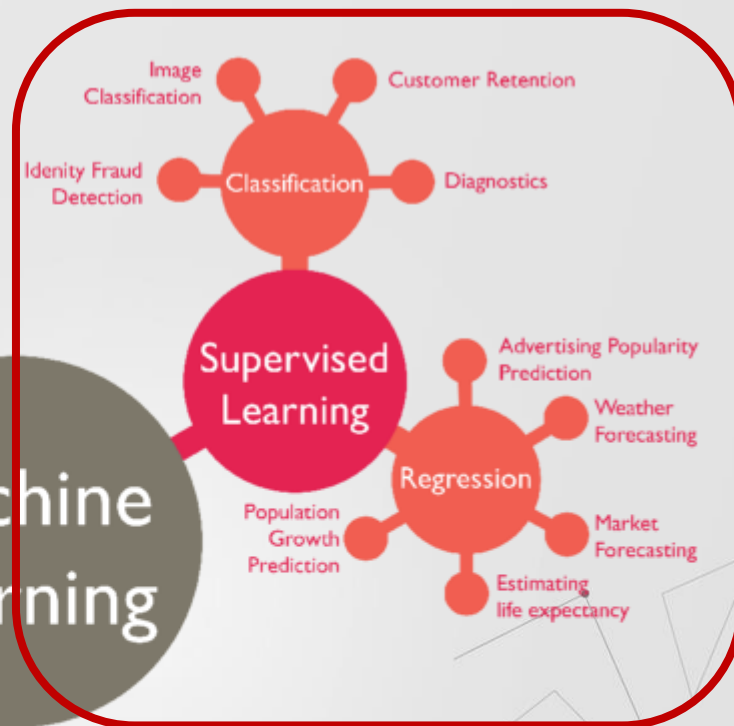
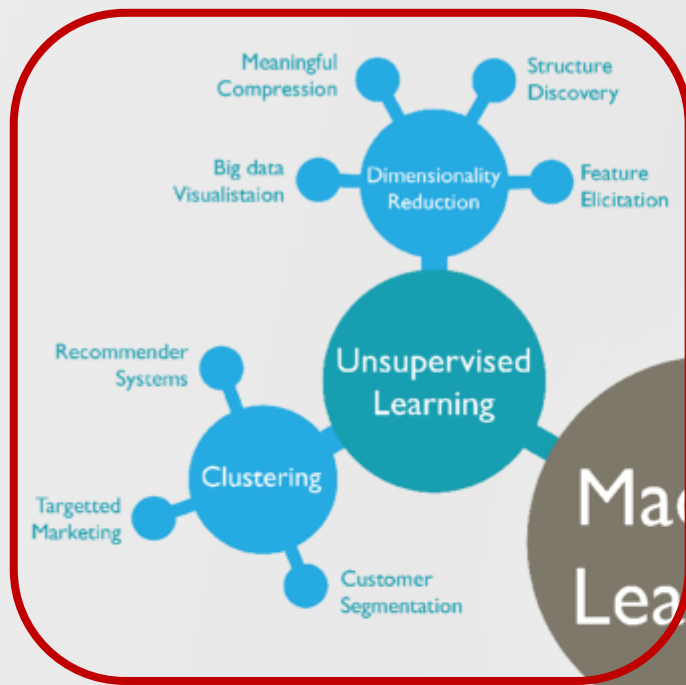
Mémorisation /
apprentissage



Nouvelles données



Généralisation /
Prédiction



APPRENTISSAGE SUPERVISÉ

Features + Label

$(X_1, X_2, X_3, X_4, \dots) (Y)$

APPRENTISSAGE NON SUPERVISÉ

Features + ~~Label~~

$(X_1, X_2, X_3, X_4, \dots) (\text{PAS d}'Y)$



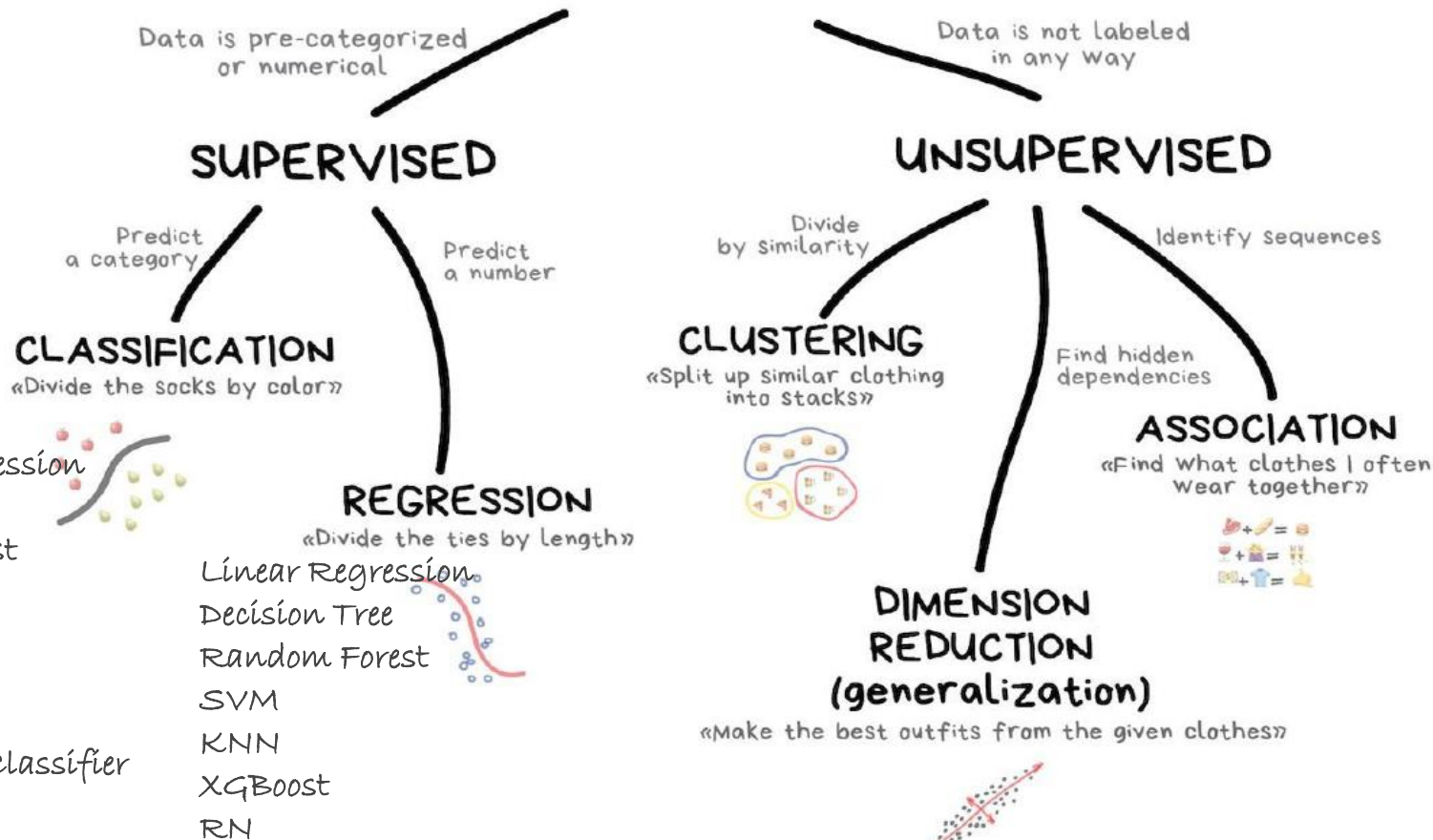
Problématique de l'analyse prédictive supervisée

- Y : label
- X_1, X_2, \dots : features
- Nous cherchons $f()$: une fonction qui essaie d'établir la relation $Y = f(X_1, X_2, \dots)$ **pour faire de la prédiction**
- $f()$ doit être aussi précise que possible pour pouvoir prédire sur les nouvelles données



Les principales familles

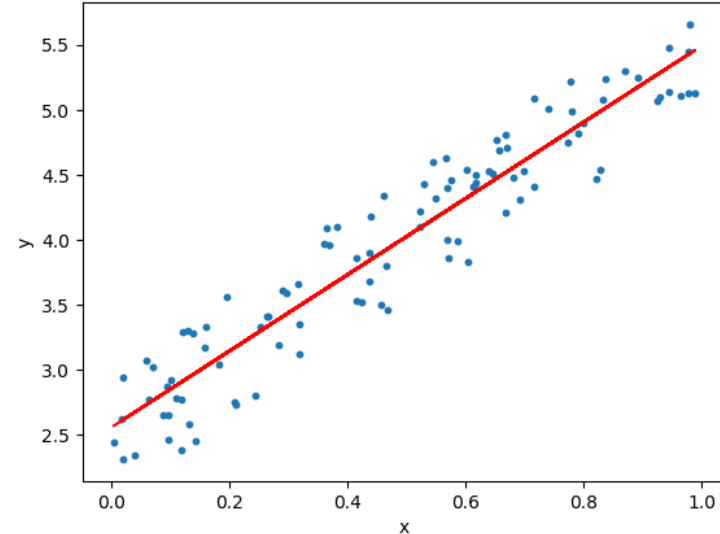
CLASSICAL MACHINE LEARNING



Usuel MACHINE LEARNING ALGORITHMS

Linear Regression

Linear Regression tends to establish a relationship between a dependent variable(Y) and one or more independent variable(X) by finding the best fit of the straight line.

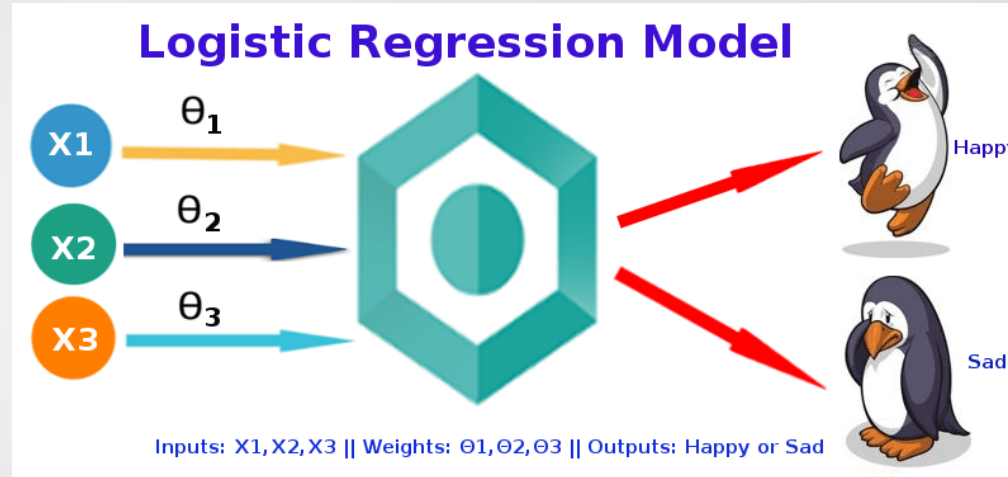


Regressor problems

Usuel MACHINE LEARNING ALGORITHMS

Logistic Regression

The logistic regression technique involves the dependent variable, which can be represented in the binary (0 or 1, true or false, yes or no) values or the probability of a successful or fail event.

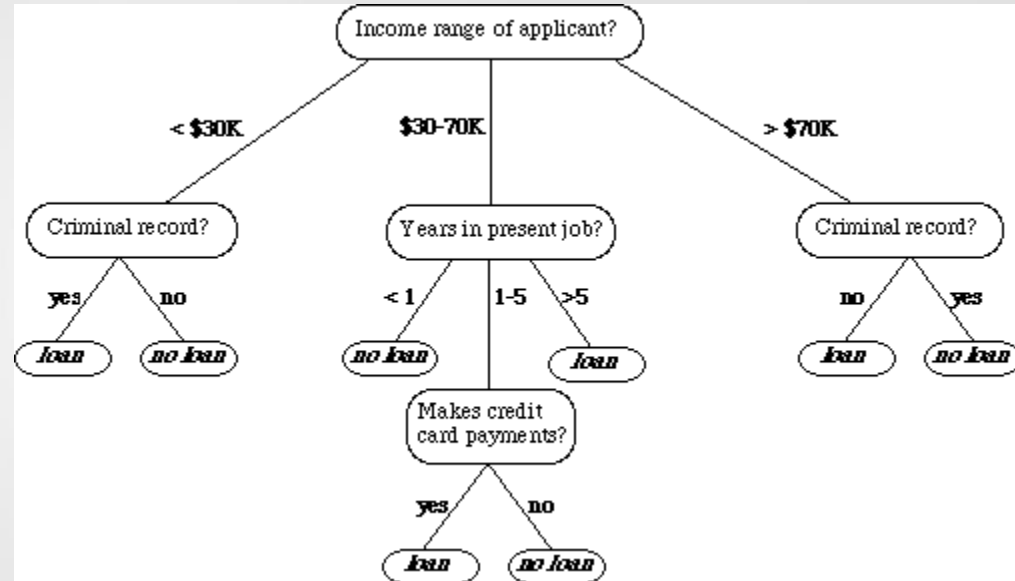


Classification problems

Usuel MACHINE LEARNING ALGORITHMS

Decision Tree

The decision tree works on an if-then statement. Decision tree tries to solve a problem by using tree representation

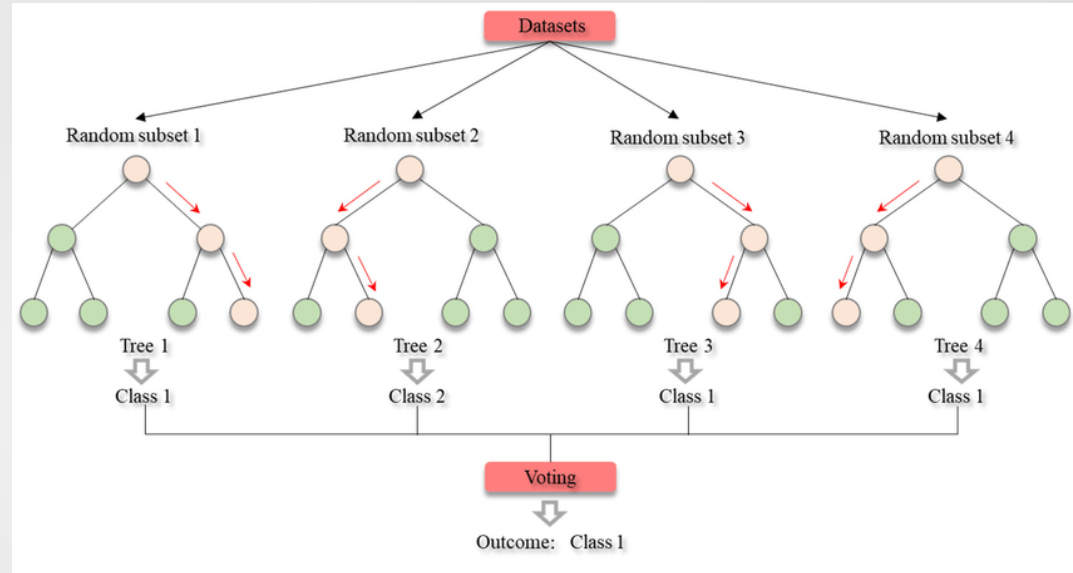


Classification as well as Regressor problems

Usuel MACHINE LEARNING ALGORITHMS

Random Forest

Random Forest is an ensemble machine learning algorithm that follows the bootstrapping technique. Random forest randomly selects a set of features that are used to decide the best split at each node of the decision tree.

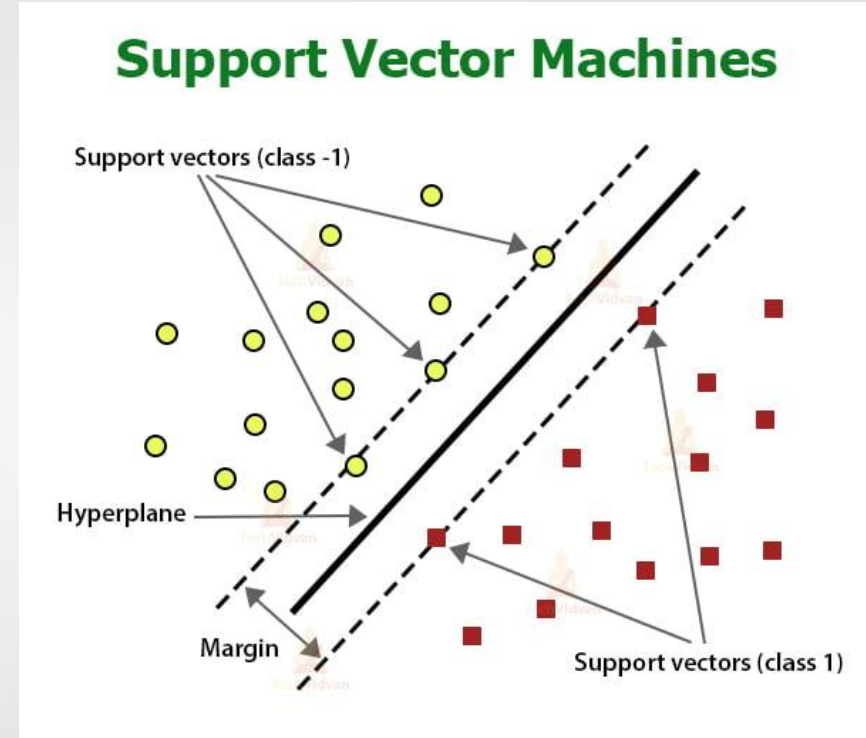


Classification as well as Regressor problems

Usuel MACHINE LEARNING ALGORITHMS

Support Vector Machine

SVM tries to find a line/hyperplane (in multidimensional space) that separates these two classes. Then it classifies the new point depending on whether it lies on the positive or negative side of the hyperplane depending on the classes to predict.



Classification as well as Regressor problems

Usuel MACHINE LEARNING ALGORITHMS

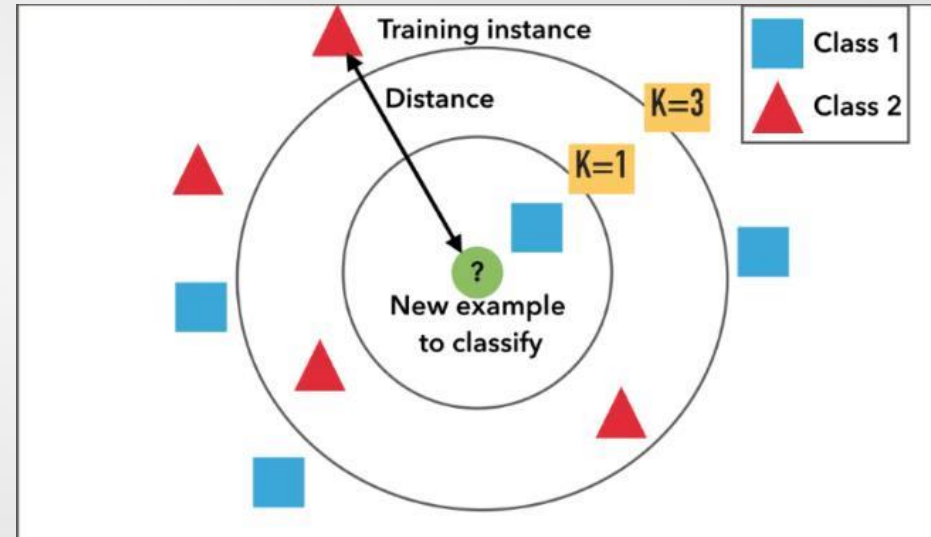
K Nearest Neighbor

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

It does not create a generalized model during the time of training.

Testing is very costly.

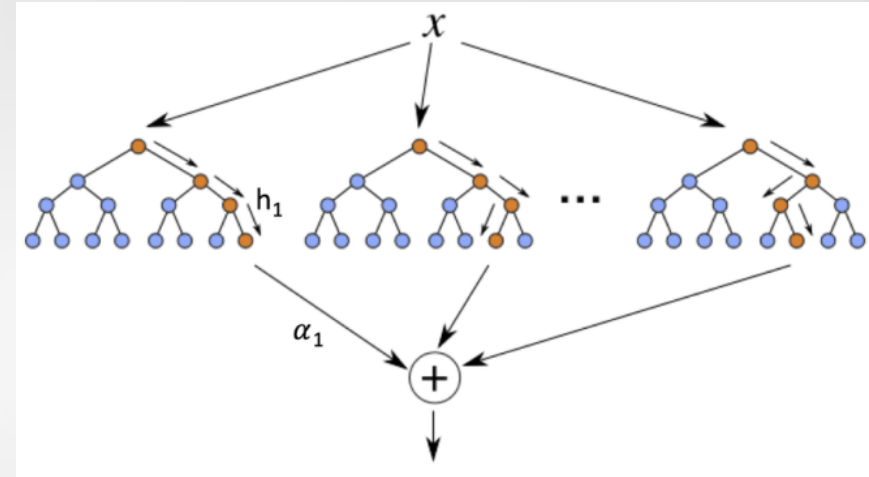
Classification as well as Regressor problems



Usuel MACHINE LEARNING ALGORITHMS

Gradient Boosting

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. The approach consists in using a gradient descent to minimize the empirical risk



Classification as well as Regressor problems

Usuel MACHINE LEARNING ALGORITHMS

Naive Bayes Classifier

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Diagram illustrating the Naive Bayes Classifier formula with labels:

- Posterior: $P(A|B)$
- Likelihood: $P(B|A)$
- Prior: $P(A)$
- Normalizing constant: $P(B)$

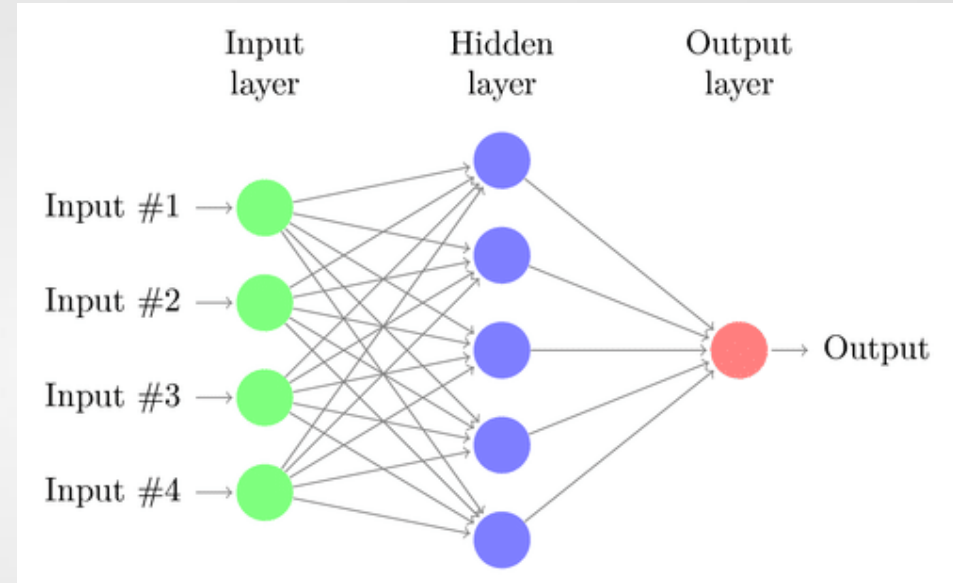
$$P(B) = \sum_Y P(B|A)P(A)$$

Classification problems

Usuel MACHINE LEARNING ALGORITHMS

Neural Network

As neural suggests, they are braininspired systems which are intended to replicate the way that we humans learn. NNs consist of input and output layers, as well as a hidden layer consisting of units that transform the input.



Classification as well as Regressor problems

Pourquoi existe-t-il autant d'algorithmes ?

Le « théorème » du « No Free Lunch »

Il n'existe pas d'algorithme qui soit le meilleur quelque soit le problème d'apprentissage...



Schéma typique (standard) de l'analyse prédictive supervisée

Y : variable cible

X1, X2, ... : variables explicatives

f(.) une fonction qui essaie d'établir la relation $Y = f(X1, X2, \dots)$

f(.) doit être « aussi précise que possible »...



Construction de la fonction f(.) à partir des données d'apprentissage

Training set

A small thumbnail image of a data table with multiple columns and rows, representing the training set.

$$Y = f(X1, X2, \dots) + \epsilon$$

Application du modèle (prédiction) sur l'ensemble de test

Validation set

A small thumbnail image of a data table with multiple columns and rows, representing the validation set.

$$(Y, \hat{Y})$$

Y : valeurs observées

\hat{Y} : valeurs prédites par f(.)

Mesures de **performances** par confrontation entre Y et \hat{Y} : matrice de confusion + mesures

Age	Sex	Smoker	Age*Sex	Smoker*Age	Smoker*Sex	Outcome
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	0	0	0	0	0	0
21	0	0	0	0	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0
24	0	0	0	0	0	0
25	0	0	0	0	0	0
26	0	0	0	0	0	0
27	0	0	0	0	0	0
28	0	0	0	0	0	0
29	0	0	0	0	0	0
30	0	0	0	0	0	0
31	0	0	0	0	0	0
32	0	0	0	0	0	0
33	0	0	0	0	0	0
34	0	0	0	0	0	0
35	0	0	0	0	0	0
36	0	0	0	0	0	0
37	0	0	0	0	0	0
38	0	0	0	0	0	0
39	0	0	0	0	0	0
40	0	0	0	0	0	0
41	0	0	0	0	0	0
42	0	0	0	0	0	0
43	0	0	0	0	0	0
44	0	0	0	0	0	0
45	0	0	0	0	0	0
46	0	0	0	0	0	0
47	0	0	0	0	0	0
48	0	0	0	0	0	0
49	0	0	0	0	0	0
50	0	0	0	0	0	0
51	0	0	0	0	0	0
52	0	0	0	0	0	0
53	0	0	0	0	0	0
54	0	0	0	0	0	0
55	0	0	0	0	0	0
56	0	0	0	0	0	0
57	0	0	0	0	0	0
58	0	0	0	0	0	0
59	0	0	0	0	0	0
60	0	0	0	0	0	0
61	0	0	0	0	0	0
62	0	0	0	0	0	0
63	0	0	0	0	0	0
64	0	0	0	0	0	0
65	0	0	0	0	0	0
66	0	0	0	0	0	0
67	0	0	0	0	0	0
68	0	0	0	0	0	0
69	0	0	0	0	0	0
70	0	0	0	0	0	0
71	0	0	0	0	0	0
72	0	0	0	0	0	0
73	0	0	0	0	0	0
74	0	0	0	0	0	0
75	0	0	0	0	0	0
76	0	0	0	0	0	0
77	0	0	0	0	0	0
78	0	0	0	0	0	0
79	0	0	0	0	0	0
80	0	0	0	0	0	0
81	0	0	0	0	0	0
82	0	0	0	0	0	0
83	0	0	0	0	0	0
84	0	0	0	0	0	0
85	0	0	0	0	0	0
86	0	0	0	0	0	0
87	0	0	0	0	0	0
88	0	0	0	0	0	0
89	0	0	0	0	0	0
90	0	0	0	0	0	0
91	0	0	0	0	0	0
92	0	0	0	0	0	0
93	0	0	0	0	0	0
94	0	0	0	0	0	0
95	0	0	0	0	0	0
96	0	0	0	0	0	0
97	0	0	0	0	0	0
98	0	0	0	0	0	0
99	0	0	0	0	0	0
100	0	0	0	0	0	0

Metrics Classification

1. Accuracy, Precision, Recall...

$$\text{Accuracy} = (TP+TN)/(TP+FP+FN+TN)$$

Precision = $(TP)/(TP+FP)$: is the proportion of all positive cases that were correctly classified => **FP intolerable**

Specificity = $TN/(TN+FP)$: is the proportion of all negative cases that were correctly classified

Sensitivity/Recall = $(TP)/(TP+FN)$ => **FN intolerable**

False positive rate (FPR) = $1 - \text{specificity} = 1 - (TN/(TN+FP))$: is the proportion of false positives among true negatives

2. F1 Score

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

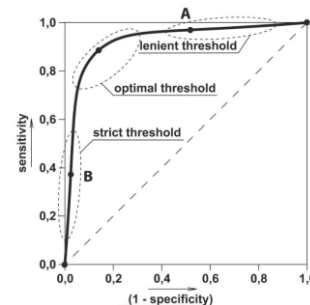
Plus d'information :

<https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226>

3. AUC

AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes => Allowing threshold selection

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative



Metrics Regression

1

MAE : Mean Average Error

Mesure de l'erreur moyenne réalisée

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

2

RMSE : Root Mean Squared Error

Mesure de l'erreur pénalisant les grandes erreurs

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

3

Root Mean Squared Logarithmic Error

Pénalisation des valeurs sous-estimant la valeur cible

$$\epsilon = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

4

Autres mesures

Mean Squared Error, Weigthed Mean Average Error, ...

Processus général d'un problème de data science

1

Définition de la cible et des variables explicatives

- Identification ou construction de la cible
- Identification des variables explicatives

2

Construction du jeu de données

- Filtres de la population concernée
- Exploration et visualisation

3

Préparation des données

- Feature Engineering : création de variables explicatives
- Traitement des valeurs manquantes
- Encodage numérique des variables catégoriques
- Séparation des données en échantillons

4

Entraînement du modèle

- Sélection du meilleur jeu de paramètres
- Entraînement du modèle

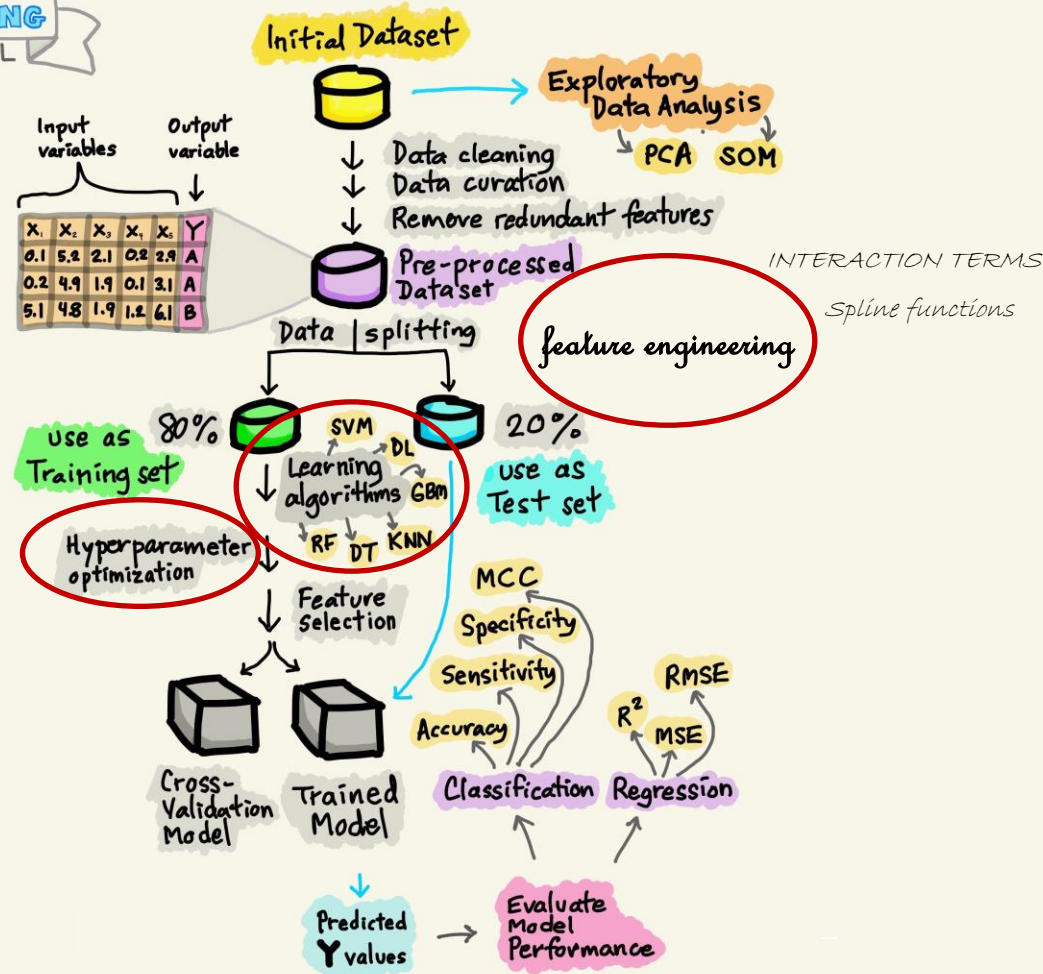


Résultats et performance du modèle

- Application du modèle sur l'échantillon de test
- Etude des caractéristiques clés du modèle

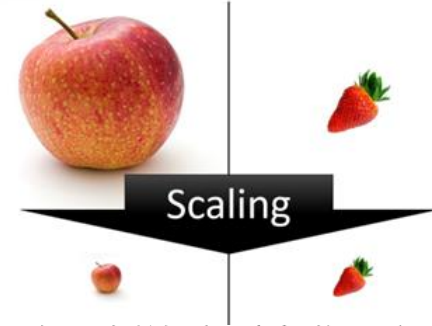
et La mise à l'échelle

BUILDING THE MACHINE LEARNING MODEL



La mise à l'échelle

Name	Weight	Price
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5



La mise à l'échelle

Sr No.	Algorithms	Feature Scaling
1.	Linear/Non-Linear Regressions	Yes
2.	Logistic Regression	Yes
3.	KNN	Yes
4.	SVM	Yes
5.	Neural Networks	Yes
6.	K-means clustering	Yes
7.	CART	No
8.	Random Forests	No
9.	Gradient Boosted Decision Trees	No
10.	Naïve Bayes	NO
11.	PCA	Yes
12.	SVD	Yes
13.	Factorization Machines	Yes

Fig: Feature Scaling requires based on Machine Learning

Schéma typique (standard) de l'analyse prédictive supervisée

Y : variable cible

X1, X2, ... : variables explicatives

f(.) une fonction qui essaie d'établir la relation $Y = f(X1, X2, \dots)$

f(.) doit être « aussi précise que possible »...



Ensemble
d'apprentissage
Training set

Construction de la fonction f(.) à
partir des données d'apprentissage

A small thumbnail image of a dataset table with multiple columns and rows of data.

$$Y = f(X1, X2, \dots) + \epsilon$$

Application du modèle (prédiction)
sur l'ensemble de test

Test set

A small thumbnail image of a dataset table, similar to the one in the training set section.

$$(Y, \hat{Y})$$

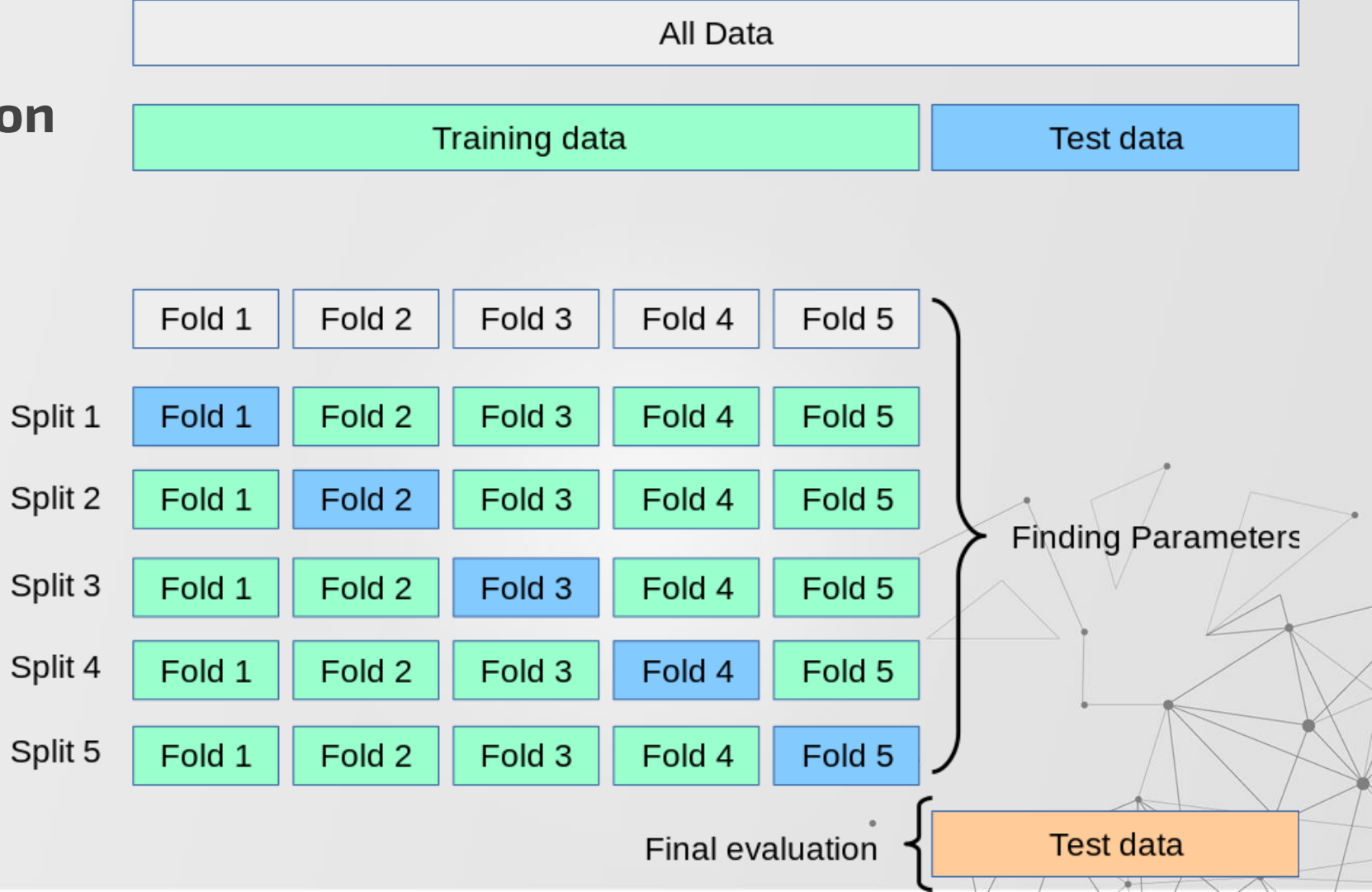
Mesures de **performances**
par confrontation entre Y
et \hat{Y} : matrice de
confusion + mesures

Y : valeurs observées
 \hat{Y} : valeurs prédites par f(.)

A small thumbnail image of a dataset table, showing a sample of data with columns and rows.

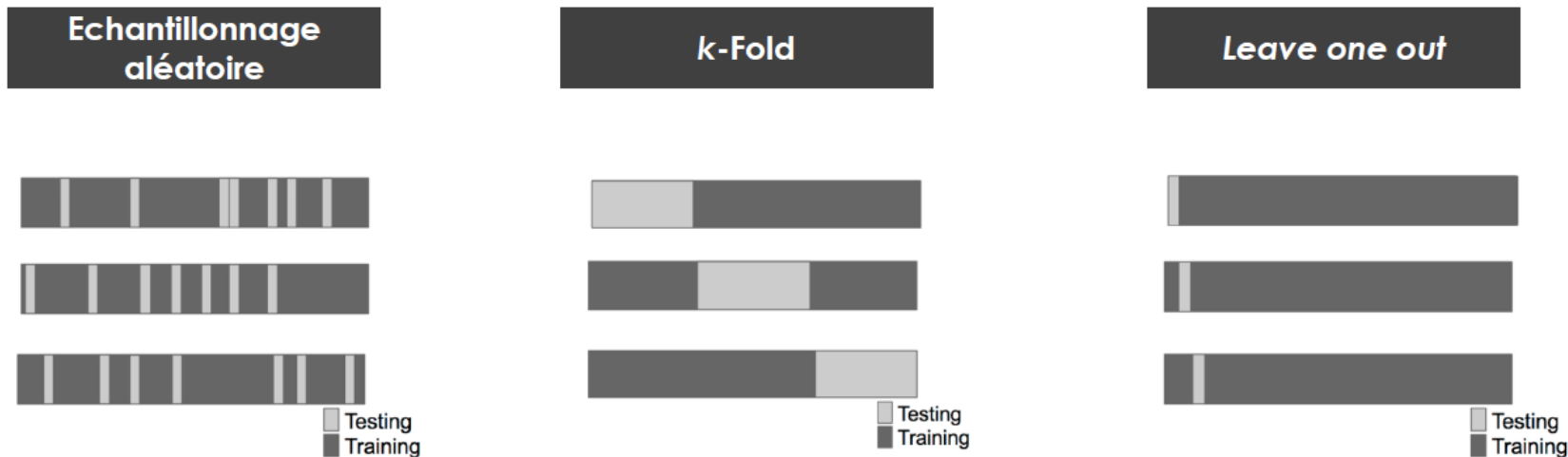
Ensemble
de données
(dataset)

Cross validation



Model cross-validation

Différentes méthodes pour s'assurer que le modèle « apprend » bien, qu'il « généralise »

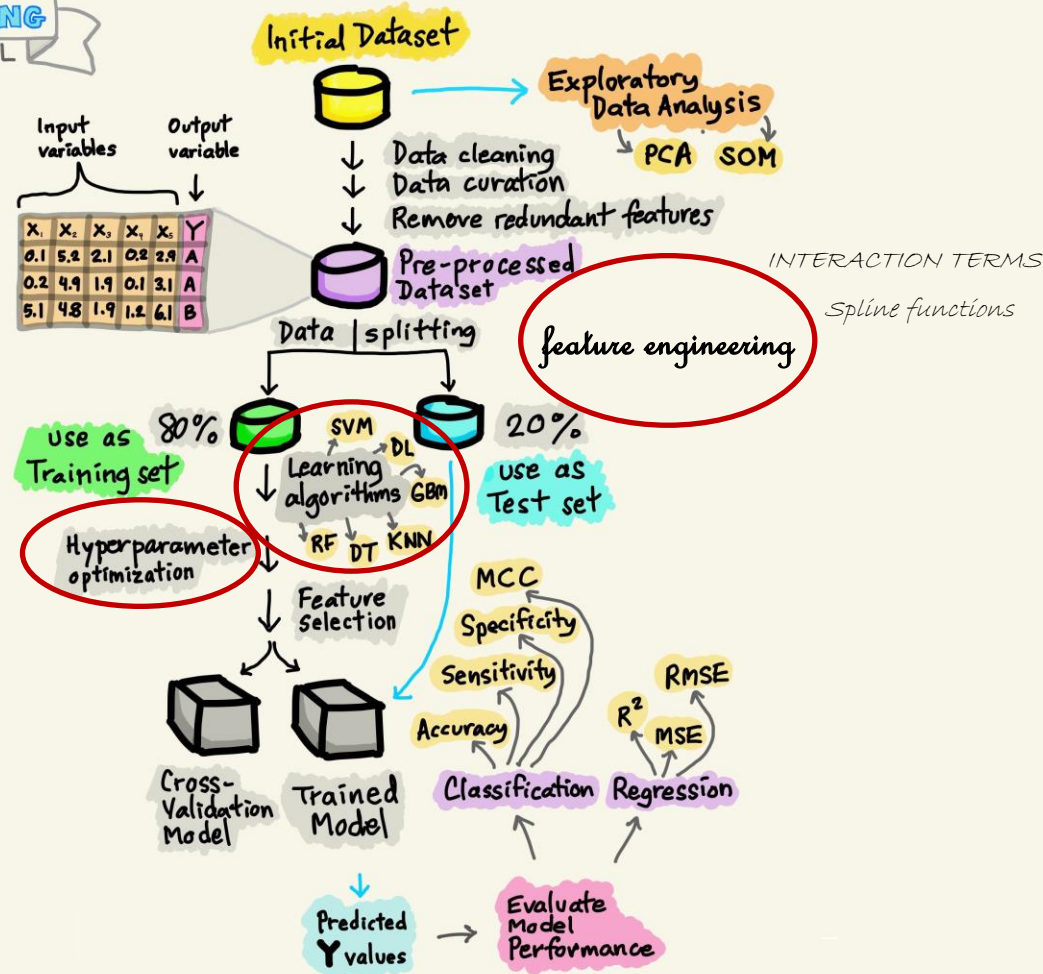


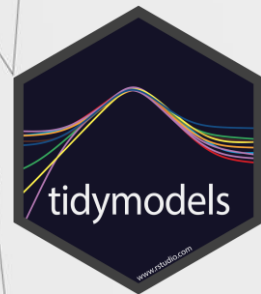
Idée de la validation croisée pour comparer la performance des modèles, indépendamment des « individus » utilisés pour apprendre (risque si l'on utilise 1 seul jeu d'apprentissage).

Habituellement, on utilise entre 5 ou 10 « folds ».



BUILDING THE MACHINE LEARNING MODEL





02

Tidymodel

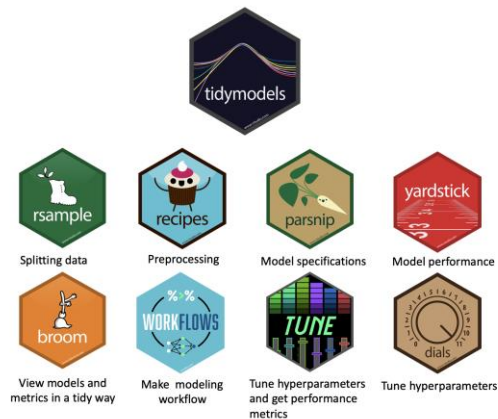
A collection of packages for
modeling and machine learning

Tidymodel



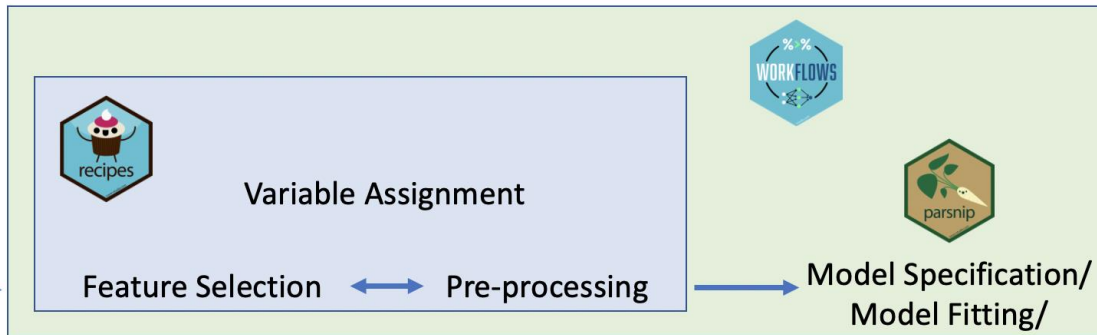
The tidymodels framework is a collection of packages for modeling and machine learning using [tidyverse](#) principles.

- rsample : Different types of re-samples
- recipes : Transformations for model data pre-processing
- parsnip : A common interface for model creation
- Yardstick and tune : Measure model performance
- combining recipe and parsnip objects into a workflow



Data Exploration

Data Splitting



Model
Performance
Evaluation

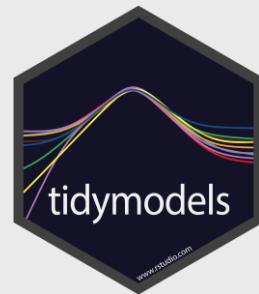


Pre-Process → Train → Validate

Other tidymodels packages include:



Pre-Process : Data Sampling



Separating Testing and Training Data

```
initial_split(df, strata = outcome variable)
```

Cross validation

```
vfold_cv(training_df, v = 3)
```

* strata : stratified random sample

Feature Engineering: recipes



recipe() : Create a recipe for preprocessing data

- * processing of missing data
- * Dummy Variables Creation (numerical encoding of categorical variables)
- * Scale your Features

steps()

prep() : Estimate a preprocessing recipe

bake() : Apply a trained preprocessing recipe

juice() : Extract transformed training set

formula(<recipe>) : Create a Formula from a Prepared Recipe

Feature Engineering: recipes



- **step_dummy()** converts characters or factors (i.e., nominal variables) into one or more numeric binary model terms for the levels of the original data.
- **step_normalize()** centers and scales numeric variables.
- **step_scale()** Scaling Numeric Data
- **step_zv()** removes indicator variables that only contain a single unique value (e.g. all zeros). This is important because, for penalized models, the predictors should be centered and scaled.

2. Mean Normalisation:

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

1. Standardisation:

Standardisation replaces the values by their Z scores.

$$x' = \frac{x - \bar{x}}{\sigma}$$

Feature Engineering: recipes



- **step_impute_mean()** : Impute numeric data using the mean
 - **step_impute_mode()** : Impute nominal data using the most common value
 - **step_impute_linear()** : Impute numeric variables via a linear model
 - **step_other()** : Collapse Some Categorical Levels
- * all_numeric(), all_nominal(), all_predictors()

Reference : <https://recipes.tidymodels.org/reference/>

More sophisticated recipes



- **`step_interact(~ X1:X2)`** : Create Interaction Variables
- **`step_ns(X, deg_free = n)`** : Natural Spline Basis Functions : Generate the B-spline basis matrix for a natural cubic spline
deg_free : The degrees of freedom for the natural spline. As the degrees of freedom for a natural spline increase, more flexible and complex curves can be generated. When a single degree of freedom is used, the result is a rescaled version of the original data.

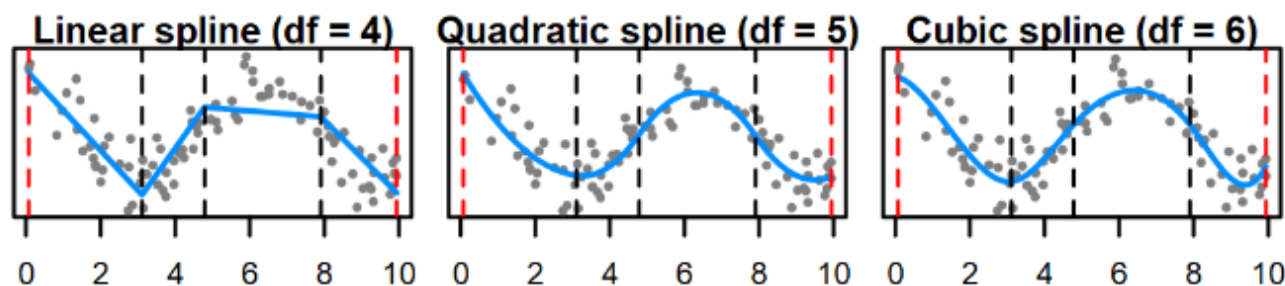
More information :

<https://stats.stackexchange.com/questions/517375/splines-relationship-of-knots-degree-and-degrees-of-freedom>

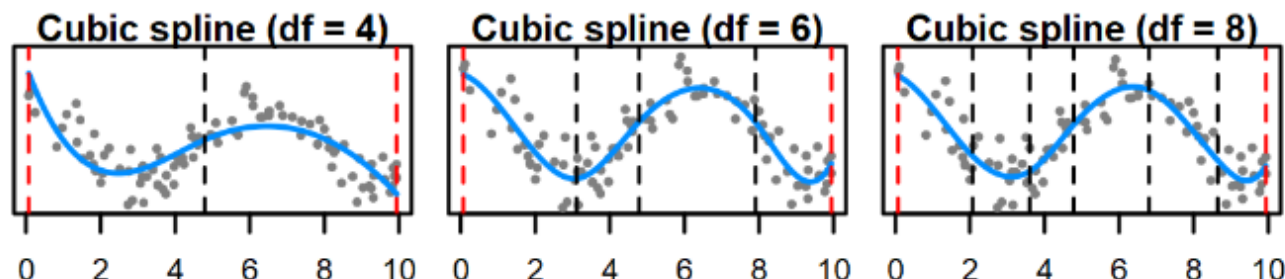
For B-splines: $df = k + \text{degree}$ if you specify the knots or $k = df - \text{degree}$ if you specify the degrees of freedom and the degree. For natural (restricted) cubic splines: $df = k - 1$ if you specify the knots or $k = df + 1$ if you specify the degrees of freedom.

As an example: A cubic spline ($\text{degree} = 3$) with 4 (internal) knots will have $df = 4 + 3 = 7$ degrees of freedom. Or: A cubic spline ($\text{degree} = 3$) with 5 degrees of freedom will have $k = 5 - 3 = 2$ knots.

Let's see some illustrations. In the scatterplots below you see some artificial data together with the spline fits of different degrees but the same amount of knots ($k = 3$). The knots are indicated by dashed vertical lines (Boundary knots by red dashed lines) and are placed at the 25th, 50th and 75th percentile of x . The first plot shows a linear spline (degree = 1), the second one a quadratic spline (degree = 2) and the third is a cubic spline with degree = 3.



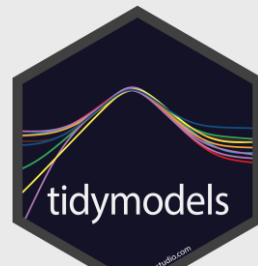
In the next plot, you see three cubic splines with different degrees of freedom. As before, the knots are shown as dashed vertical lines. With increasing degrees of freedom, the number of knots gets larger (from 1 to 3 to 5). The spline gets wigglier although the difference is only really noticeable between the first and second plot.



spline function procedures

A cubic spline function, with three knots (τ_1, τ_2, τ_3) will have 7 degrees of freedom. Using representation given in Eq. [2](#), the function can be written as:

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 (X - \tau_1)^3 + \beta_5 (X - \tau_2)^3 + \beta_6 (X - \tau_3)^3$$



Defining and Fitting Models: parsnip



- The **parsnip** package has wrappers around many popular machine learning algorithms, and you can fit them using a unified interface.
- **Philosophy is to unify & make interfaces more predictable.**
- <https://www.tidymodels.org/find/parsnip/>

parsnip



```
# model setup
lm_mod <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

# check
lm_mod
```

- Specify model type (e.g. linear regression, random forest ...)
 `linear_reg()`
 `rand_forest()`
- Specify engine (i.e. package implementation of algorithm)
 `set_engine("some package's implementation")`
- Declare mode (e.g. classification vs linear regression). *Use this when model can do both classification & regression*
 `set_mode("regression")`
 `set_mode("classification")`

Defining and Fitting Models: parsnip



- Examples :

```
log_model <-  
logistic_reg() %>%  
  set_engine("glm") %>%  
  set_mode("classification")
```

```
rf_model <-  
rand_forest(mtry = 3, trees = 500, min_n = 5) %>%  
  set_engine("ranger") %>%  
  set_mode("classification")
```

Combine Everything: workflows



The **workflows** package is designed to bundle together different parts of a machine learning pipeline like a recipe or a model.

A **workflow object** is a combination of a preprocessor (e.g. a formula or recipe) and a parsnip model specification.

Examples :

```
rf_wflow <- workflow() %>%  
  add_recipe(rec) %>%  
  add_model(rf_model)
```

```
log_fit <- fit(log_wflow, DF_trainning)
```

```
predict(log_fit, ames_test, type='prob')
```

Tuning Model and recipe Parameters



Tuning **Model** Parameters :

Examples :

rf_model <-

```
rand_forest( Trees = tune(), min_n = tune() ) %>%  
set_mode("regression") %>%  
set_engine("ranger")
```

Tuning Model and recipe Parameters










Tuning **recipe** Parameters :

Examples :

```
ames_rec <-  
  recipe(grav_or_not ~ ., data = ames_train) %>%  
  step_ns(age, deg_free = tune())
```

Overview of *tidymodels* Basics

Package	Step	Functions
	1. Split into testing and training sets	initial_split() training() testing()
	2. Create recipe + assign variable roles	recipe() update_role()
	3. Specify model, engine, and mode	parsnip function for specifying model (ex. decision_tree()) (https://www.tidymodels.org/find/parsnip/) set_engine() set_mode()
	4. Create workflow, add recipe, add model	workflow() add_recipe() add_model()
	5. Fit workflow	fit()
	6. Get predictions	predict()
	7. Use predictions to get performance metrics	rmse() (continuous outcome) accuracy() (categorical outcome) metrics() (either type of outcome)

Comparing Models



A **Workflow sets** are collections of tidymodels workflow objects that are created as a set.

A **workflow object** is a combination of a preprocessor (e.g. a formula or recipe) and a parsnip model specification.

In stead of creating a large number of individual **workflow objects**, a **cohort of workflows** can be created simultaneously.

Comparing Models

A **Workflow sets** are collections of tidymodels workflow objects that are created as a set.

```
My_workflow_set <-  
  workflow_set(  
    preproc = list(  
      basic = ames_rec,  
      interact = interaction_rec,  
      spline_tune = spline_rec_tune  
    ),  
    models = list(  
      log = log_model,  
      rf = rf_model,  
      neural_network = nnet_spec_tune  
    ),  
    cross = TRUE  
  )
```



Comparing Models



A **Workflow sets** are collections of tidymodels workflow objects that are created as a set.

```
My_workflow_set %>%
```

```
  workflow_map(
```

```
    seed = 1101,
```

```
    resamples = ames_resampling,
```

```
    verbose = TRUE,
```

```
    grid = 20,
```

```
    metrics = metric_set(accuracy, roc_auc, pr_auc),
```

```
    control = keep_pred
```

```
)
```

```
* keep_pred <- control_resamples(save_pred = TRUE, save_workflow = TRUE)
```

Comparing Models



Grid search : to optimize the tuning parameters and choose the best tuning parameter combination

grid = 20 : 20 candidate sets of all tuning parameters (model & recipes)

Selecting the Best Model



```
rank_results(My_workflow_set, rank_metric = "accuracy")
```

```
autoplot(  
  My_workflow_set,  
  rank_metric = "accuracy", # <- how to order models  
  metric = "accuracy",     # <- which metric to visualize  
  select_best = TRUE      # <- one point per workflow  
)
```

Selecting the Best Model



```
best_results <-  
  My_workflow_set %>%  
  pull_workflow_set_result("spline_tune_bt") %>%  
  select_best(metric = "accuracy")
```

```
best_results_fit <-  
  My_workflow_set %>%  
  pull_workflow("spline_tune_bt") %>%  
  finalize_workflow(best_results) %>%  
  last_fit(split = ames_split)
```

Make the Final Predictions

```
test_prediction <-  
  best_results_fit %>%  
  collect_predictions()
```

- accuracy(test_prediction, Y, .pred_class)
- roc_auc(test_prediction, Y, .pred_0,)



Detecting Interaction Effects



REF :

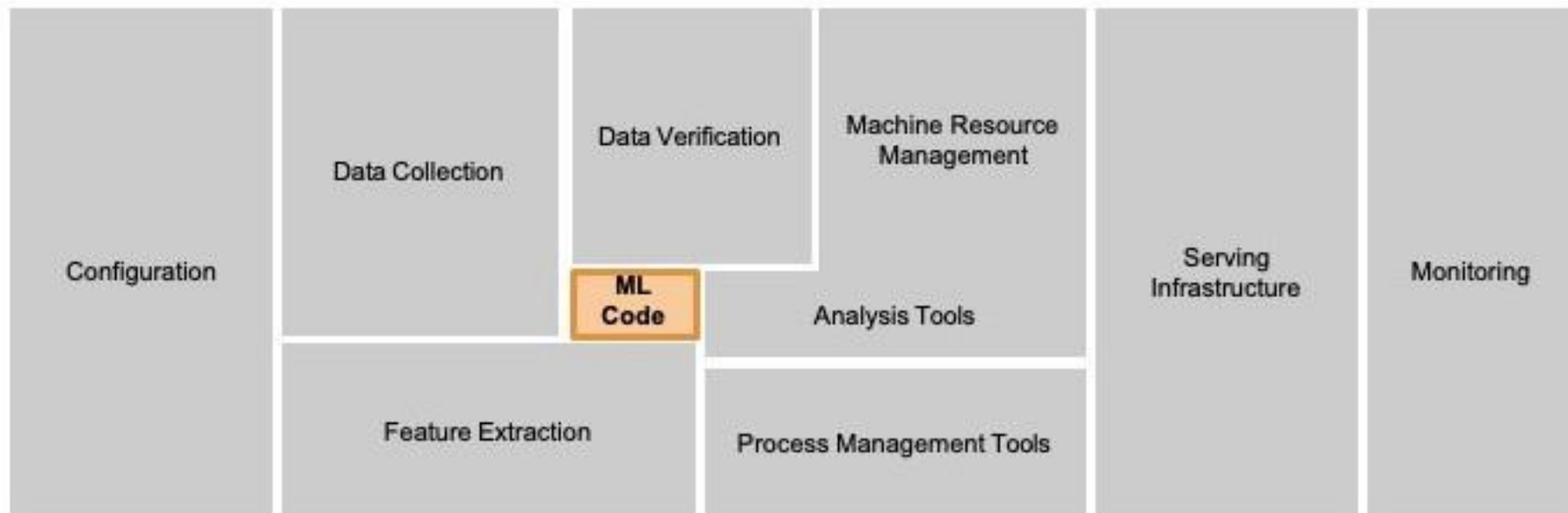
<https://bookdown.org/max/FES/detecting-interaction-effects.html>



Exercice



The Requirements Surrounding ML Infrastructure



Expressions régulières (1/1)

- **ion** : recherche les mots qui contiennent la chaîne "ion", dans n'importe quelle position
- **ion\$** : mots se terminant par "ion" (\$=fin de mot)
- **^anti** : recherche tous les mots commençant par "anti" (^=début de mot)
- **^maison\$** : recherche exactement le mot "maison"
- **p.r** : recherche les mots qui contiennent un "p", suivi d'une lettre quelconque, puis d'un "r" (le point correspond à n'importe quel caractère)
- **^p...r\$** : mots commençant par "p", suivi de trois lettres quelconques, et finissant par "r" (le symbole . dans une regex correspond à n'importe quel caractère)
- **^p.*r\$** : mots commençant par "p" et finissant par "r" (*= répétitions – 0 ou plusieurs fois – du caractère précédent, ici '.', donc n'importe quel caractère)
- **oid|ion|ein** : recherche les mots qui contiennent (au moins) une des trois chaînes "iod", "ion" ou "ein" (| = ou).

Source : http://www.lexique.org/?page_id=101
<https://buzut.net/la-puissance-des-regex/>

Expressions régulières (2/2)

- `[A-Za-z]` : n'importe quoi comme caractère, majuscule ou minuscule
- `+` : 1 ou plus
- `()` : contenu à extraire
- `\w` : un caractère alphanumérique ou un tiret de soulignement (tiret de 8).

Source : http://www.lexique.org/?page_id=101
<https://buzut.net/la-puissance-des-regex/>



RFormula

~

Separate target and terms

+

Concat terms; "+ 0" means removing the intercept (this means that the y-intercept of the line that we will fit will be 0)

-

Remove a term; "- 1" means removing the intercept (this means that the y-intercept of the line that we will fit will be 0—yes, this does the same thing as "+ 0")

:

Interaction (multiplication for numeric values, or binarized categorical values)

.

All columns except the target/dependent variable

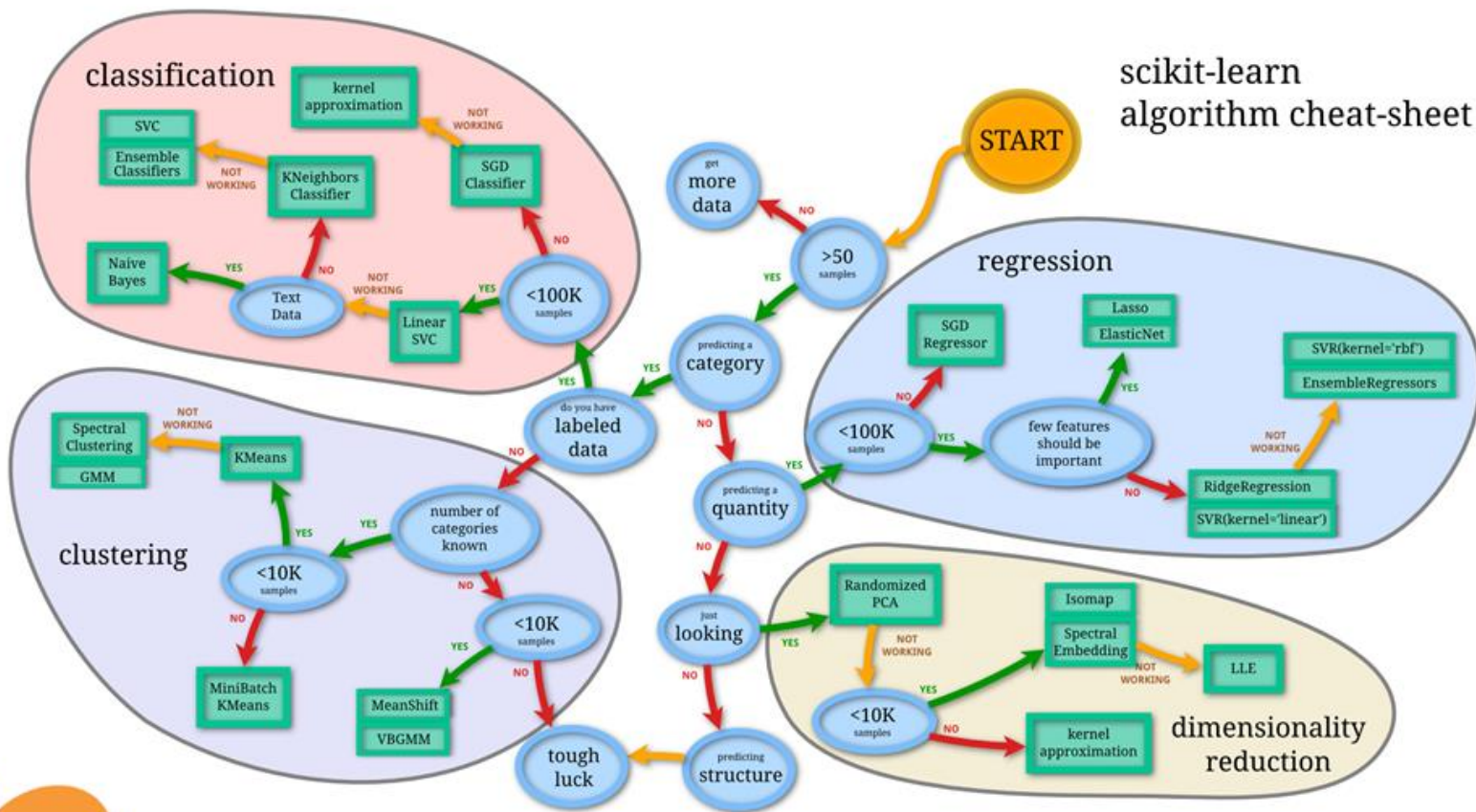
Python vs. R: What's the Difference?

**What is the difference between
Anaconda Prompt and Anaconda
Powershell Prompt?**



Choisir son algorithme

scikit-learn
algorithm cheat-sheet



Back

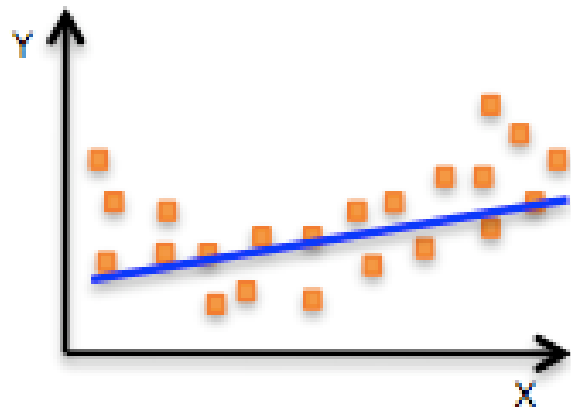
scikit-learn



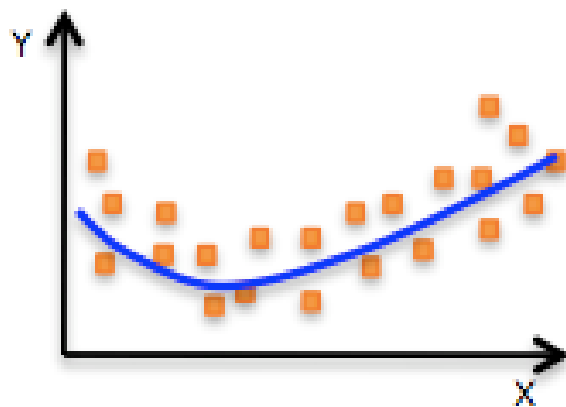
THANKS

Does anyone have any questions?

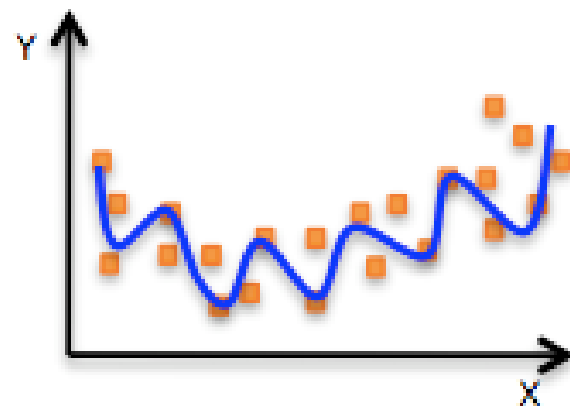
CREDITS: Fei GAO (EHESP)



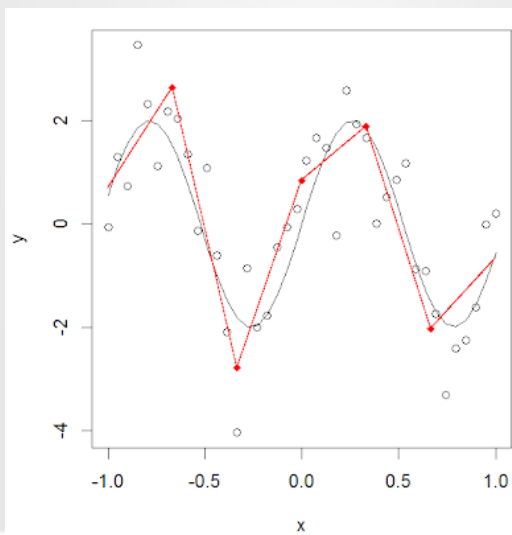
Underfitting

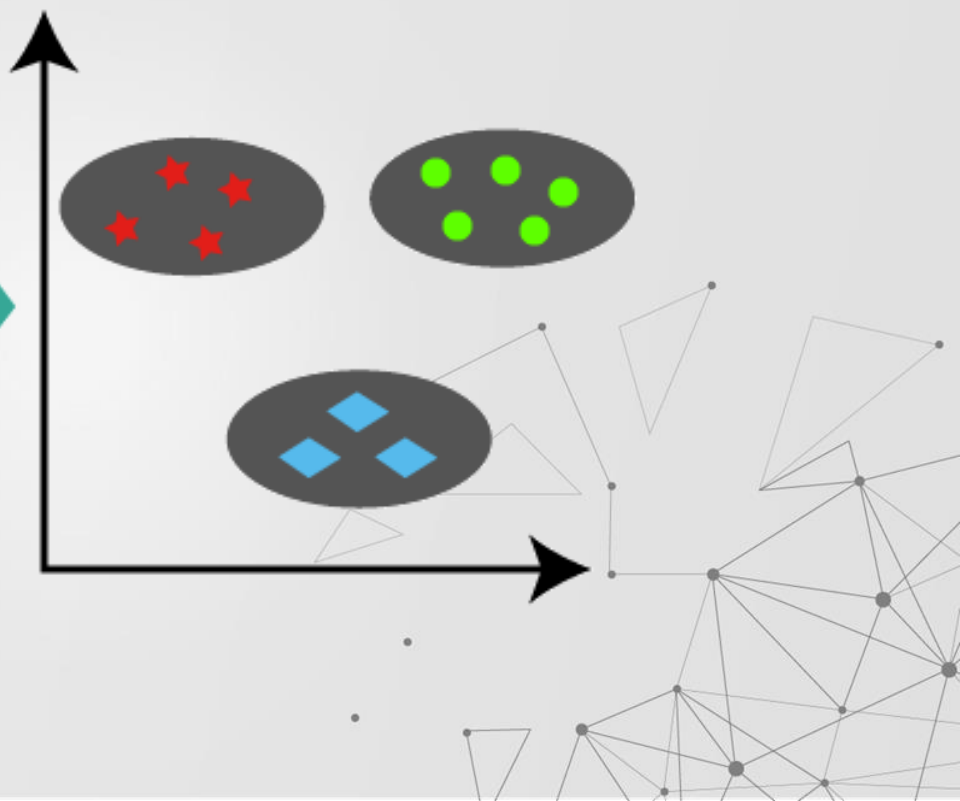
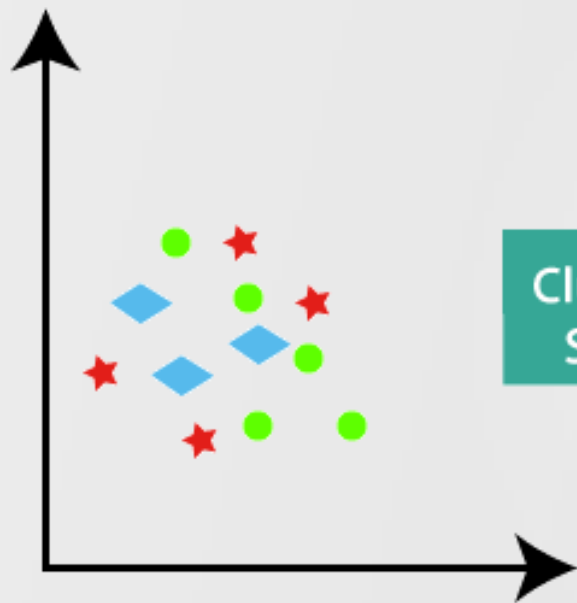


Just right!



overfitting





Créez votre compte Databricks

1/2

Prénom

Fei

Nom

GAO

E-mail professionnel

constance.fe.gao.pro@gmail.com

Entreprise

hdh

Intitulé de poste

DS

Numéro de téléphone (facultatif)

0661213981

Pays

France

- ☐ Oui, je souhaite recevoir les communications marketing concernant les services, les événements et les produits open source de Databricks. Je suis conscient(e) que je peux modifier [mes préférences](#) à tout moment.

Continuer

Sélectionnez un fournisseur de cloud

2/2



Amazon Web Services



Microsoft Azure



Google Cloud Platform

Continuer

En cliquant sur « Essayer », vous acceptez la [Politique de confidentialité](#) et les [Conditions de service](#).

Vous n'avez pas de compte cloud ?

Community Edition est un environnement Databricks limité destiné à un usage personnel et à la formation.

[Essayer Community Edition →](#)

En cliquant sur « Essayer Community Edition », vous acceptez la [Politique de confidentialité](#) et les [Conditions de service](#) undefined.