Introduction to Supervised Learning & Learning Theory

Apprentissage supervisé

On se donne des données d'entraînement étiquetées, c'est à dire qu'on connait la réponse que le modèle devra apporter. On peut entraîner et tester modèle sur ces données, ce qui est pratique pour estimer à quel point il est performant. Il existe deux types de problèmes en apprentissage supervisé : la classification et la régression. Il existe plusieurs types de classifications : la classification binaire (on classe en deux catégories les data), la classification multi-classe (on a plus de deux catégories) et la classification multi labels (on peut associer plusieurs catégories à une même data).

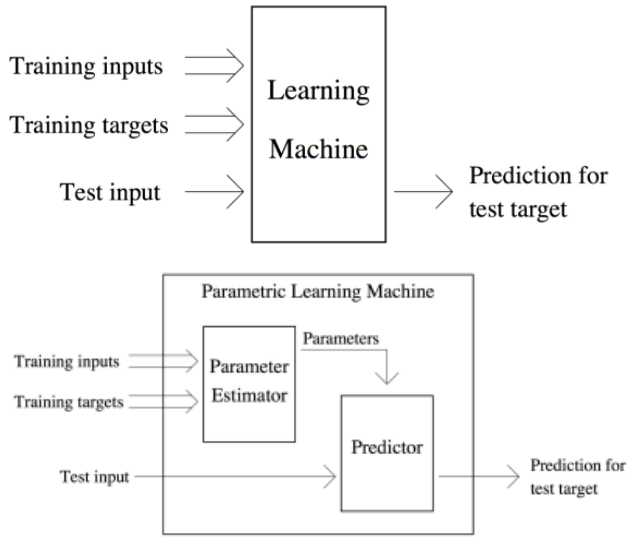
Définition

Data mining : rechercer des données pour avoir le plus de données possible pour l'entraînement.

Complexité

Il n'est pas toujours nécessaire d'entraîner un modèle avec énormément de données et énormément de features. Il faut parfois savoir optimiser sa base de données pour éviter de prendre trop de temps et d'énergie pour entraîner le modèle.
Pour réduire la complexité des données, il est par exemple possible d'effectuer une réduction de dimension, en enlevant les features qui sont les plus liées les unes aux autres (PCA).

Modèle de l'apprentissage supervisé



Cross-validation

Pour évaluer la qualité d'un algorithme d'apprentissage supervisé, et donc pour savoir quel modèle utiliser, on réalise la plupart du temps validation croisée. On choisis en général on divise l'échantillon original en k échantillons (ou "blocs"), puis on sélectionne un des k échantillons comme ensemble de validation pendant que les $k-1$ autres échantillons constituent l'ensemble d'apprentissage. Après apprentissage, on peut calculer une performance de validation. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les blocs prédéfinis. À l'issue de la procédure nous obtenons ainsi k scores de performances, un par bloc. La moyenne et l'écart type des k scores de performances peuvent être calculés pour estimer le biais et la variance de la performance de validation.

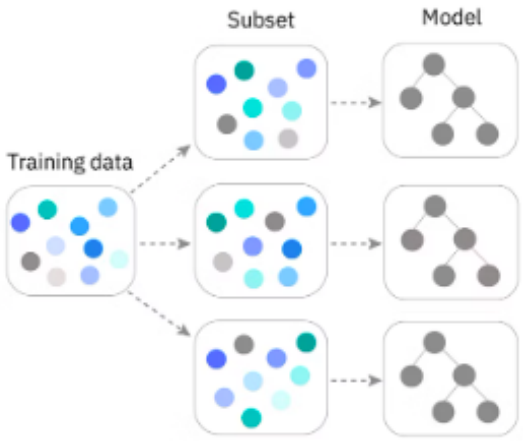
Deux classes d'algorithmes de classification

* Discriminatif ($P(y|x)$) : on cherche des frontières (perceptron, arbres de décision, SVM, ...).
Génératif ($P(x|y)$) : on cherche des clusters (Naive Bayes, LDA, QDA, ...).

Ensemble Methods

Les méthodes d'ensemble (ensemble methods) sont des techniques d'apprentissage automatique qui combinent plusieurs modèles de base pour améliorer les performances globales. Elles permettent de réduire la variance et le biais, et d'améliorer la précision des prédictions. Voici les principales méthodes d'ensemble :

Bagging (Bootstrap Aggregating)



Explication : Le Bagging, ou Bootstrap Aggregating, est une méthode qui consiste à entraîner plusieurs modèles sur des sous-ensembles aléatoires des données d'entraînement. Chaque sous-ensemble est créé en utilisant la technique du bootstrap, c'est-à-dire en tirant aléatoirement avec remise des échantillons de la même taille dans le jeu de données original. Les prédictions des modèles individuels sont ensuite combinées, généralement par vote majoritaire pour la classification ou par moyenne pour la régression.
Exemple : Un exemple bien connu de Bagging est la méthode des forêts aléatoires (Random Forests). Dans une forêt aléatoire, chaque arbre de décision est entraîné sur un sous-ensemble aléatoire. Les prédictions de tous les arbres sont ensuite combinées pour obtenir une prédiction finale plus robuste.
Avantages : - Réduit la variance du modèle, limitant ainsi le surapprentissage (overfitting). - Améliore la généralisation du modèle sur de nouvelles données [1].

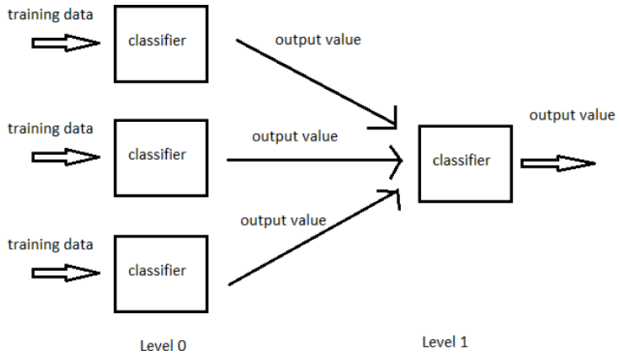
Boosting

Explication : Le Boosting est une technique itérative qui vise à améliorer les performances d'un modèle en se concentrant sur les erreurs commises par les modèles précédents. À chaque itération, un nouveau modèle est entraîné pour corriger les erreurs des modèles précédents, en attribuant un poids plus élevé aux observations mal classées.

Exemple : Des exemples populaires de Boosting incluent AdaBoost, Gradient Boosting, et XGBoost. Par exemple, dans AdaBoost, chaque nouvel arbre de décision est entraîné pour corriger les erreurs des arbres précédents, en ajustant les poids des observations dans le jeu de données.

Avantages : - Améliore la précision des modèles sur des jeux de données complexes. - Peut capturer des relations non linéaires dans les données [2].

Stacking



Explication : Le Stacking est une méthode qui combine plusieurs modèles de base en utilisant un méta-modèle pour apprendre comment combiner au mieux leurs prédictions. Les modèles de base sont d'abord entraînés sur les données d'entraînement. Leurs prédictions sont ensuite utilisées comme nouvelles caractéristiques pour entraîner le méta-modèle.

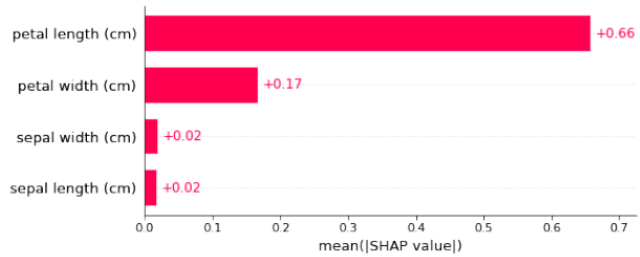
Exemple : Par exemple, si vous avez trois modèles de base (un arbre de décision, une régression logistique, et un SVM), le Stacking utilise un quatrième modèle (le méta-modèle) pour apprendre à combiner les prédictions des trois modèles de base de manière optimale.

Avantages : - Peut souvent surpasser les performances de chacun des modèles de base. - Exploite les forces de chaque modèle individuel pour améliorer la précision globale [3].

Model Interpretability

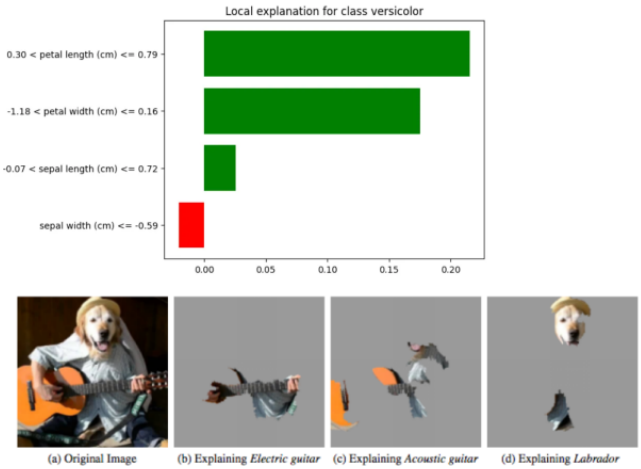
L'interprétabilité des modèles est cruciale pour comprendre comment un modèle prend ses décisions. Cela est particulièrement important dans des domaines sensibles comme la santé, la finance ou la cybersécurité, où la transparence est essentielle.

SHAP (SHapley Additive exPlanations)



Explication : SHAP est une méthode basée sur la théorie des jeux coopératifs pour expliquer les prédictions des modèles. Elle attribue une valeur d'importance à chaque caractéristique d'entrée, indiquant son impact sur la prédiction finale. Chaque valeur SHAP représente la contribution marginale d'une caractéristique à la prédiction, en tenant compte de toutes les combinaisons possibles de caractéristiques.
Exemple : Par exemple, si un modèle prédit le risque de défaut de paiement d'un client, SHAP peut montrer que le revenu et l'historique de crédit sont les caractéristiques les plus influentes dans la prédiction.
Avantages : - Fournit une explication globale et locale des prédictions. - Permet de comprendre quelles variables influencent le plus les décisions du modèle [5].

LIME (Local Interpretable Model-agnostic Explanations)



Explication : LIME explique les prédictions des modèles en approximant localement le comportement du modèle complexe par un modèle interprétable (comme une régression linéaire). LIME génère des perturbations autour d'une observation spécifique et entraîne un modèle simple sur ces perturbations pour expliquer la prédiction locale.
Exemple : Par exemple, pour expliquer pourquoi un modèle a classé une image comme un chat, LIME peut identifier les pixels ou les régions de l'image qui ont le plus contribué à cette classification.
Avantages : - Permet de comprendre comment le modèle se comporte dans le voisinage d'une prédiction spécifique. - Utile pour expliquer les prédictions de modèles complexes comme les réseaux de neurones [5].

References

[1] IBM. (2025). What is ensemble learning? `https://www.ibm.com/think/topics/ensemble-learning`

[2] IEEE. (2025). A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. https://ieeexplore.ieee.org/document/9893798/

[3] Wikipedia. (2025). Ensemble learning. https://en.wikipedia.org/wiki/Ensemble_learning

[4] ScienceDirect. (2024). Ensemble methods and semi-supervised learning for information fusion: A review and future research directions. https://www.sciencedirect.com/science/article/pii/S1566253524000885

[5] GeeksforGeeks. (2025). The Future of Machine Learning in 2025 [Top Trends and Predictions]. https://www.geeksforgeeks.org/blogs/future-of-machine-learning/

Hyperparameter Tuning

L'optimisation des hyperparamètres est une étape cruciale dans la construction de modèles d'apprentissage automatique. Les hyperparamètres sont des paramètres qui ne sont pas appris pendant l'entraînement, mais qui sont définis avant le processus d'apprentissage. Leur choix peut avoir un impact significatif sur les performances du modèle.

Qu'est-ce qu'un hyperparamètre ?

Un hyperparamètre est un paramètre dont la valeur est définie avant le début de l'apprentissage. Contrairement aux paramètres du modèle (comme les poids dans un réseau de neurones), qui sont appris à partir des données, les hyperparamètres contrôlent le processus d'apprentissage lui-même. Voici quelques exemples d'hyperparamètres :

- Le learning rate dans les réseaux de neurones. - Le nombre d'arbres dans une forêt aléatoire. - Le paramètre de régularisation dans une régression logistique.

Techniques d'optimisation des hyperparamètres

- **Grid Search**

  Explication : La recherche sur grille (Grid Search) est une méthode exhaustive qui évalue toutes les combinaisons possibles des valeurs d'hyperparamètres spécifiées. Vous définissez une grille de valeurs possibles pour chaque hyperparamètre, et Grid Search teste toutes les combinaisons possibles pour trouver celle qui donne les meilleures performances en utilisant la validation croisée.

  Exemple : Supposons que vous ayez deux hyperparamètres : le nombre d'arbres dans une forêt aléatoire (avec des valeurs possibles de 50, 100, et 200) et la profondeur maximale des arbres (avec des valeurs possibles de 5, 10, et 15). Grid Search testera toutes les combinaisons possibles (3 x 3 = 9 combinaisons) et choisira celle qui offre les meilleures performances sur un jeu de validation.

  - Avantage : Simple à mettre en œuvre et garantit de trouver la meilleure combinaison dans la grille définie. - Inconvénient : Peut être très coûteux en termes de calcul, surtout si la grille est grande ou si le modèle est complexe [4].

- **Random Search**

  Explication : La recherche aléatoire (Random Search) est une alternative à Grid Search. Au lieu de tester toutes les combinaisons possibles, elle sélectionne aléatoirement des combinaisons d'hyperparamètres dans les distributions spécifiées. Cela permet de couvrir un espace de recherche plus large sans explorer toutes les combinaisons en utilisant la validation croisée.

  Exemple concret : En utilisant le même exemple que pour Grid Search, Random Search sélectionnera aléatoirement des combinaisons de valeurs pour le nombre d'arbres et la profondeur maximale, par exemple (100, 10), (50, 15), etc., sans tester toutes les combinaisons possibles.

Avantages et inconvénients : - Avantages : Souvent plus efficace que Grid Search, surtout pour les espaces de recherche de grande dimension. - Inconvénients : Ne garantit pas de trouver la meilleure combinaison, mais peut trouver de bonnes combinaisons plus rapidement [4].

- **Bayesian Optimization**

  Explication : L'optimisation bayésienne est une méthode plus avancée qui utilise des techniques probabilistes pour trouver les meilleures valeurs d'hyperparamètres. Elle construit un modèle probabiliste des performances du modèle en fonction des hyperparamètres et utilise ce modèle pour guider la recherche vers les régions prometteuses de l'espace des hyperparamètres.

  Exemple concret : Supposons que vous optimisez le taux d'apprentissage et le nombre de couches dans un réseau de neurones. L'optimisation bayésienne commence par évaluer quelques combinaisons aléatoires, puis utilise ces résultats pour construire un modèle probabiliste. Ce modèle est ensuite utilisé pour prédire quelles combinaisons d'hyperparamètres devraient être testées ensuite.

  Avantages et inconvénients : - Avantages : Plus efficace que Grid Search et Random Search, car elle se concentre sur les régions prometteuses de l'espace des hyperparamètres. - Inconvénients : Plus complexe à mettre en œuvre et nécessite une compréhension plus approfondie des méthodes probabilistes [4].

Model Deployment

Le déploiement de modèles d'apprentissage automatique est une étape cruciale pour rendre les modèles utilisables en production. Cela implique de rendre le modèle accessible pour faire des prédictions sur de nouvelles données en temps réel ou par lots.

Étapes du déploiement

- **Préparation de l'environnement**

  Explication : Avant de déployer un modèle, il est essentiel de préparer l'environnement de production. Cela inclut l'installation des dépendances nécessaires, la configuration des serveurs, et la mise en place des pipelines de données pour alimenter le modèle avec les nouvelles données.

  Exemple : Si vous déployez un modèle de classification d'images, vous devez vous assurer que l'environnement de production dispose des bibliothèques nécessaires (comme TensorFlow ou PyTorch) et des ressources matérielles (comme des GPU) pour exécuter le modèle efficacement.

- **Monitoring et maintenance**

  Explication : Une fois le modèle déployé, il est crucial de le surveiller pour détecter toute dégradation des performances. Cela inclut la détection de la dérive des données (data drift), où les caractéristiques des nouvelles données diffèrent de celles utilisées pour entraîner le modèle, et la mise à jour régulière du modèle avec de nouvelles données.

  Exemple concret : Si un modèle de détection de fraude est déployé, il est important de surveiller régulièrement ses performances pour s'assurer qu'il continue à détecter correctement les transactions frauduleuses. Si les performances se dégradent, il peut être nécessaire de réentraîner le modèle avec des données plus récentes.

- **Sécurité et conformité**

  Explication : Lors du déploiement, il est essentiel de s'assurer que le modèle respecte les réglementations en vigueur, comme le RGPD pour la protection des données. Cela inclut la mise en place de mesures de sécurité pour protéger les données sensibles et garantir que le modèle est utilisé de manière éthique.

  Exemple concret : Si vous déployez un modèle de recommandation pour un site de commerce électronique, vous devez vous assurer que les données des utilisateurs sont protégées et que le modèle ne fait pas de recommandations biaisées ou discriminatoires.

References

[1] Analytics Vidhya. (2025). A Comprehensive Guide on Hyperparameter Tuning and its Techniques. https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

[2] Medium. (2025). Best Practices for Deploying Machine Learning Models in Production. https://medium.com/@nemagan/best-practices-for-deploying-machine-learning-models-in-production-10b690503e6d

Non-Linear Models & Feature Engineering

Non-Linear Models for Supervised Classification

Pourquoi des modèles non-linéaires ?

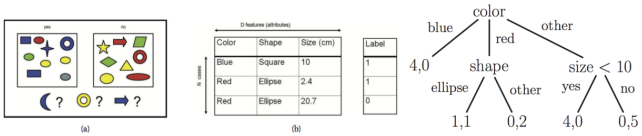
Les données du monde réel présentent souvent des motifs complexes et non-linéaires. Les modèles non-linéaires permettent de capturer ces relations d'ordre supérieur, ce qui les rend plus adaptés pour traiter des problèmes où les relations entre les variables ne sont pas linéaires.

Avantages des modèles non-linéaires

- Capturent des frontières de décision complexes. - Flexibles et adaptables à diverses distributions de données. - Souvent plus précis que les modèles linéaires.

Exemples de modèles non-linéaires

- **Arbres de décision (Decision Trees)** : Divisent récursivement les données en fonction de seuils sur les caractéristiques. Chaque nœud représente une décision basée sur une caractéristique, et chaque feuille représente une classe ou une valeur de régression.



  Exemple : Un arbre de décision peut être utilisé pour prédire si un client achètera un produit en fonction de son âge, de son revenu et de son historique d'achat.

- **Forêts aléatoires (Random Forest)** : Ensemble d'arbres de décision où chaque arbre est entraîné sur un sous-ensemble aléatoire des données. Les prédictions des arbres individuels sont combinées par vote majoritaire pour la classification ou par moyenne pour la régression.

  Exemple : Les forêts aléatoires sont souvent utilisées pour la détection de fraudes, où chaque arbre de décision contribue à une prédiction finale plus robuste.

- **Machines à vecteurs de support (SVM)** : Utilisent des astuces de noyau (kernel tricks) pour projeter les données dans des espaces de plus haute dimension, permettant de séparer les classes de manière non-linéaire.

  Exemple : Les SVM sont efficaces pour la classification d'images, où les caractéristiques peuvent ne pas être linéairement séparables dans l'espace d'origine.

- **Réseaux de neurones (Neural Networks)** : Modèles d'apprentissage profond avec plusieurs couches cachées, capables de capturer des relations complexes dans les données.

  Exemple : Les réseaux de neurones sont largement utilisés pour la reconnaissance vocale et la traduction automatique.

- **k-plus proches voisins (k-NN)** : Classifient les points de données en fonction de la similarité avec leurs voisins les plus proches.

  Exemple : k-NN est souvent utilisé pour les systèmes de recommandation, où les produits recommandés sont basés sur les préférences des utilisateurs similaires.

Decision Trees

Introduction aux arbres de décision

Les arbres de décision sont des modèles où la variable cible peut prendre un ensemble discret de valeurs (classification) ou des valeurs continues (régression). Chaque nœud de l'arbre représente une décision basée sur une caractéristique, et chaque feuille représente une classe ou une valeur de régression.

Construction d'un arbre de décision

- **Critère de division** : Pour diviser un nœud, on utilise une mesure d'impureté comme l'indice de Gini ou l'entropie. L'objectif est de maximiser la réduction de l'impureté après la division.

- **Exemple de critère de division** : L'indice de Gini mesure avec quelle fréquence un élément aléatoire de l'ensemble serait mal classé si son étiquette était choisie aléatoirement selon la distribution des étiquettes dans le sous-ensemble. L'indice de diversité de Gini peut être calculé en sommant la probabilité pour chaque élément d'être choisi, multipliée par la probabilité qu'il soit mal classé. Il atteint sa valeur minimum (zéro) lorsque tous les éléments de l'ensemble sont dans une même classe de la variable-cible.

$$Gini = \sum_{k=1}^{K} f_k(1 - f_k) = \sum_{k=1}^{K}(f_k - f_k^2) = \sum_{k=1}^{K} f_k - \sum_{k=1}^{K} f_k^2 = 1 - \sum_{k=1}^{K} f_k^2$$

  où f_k est la proportion des observations de la classe k dans le nœud.

  Voici un exemple :

	Aime Friends	Aime HIMYM	Aime PR
Individu 1	Non	Oui	Oui
Individu 2	Oui	Non	Non
Individu 3	Oui	Oui	Oui
Individu 4	Non	Oui	Oui
Individu 5	Oui	Non	Oui
Individu 6	Oui	Oui	Non

  On veut savoir si on fait un noeud sur $Friends$ ou sur $HIMYM$ pour déterminer PR.

  Commençons par calculer l'indice de Gini pour $Fiends$. On a deux sous ensembles dans cette feature : *oui* et *non*. Calcul pour le sous ensemble *oui* : $1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 = 0.5$. Calcul pour le sous ensemble *non* : $1 - (\frac{2}{2})^2 - (\frac{0}{2})^2 = 0$.

  L'indice de Gini final pour cette feature est l'indice de Gini pondéré : $\frac{4}{6} * 0.5 + \frac{2}{6} * 0 = \frac{1}{3}$.

  Pour la deuxième feature, on a les deux mêmes sous ensembles. Calcul pour le sous ensemble *oui* : $1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = 0.375$. Calcul pour le sous ensemble *non* : $1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$.

  L'indice de Gini final pour cette feature est l'indice de Gini pondéré : $\frac{4}{6} * 0.375 + \frac{2}{6} * 0.5 = \frac{5}{12}$.

  La feature que l'on choisi est celle de gain le plus faible : $Friend$.

- **Arrêt de la croissance** : La croissance de l'arbre s'arrête lorsque tous les points d'un nœud appartiennent à la même classe ou qu'un critère d'arrêt est atteint (par exemple, une profondeur maximale ou un nombre minimal d'observations par nœud).

- **Élagage (Pruning)** : Pour éviter le surapprentissage, on peut élaguer l'arbre en supprimant les branches qui contribuent peu à la réduction de l'erreur.

Feature Engineering

Pourquoi normaliser les caractéristiques ?

Certains algorithmes, comme les SVM, k-NN et les réseaux de neurones, sont sensibles à l'échelle des caractéristiques. La normalisation permet de s'assurer que toutes les caractéristiques contribuent de manière égale au modèle.

Techniques de normalisation

- **Standardisation** : Transforme les caractéristiques pour qu'elles aient une moyenne de 0 et un écart-type de 1.

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

où μ est la moyenne et σ l'écart-type de la caractéristique.

- **Min-Max Scaling** : Redimensionne les caractéristiques dans une plage fixe, généralement [0, 1].

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Gestion des variables catégorielles

Les algorithmes d'apprentissage automatique nécessitent généralement des caractéristiques numériques. Les variables catégorielles doivent donc être encodées.

Techniques d'encodage

- **One-Hot Encoding** : Crée des colonnes binaires pour chaque catégorie.

  Exemple : Si une variable catégorielle est "couleur" avec les catégories "rouge", "vert" et "bleu", le One-Hot Encoding créera trois colonnes binaires, chacune représentant une couleur (par exemple 00, 01 et 10).

- **Label Encoding** : Assigne un entier unique à chaque catégorie. Il faut faire attention, car une relation d'ordre est créée entre des catégories qui n'en n'ont pas forcément.

  Exemple : Pour la même variable "couleur", Label Encoding pourrait assigner 0 à "rouge", 1 à "vert" et 2 à "bleu".

- **Ordinal Encoding** : Utilise des entiers avec un ordre significatif.

  Exemple : Si la variable catégorielle est "taille" avec les catégories "petit", "moyen" et "grand", Ordinal Encoding pourrait assigner 0 à "petit", 1 à "moyen" et 2 à "grand".

Pourquoi réduire la dimensionnalité ?

La réduction de dimensionnalité permet d'éliminer les caractéristiques redondantes ou non pertinentes, réduisant ainsi le surapprentissage et améliorant l'interprétabilité du modèle.

Techniques de réduction de dimensionnalité

- **Filter Methods** : Sélectionnent les caractéristiques en fonction de tests statistiques (par exemple, chi-carré, corrélation).
- **Wrapper Methods** : Utilisent les performances du modèle pour sélectionner les caractéristiques (par exemple, élimination récursive de caractéristiques).
- **PCA (Principal Component Analysis)** : Projette les caractéristiques dans un espace de dimension inférieure tout en conservant la variance maximale.
- **t-SNE (t-Distributed Stochastic Neighbor Embedding)** : Technique non-linéaire pour la réduction de dimensionnalité, souvent utilisée pour la visualisation de données de haute dimension.

References

[1] Scikit-learn. (2025). Decision Trees. `https://scikit-learn.org/stable/modules/tree.html`

[2] StatQuest. (2025). Principal Component Analysis (PCA). `https://statquest.org/principal-component-analysis-pca/`

[3] Towards Data Science. (2025). Feature Engineering Techniques. `https://towardsdatascience.com/feature-engineering-techniques-36ds84f23423`

Linear Models

Logistic Regression

Idée générale

La régression logistique est un algorithme de classification pour les problèmes binaires ou multi-classes. Elle modélise la probabilité qu'une entrée donnée appartienne à une classe particulière.

Fonctionnement

- **Combinaison linéaire des caractéristiques** :

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

où β_0 est le biais, et $\beta_1, \beta_2, \ldots, \beta_n$ sont les poids associés aux caractéristiques $x_1, x_2, \ldots, x_n$.

- **Fonction sigmoïde** : La fonction sigmoïde (ou fonction logistique) transforme la combinaison linéaire en une probabilité comprise entre 0 et 1.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Cette fonction permet de modéliser la probabilité $P(y = 1|x)$ qu'une observation appartienne à la classe 1.

- **Entraînement** : L'entraînement de la régression logistique consiste à maximiser la fonction de log-vraisemblance :

$$\mathcal{L}(\beta) = \sum_{i=1}^{m} [y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i))]$$

où y_i est la vraie classe de l'observation i, et $\sigma(z_i)$ est la probabilité prédite.

Exemple

Supposons que nous voulons prédire si un étudiant sera admis dans une université en fonction de ses notes et de ses scores aux tests. La régression logistique peut être utilisée pour calculer la probabilité d'admission en fonction de ces caractéristiques.

Perceptron

Idée générale

Le Perceptron est un algorithme d'apprentissage supervisé utilisé pour les problèmes linéairement séparables en deux classes. Il calcule une somme pondérée des entrées et applique une fonction d'activation pour décider de la sortie.

Fonctionnement

- **Somme pondérée des entrées** :

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w}^T \mathbf{x}$$

  où w_0 est le biais, et $w_1, w_2, \ldots, w_n$ sont les poids associés aux entrées $x_1, x_2, \ldots, x_n$.

- **Fonction d'activation** : La fonction d'activation du Perceptron est une fonction de seuil :

$$f(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ -1 & \text{sinon} \end{cases}$$

- **Règle d'apprentissage de Rosenblatt** : Les poids sont mis à jour en utilisant la règle suivante :

$$w_i \leftarrow w_i + \alpha(y - \hat{y}) x_i$$

  où α est le taux d'apprentissage, y est la vraie classe, et $\hat{y}$ est la classe prédite.

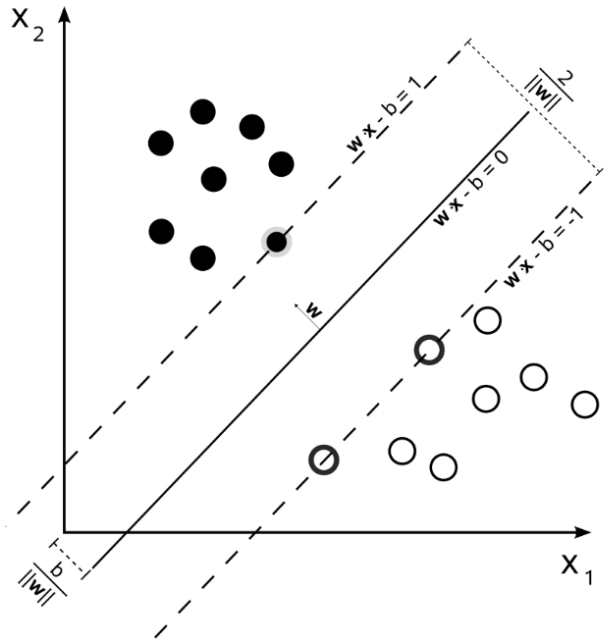
Exemple

Supposons que nous voulons classer des fleurs en deux catégories en fonction de la longueur et de la largeur de leurs pétales. Le Perceptron peut apprendre une frontière de décision linéaire pour séparer ces deux catégories.

Large Margin Classifiers

Idée générale

Les classifieurs à grande marge sont des modèles qui cherchent à maximiser la marge entre les classes. Ils sont à la base des machines à vecteurs de support (SVM).



Fonctionnement

- **Hyperplan séparateur** : Un hyperplan est défini par l'équation $\mathbf{w}^T\mathbf{x} + b = 0$. L'objectif est de trouver l'hyperplan qui maximise la marge, c'est-à-dire la distance minimale entre les points de données et l'hyperplan.
- **Optimisation** : Le problème d'optimisation consiste à minimiser $\|\mathbf{w}\|^2$ sous les contraintes :

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \text{pour } i = 1, \ldots, N$$

  où y_i est la classe de l'observation $\mathbf{x}_i$.
- **Vecteurs de support** : Les points de données qui se trouvent sur les marges sont appelés vecteurs de support. Ils sont critiques pour définir l'hyperplan optimal.

Exemple

Supposons que nous avons deux classes de points dans un espace 2D. Un classifieur à grande marge trouvera la ligne qui sépare ces deux classes avec la plus grande marge possible.

Support Vector Machines (SVM)

Idée générale

Les machines à vecteurs de support (SVM) sont une extension des classifieurs à grande marge. Elles utilisent le "kernel trick" pour traiter les problèmes non-linéairement séparables.

Fonctionnement

- **Kernel Trick** : Le "kernel trick" permet de projeter les données dans un espace de plus haute dimension où elles deviennent linéairement séparables. Les noyaux (kernels) couramment utilisés incluent le noyau linéaire, polynomial, et gaussien.
- **Classification multi-classes** : Pour les problèmes de classification multi-classes, les SVM utilisent des stratégies comme "one-vs-one" ou "one-vs-all".

Exemple

Supposons que nous voulons classer des images de chiffres manuscrits. Les SVM avec un noyau gaussien peuvent être utilisées pour apprendre les frontières de décision non-linéaires entre les différentes classes de chiffres.

References

[1] Scikit-learn. (2025). Logistic Regression. `https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`

[2] SVM Guide. (2025). Understanding Support Vector Machines. `https://www.svm-tutorial.com/`

[3] Wikipedia. (2025). Perceptron. `https://en.wikipedia.org/wiki/Perceptron`

Advanced Topics & Practical Considerations

Ensemble Methods

Instabilité et Forêts Aléatoires

Instabilité : Les arbres de décision sont sensibles aux petites variations dans les données d'entraînement. Un léger changement peut entraîner une structure d'arbre complètement différente, ce qui conduit à une grande variance. D'où l'intéret des random forests.

Comparaison :

- **SHAP** : Cohérent et théoriquement solide ; calcule l'importance des caractéristiques de manière globale et locale.
- **LIME** : Rapide et flexible ; se concentre sur les explications locales.

Recherche sur Grille (Grid Search) :

Comment ça marche :

- Définir une grille de valeurs d'hyperparamètres.
- Entraîner et évaluer le modèle pour chaque combinaison.
- Sélectionner la combinaison avec le meilleur score de validation croisée.

Recherche Aléatoire (Randomized Search) :

Comment ça marche :

- Définir une distribution pour chaque hyperparamètre.
- Échantillonner aléatoirement des combinaisons et les évaluer.
- Sélectionner la combinaison la mieux performante.

Métriques pour la classification

Recall / Sensitivity : $\frac{TP}{TP+FN}$ (combien de bons positifs par rapport à tous ceux qui sont sensé être positifs, à quel point il vise juste pour les positifs).
Specificity : $\frac{TN}{TN+FP}$ (combien de bons négatifs par rapport à tous ceux qui sont sensé être négatis, à quel point il vise juste pour les négatifs).
Precision : $\frac{TP}{TP+FP}$ (combien de bons positifs sur l'ensemble des classés positifs, à quel point il peut se tromper dans les positifs).

References

[1] Scikit-learn. (2025). Ensemble Methods. `https://scikit-learn.org/stable/modules/ensemble.html`

[2] SHAP. (2025). SHAP Documentation. `https://shap.readthedocs.io/en/latest/`

[3] LIME. (2025). LIME Documentation. `https://lime-ml.readthedocs.io/en/latest/`

[4] Scikit-learn. (2025). Hyperparameter Tuning. `https://scikit-learn.org/stable/modules/grid_search.html`