# **Définitions**



#### **Machine Learning**

Entraîner un modèle de prédiction sur des données pour qu'il puisse extrapoler sur des nouvelles données.

#### Variable cible.

Variable que le modèle apprend à prédire.

### Approche supervisée / Approche non supervisée

On a un exemple des valeurs de la variable cible / ou non.

#### **Erreur d'estimation**

Différence entre la valeur prédite par le modèle et la valeur réelle.

#### Régression linéaire

Modèle simple pour la prédiction de valeurs continues.

## Régression logistique

Modèle simple pour la prédiction de valeurs catégoriques.

#### Arbre de décision

Enchaînement de règles de classification établies automatiquement à partir des variables prédictrices.

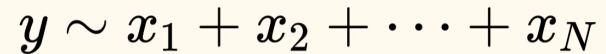
## Régularisation

Contrainte apporté au modèle pour l'empêcher d'overfitter.

### Bagging

Technique d'ensemblage de plusieurs modèles par la moyenne de leurs prédictions.

### Formules **\_**



permet de résumer la régression d'une variable cible y par rapport aux N prédicteurs  $x_1, x_2, \cdots, x_N$ 

$$ext{RMSE} = \sqrt{rac{1}{N}\sum_{k=1}^{N}(y_k-\hat{y}_k)^2}$$

permet d'évaluer la performance d'un modèle de régression.

# Rappel = TP / (TP + FN)

permet d'évaluer la performance d'un modèle de classification en minimisant les faux négatifs.

## Précision = TP / (TP + FP)

permet d'évaluer la performance d'un modèle de classification en minimisant les faux positifs.

# **Bonnes pratiques**



- Connaître le jeu de données avant d'entraîner un modèle.
- Nettoyer le dataset des données aberrantes, extrêmes ou manquantes.
- Adapter les valeurs brutes au modèle : mise à l'échelle, numérisation.
- Détecter rapidement l'overfit ou le biais du modèle
- Tracer l'histogramme des probabilités des prédictions dans le cadre d'une classification binaire.
- Travailler les prédicteurs (feature engineering) en les transformant ou en ajoutant de nouveaux, apporte souvent des gains de performances.

# X = matrice des prédicteurs y = array de la variable cible Répartition train / test from sklearn.model selection import train test split X train, X test, y train, y test = train test split( X, y, train size=0.8, random state=808) Instancier le modèle from sklearn.tree import DecisionTreeClassifier clf = DecisionTreeClassifier() Entraîner le modèle clf.fit(X train, y train) Prediction sur test y pred = clf.predict(X test) Evaluation sur test clf.score(y test, y pred) Matrice de confusion from sklearn.metrics import confusion matrix confusion matrix(y\_test, y\_test\_pred)

# **Erreurs classiques**



- Ne pas prendre en compte la reproductibilité des expériences en oubliant de fixer le random state.
- ★ Optimiser le modèle sur une unique répartition train / test → validation croisée.
- Multiplier le nombre de variables prédictives en appliquant le one hot encoding aveuglement (curse of dimension / piège des grandes dimensions).
- X Se satisfaire d'un score excellent qui pourrait être le résultat de fuite d'information dans les prédicteurs.
- ne pas faire de benchmark avec un modèle simple avant d'entraîner des modèles plus complexes

