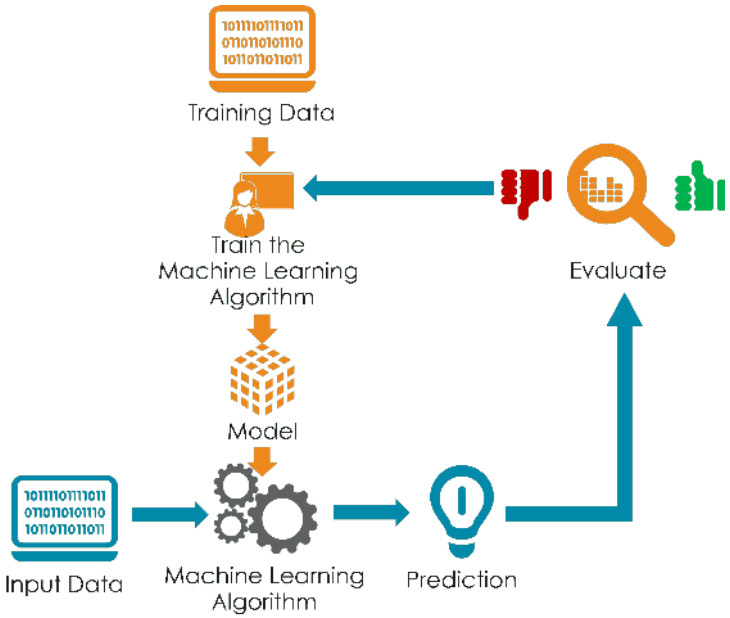
Évaluations



Précision, rappel, matrice de confusion

Précision

$$\text{precision} = \frac{\left|\left\{\text{relevant documents}\right\} \cap \left\{\text{retrieved documents}\right\}\right|}{|\{\text{retrieved documents}\}|}$$

Rappel :

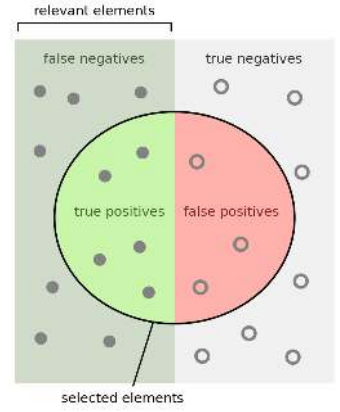
$$\text{recall} = \frac{\left|\left\{\text{relevant documents}\right\} \cap \left\{\text{retrieved documents}\right\}\right|}{|\{\text{relevant documents}\}|}$$

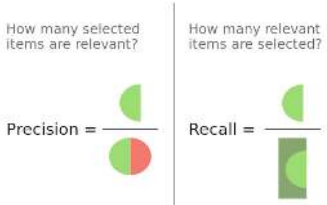
Matrice de confusion :

classe prédite \ classe réelle	*chat*	*chien*
chat	5	2
chien	3	3

classe prédite \ classe réelle	*Pos*	*Neg*
Pos	*TP*	*FP*
Neg	*FN*	*TN*

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Recall} = \frac{TP}{TP+FN}$$





Différents indicateurs (1)

- sensitivity, recall, hit rate, or true positive rate (TPR) :
  $\text{TPR} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP}+\text{FN}} = 1 - \text{FNR}$
- specificity, selectivity or true negative rate (TNR) :
  $\text{TNR} = \frac{\text{TN}}{\text{N}} = \frac{\text{TN}}{\text{TN}+\text{FP}} = 1 - \text{FPR}$
- precision or positive predictive value (PPV) :
  $\text{PPV} = \frac{\text{TP}}{\text{TP}+\text{FP}} = 1 - \text{FDR}$
- negative predictive value (NPV) : $\text{NPV} = \frac{\text{TN}}{\text{TN}+\text{FN}} = 1 - \text{FOR}$
- miss rate or false negative rate (FNR) :
  $\text{FNR} = \frac{\text{FN}}{\text{P}} = \frac{\text{FN}}{\text{FN}+\text{TP}} = 1 - \text{TPR}$
- fall-out or false positive rate (FPR) : $\text{FPR} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{FP}+\text{TN}} = 1 - \text{TNR}$
- false discovery rate (FDR) : $\text{FDR} = \frac{\text{FP}}{\text{FP}+\text{TP}} = 1 - \text{PPV}$
- false omission rate (FOR) : $\text{FOR} = \frac{\text{FN}}{\text{FN}+\text{TN}} = 1 - \text{NPV}$

Sources : Fawcett (2006),[4] Powers (2011),[1] Ting (2011),[5] and CAWCR[6]

Différents indicateurs (2)

- Threat score (TS) or Critical Success Index (CSI) $\text{TS} = \frac{\text{TP}}{\text{TP}+\text{FN}+\text{FP}}$
- accuracy (ACC) : $\text{ACC} = \frac{\text{TP}+\text{TN}}{\text{P}+\text{N}} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$
- Balanced Accuracy $= \frac{TPR + TNR}{2}$
- F1 score is the harmonic mean $\left(\frac{\frac{1}{a}+\frac{1}{b}}{2}\right)^{-1}$ of precision and sensitivity :
  $\text{F}_1 = 2 \cdot \frac{\text{PPV} \cdot \text{TPR}}{\text{PPV}+\text{TPR}} = \frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}$
- Matthews correlation coefficient [1975] (MCC, value $\in [-1, 1]$)
  = Pearson's phi coefficient [1946], Yule phi coefficient [1912] :

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$$

- Informedness or Bookmaker Informedness (BM) : $\text{BM} = \text{TPR} + \text{TNR} - 1$
- Markedness (MK) : $\text{MK} = \text{PPV} + \text{NPV} - 1$

Sources : Fawcett (2006),[4] Powers (2011),[1] Ting (2011),[5] and CAWCR[6]

Quelques exemples : régression polynomiale

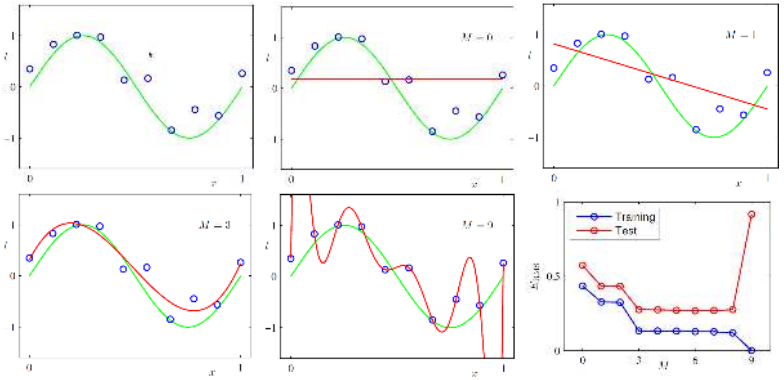


Figure from Machine Learning and Pattern Recognition, Bishop

Connaissance *a priori* / données

- Très forte connaissance *a priori* : $\mathcal{H} = \{f^*\}$

  f^*: the best function

  ⇒ Pas besoin de données !
- Très forte connaissance *a priori* : $\mathcal{H} = \{f^*, f_1\}$

  f^*: the best function

  ⇒ Quelques points suffisent pour détecter f^*

Connaissance *a priori* / données

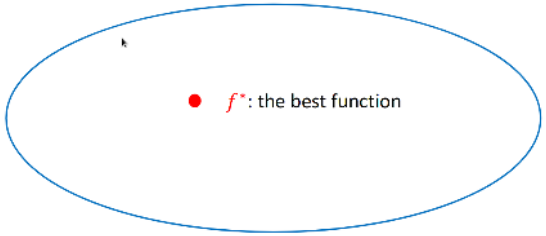
- Très forte connaissance *a priori* : $\mathcal{H} = \{f^*\}$

  f^*: the best function

  ⇒ Pas besoin de données !
- Très forte connaissance *a priori* : $\mathcal{H} = \{f^*, f_1\}$

  f^*: the best function

  ⇒ Quelques points suffisent pour détecter f^*
- Ensemble de données gigantesque : infini
  La classe des hypothèses $\mathcal{H}$ peut être l'ensemble de toutes les fonctions



Connaissance *a priori* / données

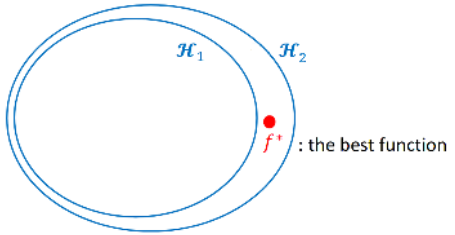
- Très forte connaissance *a priori* : $\mathcal{H} = \{f^*\}$

  f^*: the best function

  ⇒ Pas besoin de données !
- Très forte connaissance *a priori* : $\mathcal{H} = \{f^*, f_1\}$

  f^*: the best function

  ⇒ Quelques points suffisent pour détecter f^*
- Ensemble de données gigantesque : infini
  La classe des hypothèses $\mathcal{H}$ peut être l'ensemble de toutes les fonctions
- Scénarii classiques :
  ensemble fini de données, $\mathcal{H}$ de capacité moyenne, $f^* \notin \mathcal{H}$



Évaluation de l'hypothèse apprise

- Supposons
  - on a appris un modèle (arbre de décision, SMV, NN...)
  - Les données d'entraînement sont parfaitement apprises
- Est-ce un bon modèle ?
- L'erreur d'entraînement n'est pas suffisante :
  - il faut confronter le modèle à de nouveaux exemples : exemple de généralisation
  - Un modèle peut classer parfaitement les données d'entraînement et ne pas savoir classer de nouveaux exemples :
    - car les exemples d'entraînement sont incomplets (échantillon d'une distribution)
    - car les exemples d'entraînement peuvent être bruitées (erreur d'étiquetage, erreur de mesure sur les caractéristiques...)
- Rappel : Formalisation de l'induction
  étant donnés une fonction objectif I et un échantillon provenant d'une distribution inconnue D, le but est de calculer une fonction f dont l'erreur a une faible espérance sur D :

$$\mathbb{E}_{(x,y)\sim D}\{I(y, f(x))\} = \sum_{(x,y)} D(x,y) I(y, f(x))$$

Sur-apprentissage / sous-apprentissage

- Considérons une hypothèse h et
  - taux d'erreur sur l'ensemble d'entraînement $erreur_{train}(h)$
  - taux d'erreur sur toutes les données $erreur_{true}(h)$
- On dit que h sur-apprend l'ensemble d'apprentissage si

$$erreur_{train}(h) < erreur_{true}(h)$$

  évaluation du sur-apprentissage : $erreur_{true}(h) - erreur_{train}(h)$
- Problème : on ne connait pas $erreur_{true}(h)$
  Solution :
  - on met de coté un ensemble de test
    ce sont des exemples qui seront utilisés pour l'évaluation
  - on ne s'en sert pas pour l'entraînement du modèle
  - après l'entraînement, on calcule $erreur_{test}(h)$.
- **Underfitting :** un algo qui n'apprend pas suffisamment de la phase d'apprentissage
  Overfitting : un algo qui prend trop en compte les particularités de l'ensemble d'apprentissage. l'hypothèse se généralise mal
- **ce qu'on cherche :** Un modèle qui ne sur-apprend ni ne sous-apprend pas

dilemme (ou compromis) biais variance

Le dilemme (ou compromis) biais variance : minimiser simultanément deux sources d'erreurs qui empêchent les algorithmes d'apprentissage supervisé de généraliser au-delà de leur échantillon d'apprentissage :

- **Le biais :** erreur provenant d'hypothèses erronées
  manque de relations pertinentes entre les données en entrée et les sorties prévues (sous-apprentissage).
- **La variance :** erreur due à la sensibilité aux petites fluctuations de l'échantillon d'entraînement. $\Rightarrow$ peut entraîner un surapprentissage.

Décomposition biais-variance de l'erreur quadratique

- ens d'entraînement : $x_1, \ldots, x_n$ et y_i.
- $y_i = f(x_i) + \epsilon_i$, où le bruit, ϵ_i a une moyenne nulle et une variance σ^2.
- Trouver une fonction $\hat{f}$ qui se généralise à des points extérieurs à l'ensemble d'entrainement
- son erreur attendue sur un échantillon test x

$$\mathrm{E}\left[\left(y - \hat{f}(x)\right)^2\right] = \mathrm{Biais}\left[\hat{f}(x)\right]^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2$$

où

$$\mathrm{Biais}\left[\hat{f}(x)\right] = \mathrm{E}\left[\hat{f}(x) - f(x)\right]$$

et

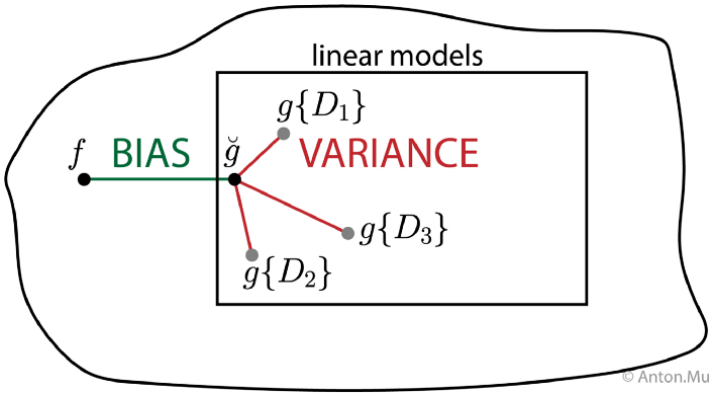
$$\mathrm{Var}\left[\hat{f}(x)\right] = \mathrm{E}\left[\left(\hat{f}(x) - \mathrm{E}[\hat{f}(x)]\right)^2\right]$$

Dérivation de la décomposition biais/variance

- $Var[X] = E[X^2] - (E[X])^2 \quad \Leftrightarrow \quad E[X^2] = Var[X] + (E[X])^2$.
- puisque f est déterministe : $E[f] = f$.
- comme $y = f + \varepsilon$ et $E[\varepsilon] = 0$, on a $E[y] = E[f + \varepsilon] = E[f] = f$.
- puisque $Var[\varepsilon] = \sigma^2$,
  $Var[y] = E[(y - E[y])^2] = E[(y - f)^2] = E[(f + \varepsilon - f)^2]$
  $= E[\varepsilon^2] = Var[\varepsilon] + (E[\varepsilon])^2 = \sigma^2$
- ainsi, puisque ε et $\hat{f}$ sont indépendants, on peut écrire :

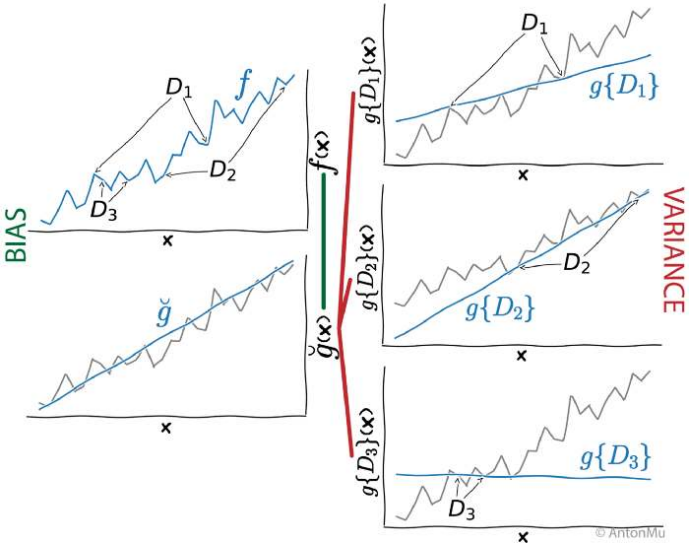
$$\begin{aligned}
E\left[(y - \hat{f})^2\right] &= E\left[(f + \varepsilon - \hat{f})^2\right] = E\left[(f + \varepsilon - \hat{f} + E[\hat{f}] - E[\hat{f}])^2\right] \\
&= E\left[(f - E[\hat{f}])^2\right] + E[\varepsilon^2] + E\left[(E[\hat{f}] - \hat{f})^2\right] \\
&\quad + 2E\left[(f - E[\hat{f}])\varepsilon\right] + 2E\left[\varepsilon(E[\hat{f}] - \hat{f})\right] + 2E\left[(E[\hat{f}] - \hat{f})(f - E[\hat{f}])\right] \\
&= (f - E[\hat{f}])^2 + E[\varepsilon^2] + E\left[(E[\hat{f}] - \hat{f})^2\right] \\
&\quad + \underbrace{2(f - E[\hat{f}])E[\varepsilon]}_{=0} + \underbrace{2E[\varepsilon]E\left[E[\hat{f}] - \hat{f}\right]}_{=0} + \underbrace{2E\left[E[\hat{f}] - \hat{f}\right](f - E[\hat{f}])}_{=0} \\
&= (f - E[\hat{f}])^2 + E[\varepsilon^2] + E\left[(E[\hat{f}] - \hat{f})^2\right] \\
&= (f - E[\hat{f}])^2 + Var[y] + Var\left[\hat{f}\right] \\
&= Bias[\hat{f}]^2 + Var[y] + Var\left[\hat{f}\right] \\
&= Bias[\hat{f}]^2 + \sigma^2 + Var\left[\hat{f}\right]
\end{aligned}$$

Illustration



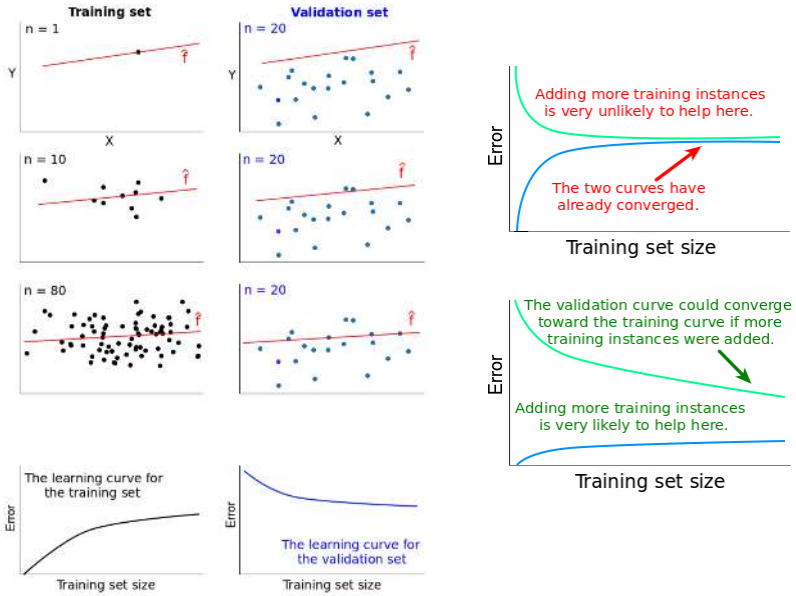
CÔTE D'AZUR
AI for bio
J-P Comet
Introduction
Trees
Clustering
kNN
Bayes
SVM
NN
Evaluation
Précision et rappel
Exemple
Formalisation
Data Mining
Project

Illustration



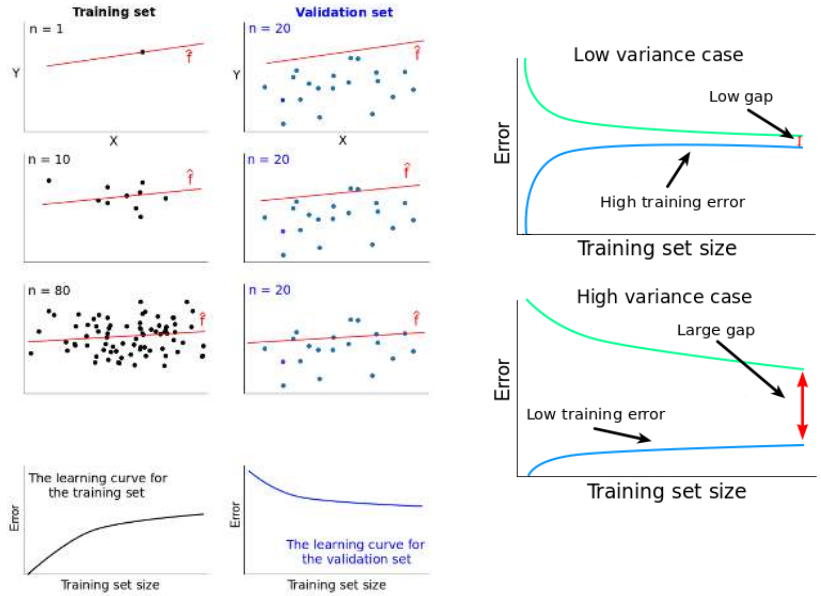
CÔTE D'AZUR
AI for bio
J-P Comet
Introduction
Trees
Clustering
kNN
Bayes
SVM
NN
Evaluation
Précision et rappel
Exemple
Formalisation
Data Mining
Project

Learning curves

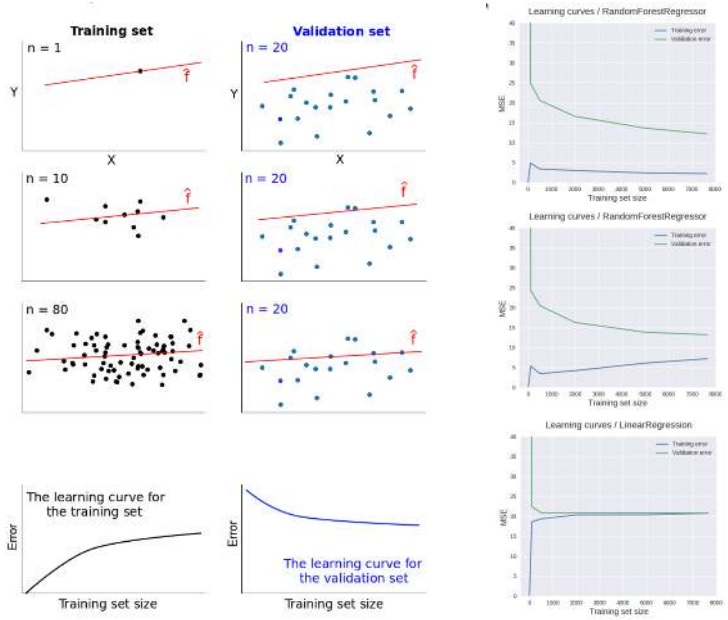


CÔTE D'AZUR
AI for bio
J-P Comet
Introduction
Trees
Clustering
kNN
Bayes
SVM
NN
Evaluation
Précision et rappel
Exemple
Formalisation
Data Mining
Project

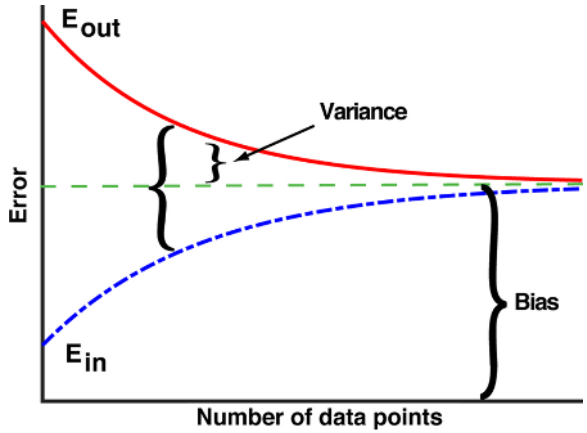
Learning curves



Learning curves

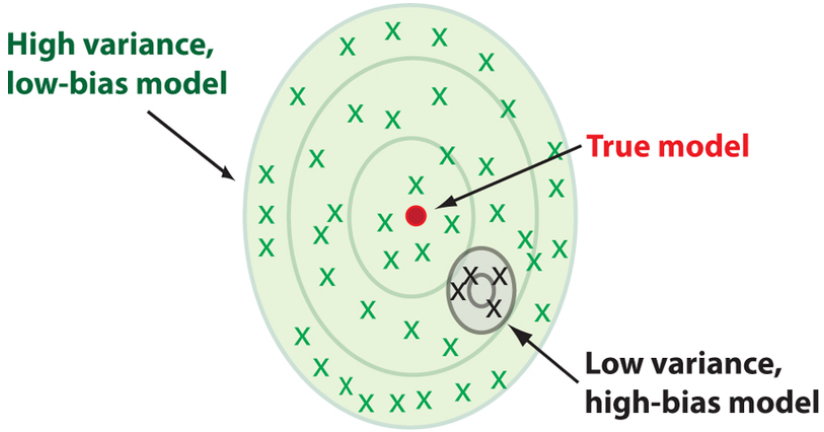


Learning curves



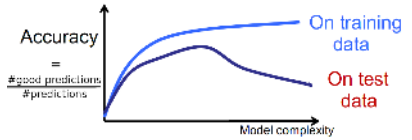
- Hypothesis : we cannot exactly learn the function $f^*(x)$
- As the number of data points gets large, the sampling noise decreases and the training data set becomes more representative of the true distribution
- in the infinite data limit, the in-sample and out-of-sample errors must approach the same value, which is called the bias of our model.
- The bias : the best our model could do if we had an infinite amount of training data to beat down sampling noise.

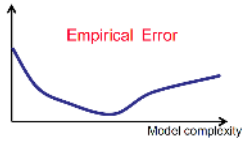
Learning curves



- Hypothesis : we cannot exactly learn the function $f^*(x)$
- As the number of data points gets large, the sampling noise decreases and the training data set becomes more representative of the true distribution
- in the infinite data limit, the in-sample and out-of-sample errors must approach the same value, which is called the bias of our model.
- The bias : the best our model could do if we had an infinite amount of training data to beat down sampling noise.

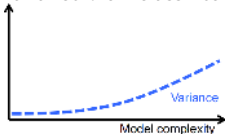
Sur-apprentissage / Overfitting





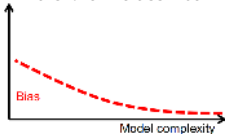
- Un modèle sur-apprend l'ensemble d'apprentissage quand la justesse (accuracy) sur l'ensemble d'apprentissage augmente alors que la justesse sur des nouvelles données diminue.
- Erreur empirique (sur un ensemble de données) : % d'items qui ne sont pas bien classés.

Variance d'un classifieur



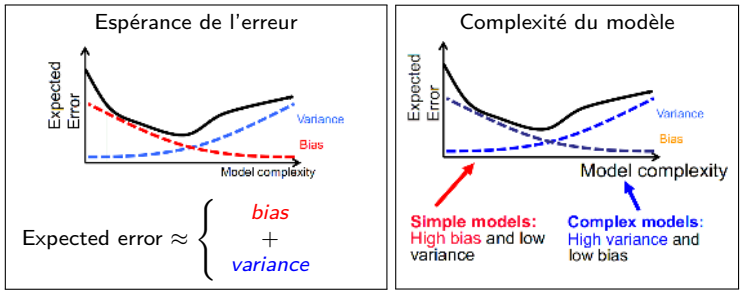
- Sensibilité du classifieur à des changements mineurs dans les données d'entraînement (*i.e.* à différents échantillons de $P(X, Y)$)
- Variance croît avec la complexité du modèle.

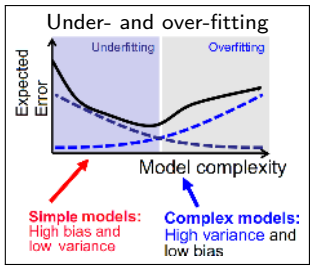
Biais d'un classifieur

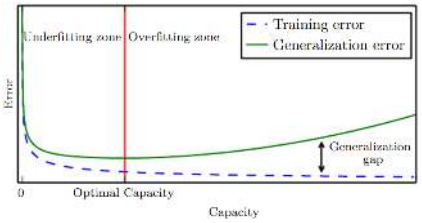


- Capacité du classifieur à identifier l'hypothèse cible ?
- Le biais est petit lorsque le modèle est expressif (peu d'erreurs empirique)
- Le biais est grand lorsque le modèle est (trop) simple

"*Plus l'espace des hyp est grand, plus il est facile de se rapprocher de l'hypothèse cible.*"

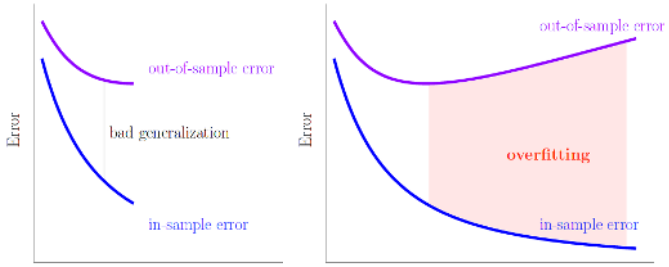


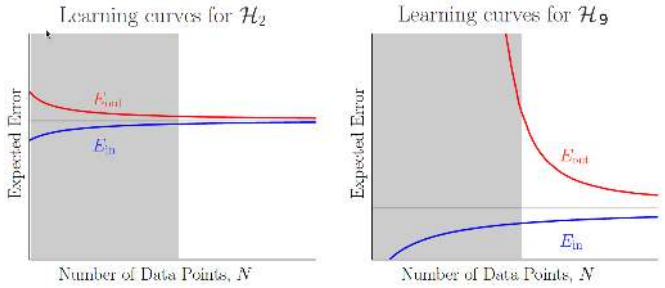




Sur-apprentissage / erreur de généralisation

Le sur-apprentissage n'est pas qu'une mauvaise erreur de généralisation





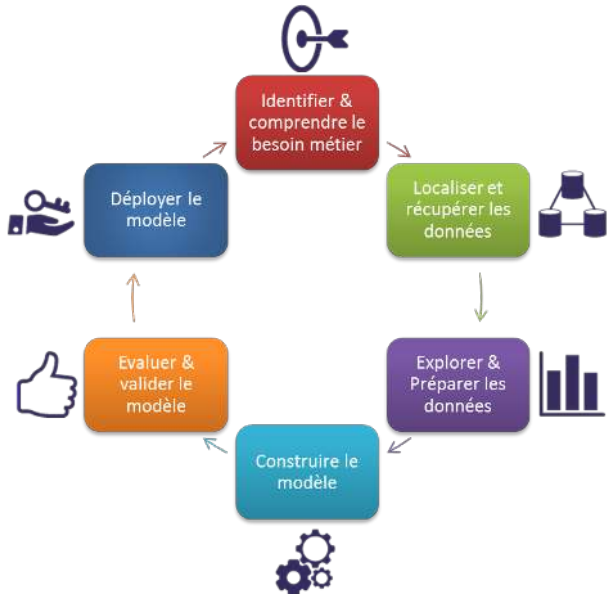
Ensembles d'apprentissage, de dev., de test

En pratique, on répartit les exemples en 3 ensembles distincts :

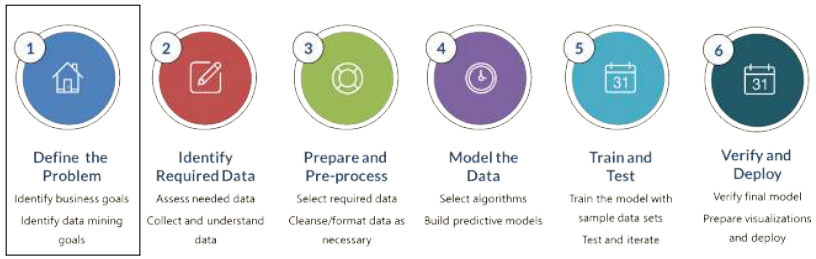
- Ensemble d'entraînement :
  - utilisé pour entraîner le modèles
  - *e.g.* les nœuds et branches d'un arbre de décision
- Ensemble de développement (tuning set/validation set/ held out data) :
  - Utilisé pour apprendre les hyperparamètres :
    - parametres contrôlant les autres paramètres
    - *e.g.* max_depth d'un arbre de décision
- Ensemble Test :
  - utilisé pour évaluer la performance sur des données pas encore vues.

⇒ Never ever touch your test data !

Déroulement d'un projet



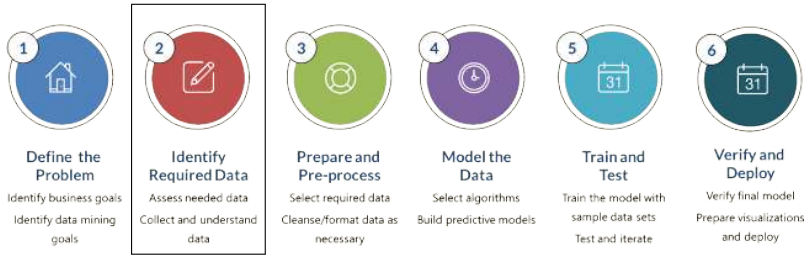
Définir le problème



Quel est le problème que l'on essaie de résoudre ?

- Se poser les bonnes questions.
  Quels types de relations faut-il trouver ?
- Définir ensuite **précisément** l'objectif **mesurable** et **quantifiable** du projet
- L'expression du besoin :
  - critère de réussite + condition d'arrêt
  - Compréhension commune de ce qui peut être espéré
  - Connaissances *a priori* sur les données ? Saisonnalité ? distribution *a priori* ?
  - éléments logistique du projet (budget, plan, deadline...)

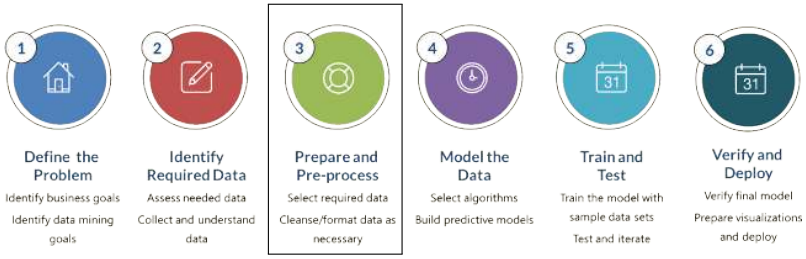
Préparer les données



- Les données peuvent être dispersées dans une entreprise et stockées dans différents formats, ou contenir des incohérences telles que des entrées incorrectes ou manquantes. Par exemple, les données peuvent indiquer qu'un client a acheté un produit avant que le produit ne soit offert sur le marché, ou que le client achète régulièrement dans un magasin situé à 3 000 km de son domicile.
- Nettoyage : traiter les données incorrectes / manquantes ; rechercher des corrélations cachées ; identifier les sources de données les plus précises ; déterminer les colonnes les plus appropriées pour une analyse.
  Par exemple, utilisation de la date d'expédition ou de commande ?

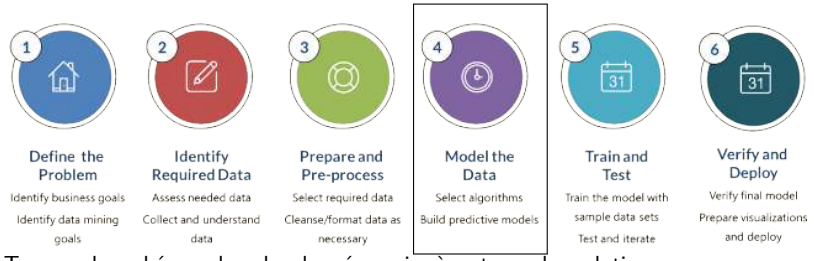
⇒ identifier tous ces problèmes et déterminer comment les résoudre.
Comme on ne peut passer en revue tous les enregistrements, on peut essayer d'abord de filtrer les données de manière automatique.

Explorer les données



- comprendre les données afin de prendre les décisions appropriées lors de la création des modèles.
  calcul des valeurs minimales et maximales, de moyennes et des écarts types, examen de la distribution des données.
- Est-ce que le jeu de données contient des données erronées ? Quelle stratégie pour résoudre les problèmes ou acquérir une compréhension plus approfondie des comportements typiques ?

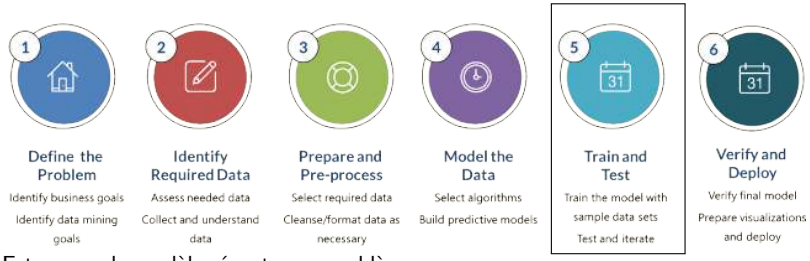
Construire des modèles



Trouver des schémas dans les données qui mènent vers les solutions

- des "va-et-vient" avec l'étape précédente peuvent être nécessaires
- Tâches de modélisation :
  - Classification
  - Prévision
  - Classement
  - Segmentation
  - trouver des associations/corrélations
- Exemple de méthodes de classification : arbre de décision, NN...

Evaluation et critique du modèle

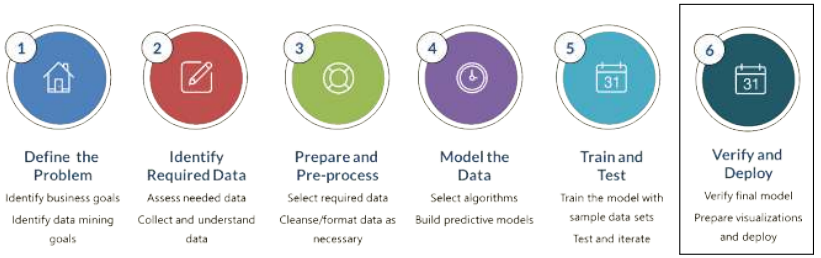


Est-ce que le modèle résout mon problème

- est-il précis selon les besoins ? Est-ce qu'il généralise bien ?
- Est-il meilleur que le "résultat évident" ou modèle de base ?
- Les résultats ont-ils du sens dans le contexte du problème ?

$\Rightarrow$ s'il y a plusieurs étapes, il faut faire une évaluation de chacune des tâches.

Déploiement et Mise à jour



une fois que le modèle ait été construit :

- Utilisation du modèle pour faire des prédictions/décisions. (mise en place d'un serveur d'interrogation (DMX queries) ?)
- Création de requêtes sur les statistiques, les règles ou formules au sein d'un modèle (DMX).
- Intégration de la fonctionnalité d'exploration de données directement dans une application
- Mise à jour les modèles. Toute mise à jour nécessite le retraitement des modèles.
- Mise à jour automatique des modèles.