Sommaire

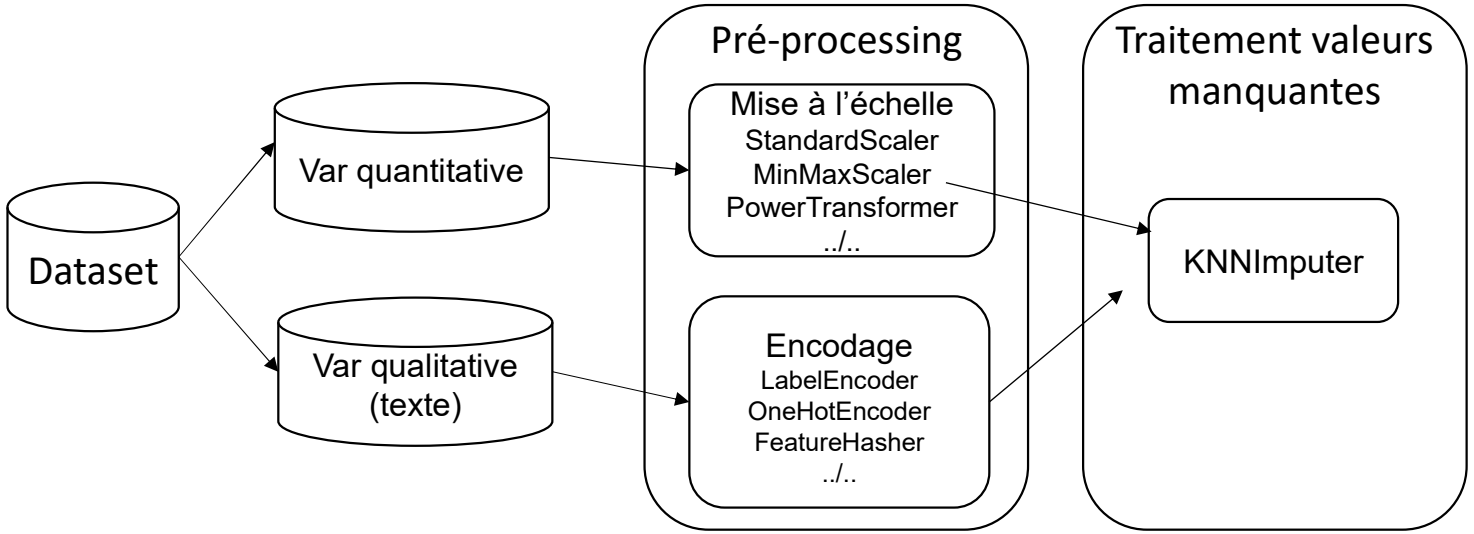
1 Imputation

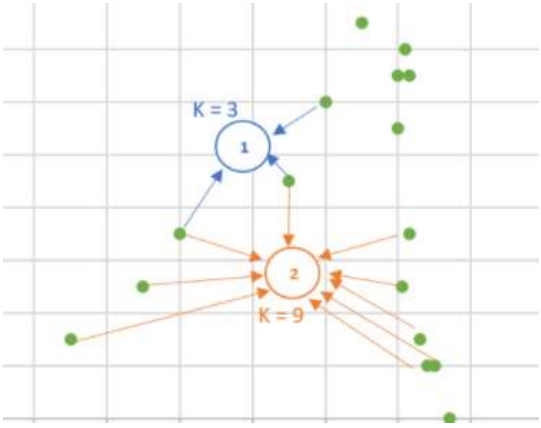
1. Imputation - KNNImputer

Basé sur l'algorithme K-Nearest Neighbors

Pour chaque valeur manquante d'un point de données :

1. KNN Imputer **cartographie** l'ensemble de données à l'exclusion des éléments ayant des valeurs manquantes dans l'espace de coordonnées à n dimensions
2. **Calcule** la **distance** euclidienne (par défaut) des **points** les **plus proches** de ce point de données.
3. **Imputation** des valeurs manquantes par la **moyenne** des éléments pertinents pour ces points les plus proches





1. Imputation - KNNImputer

- Encodage pour var. qualitatives.
- Mise à l'échelle pour les variables quantitatives.
- Inversion de la mise à l'échelle pour avoir les vraies valeurs.
- Basé sur hyper-paramètre k ➔ tri-it-all
- Hypothèse de relations entre les entités
- Mauvaises prédictions si prédicteurs faibles ou fortes relations entre les entités
- Sujet à la malédiction (fléau) de la dimensionnalité

https://ichi.pro/fr/missforest-le-meilleur-algorithme-d-imputation-des-donnees-manquantes-5291693485779

- Imputer des valeurs autres que constantes (numériques ou modalités, moyenne, mode ou médiane.

NaNimputer?
Verstack

MissForest ?
Appliquée sur données mixtes,
Encodage, pas mise à l'échelle
Pas d'hypothèse de relation entre entités
Robuste aux bruits et à la mutlicolinéarité
Non paramétrique : pas de réglages
Grande dimension
Mais plus long petits datasets

2 Normalisation

2. Normalisation : pourquoi ?

- Mettre sur une même échelle toutes les données quantitatives
- Rendre les plages cohérentes entre les variables
- Avoir une influence similaire des variables sur les modèles
- Conserver les rapports de distance
- Améliorer les performances et stabiliser le modèle

2. Normalisation : comment ?

Scaler	Caractéristiques
Min Max Scaler sklearn.preprocessing.MinMaxScaler	Std faible, distrib. non normale, [0,1] Sensible oultiers
Standard Scaler sklearn.preprocessing.StandardScaler	Std 1, distrib. Normale centrée en 0 Sensible oultiers (Z-Score)
Robust Scaler sklearn.preprocessing.RobustScaler	Std faible, distrib. non gaussienne Insensible oultiers
Max Abs Scaler sklearn.preprocessing.MaxAbsScaler	Distrib. Éparse,centrée en 0, [-1,1] Insensible oultiers
Power Transformer Scaler sklearn.preprocessing.PowerTransformer	trouve le facteur d'échelle optimal pour stabiliser la variance et minimiser l'asymétrie grâce à l'estimation du maximum de vraisemblance. rendre les données plus gaussiennes.
Quantile Transformer Scaler sklearn.preprocessing.QuantileTransformer	1. Calcule la fonction de distribution cumulative de la variable 2. Il utilise ce cdf pour faire correspondre les valeurs à une distribution normale 3. Établit une correspondance entre les valeurs obtenues et la distribution de sortie souhaitée à l'aide de la fonction quantile associée Distrib. Normale, large dataset Insensible oultiers
Unit Vector Scaler sklearn.preprocessing.normalize	Mise à l'échelle se fait en considérant que le vecteur d'élément entier est de longueur unitaire. [0,1]
Normalizer Scaler sklearn.preprocessing.Normalizer	Normalise selon la norme L1 (manhatan) ou L2 (euclidienne)

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

$$x_{new} = \frac{x - \mu}{\sigma}$$

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

$$x' = \frac{x}{||x||}$$

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad \sum_{i=1}^{n}|x_i - y_i|$$

2. Normalisation : algorithme ?

Calculer un vecteur de poids

- Scoring

Descente de gradient

- Logistic Regression
- Linear Regression
- Réseaux de neurones

Calculer des distances pour déduire le degrée de similarité de deux items

- Support Vector Machines (SVM)
- K-Nearest Neighbors (KNN)
- K-Means (clustering…)
- Principal Component Analysis (PCA)

Tree-base algorithm :

- Gradient Boosted Decision Trees
- Regression Tree
- Classification Trees
- Random Forests

Linear Discriminant Analysis(LDA)

Naive Bayes

3 Encodage

3. Encodage

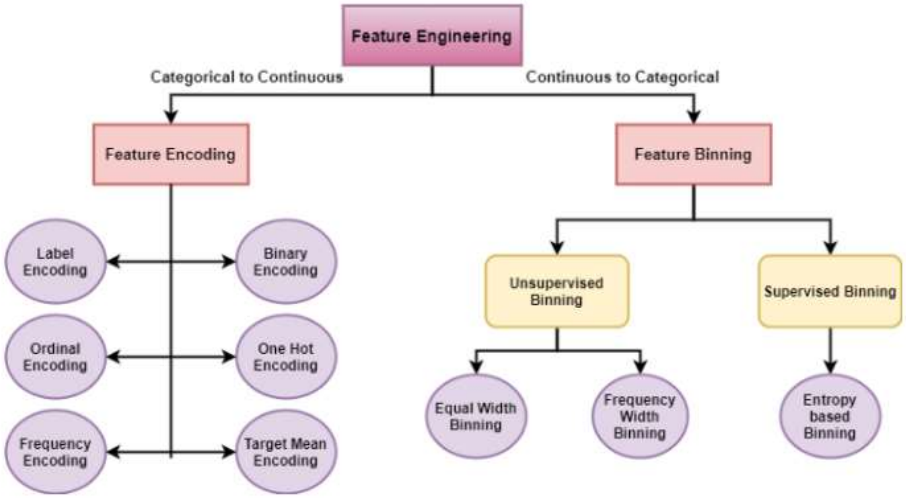
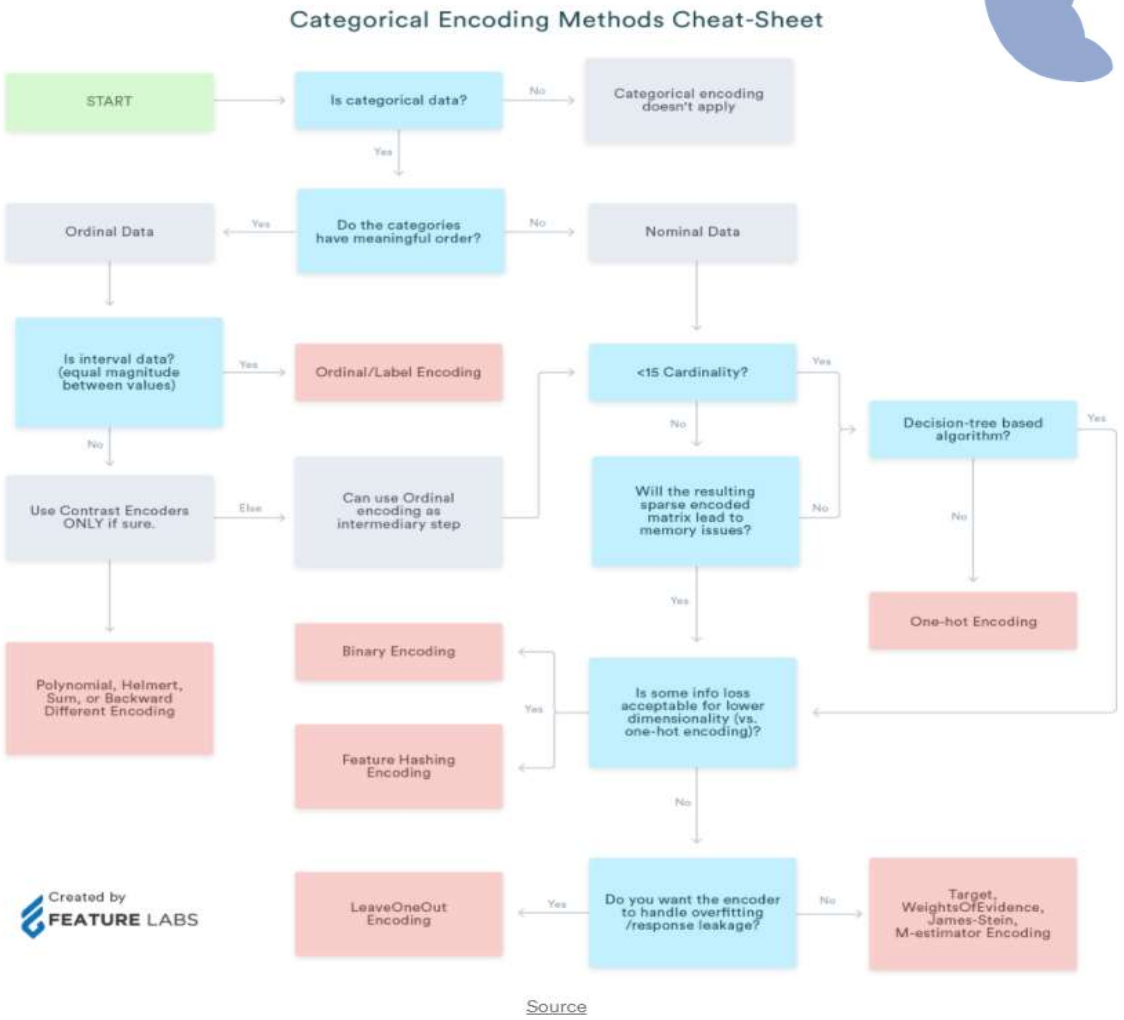


Image by Author

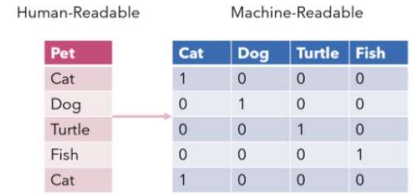


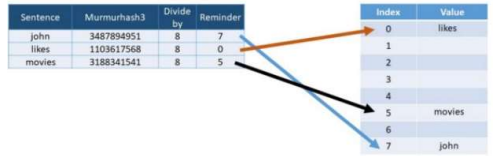
Source

* : sklearn.preprocessing - ** : category_encoders

3. Encodage

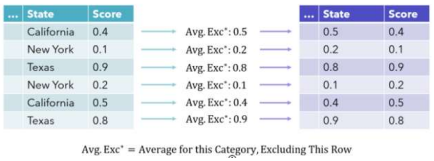
Encodeur	Description	+/-
* OneHotEncoder	connu sous le nom de variables fictives, est une méthode de conversion de variables catégorielles en plusieurs colonnes binaires, où un 1 indique la présence de cette ligne appartenant à cette catégorie.	- : augmente la dimensionnalité mais peu d'information (beaucoup 0), les nouvelles variables en relation linéaire les unes avec les autres ➔ pb parallélisme et multi-colinéarité
** HashingEncoder	Convertir les données en un vecteur de caractéristiques. Cette opération s'effectue à l'aide d'un hachage. Nous appelons cette méthode "hachage de caractéristiques" ou "l'astuce du hachage". Replace var_cat with col_0 to col_nb_vecteur [index 0 to nb_vecteur] which contains either 1 or 0.	+ : si utilisation des modèles d'arbres avec de nombreux niveaux différents. - : interprétabilité difficile de la contribution de chacun de vos niveaux.
* OrdinalEncoder	Nous effectuons un encodage ordinal pour nous assurer que l'encodage des variables conserve la nature ordinale de la variable. Ceci n'est raisonnable que pour les variables ordinales	- : Pas utiliser pour des variables non ordinales
** TargetEncoder	L'encodage de la cible est un moyen très efficace de représenter une colonne catégorique et n'occupe que l'espace d'une seule caractéristique. Également connu sous le nom d'encodage de la moyenne, chaque valeur de la colonne est remplacée par la valeur cible moyenne pour cette catégorie. Cela permet une représentation plus directe de la relation entre la variable catégorielle et la variable cible	- : rend plus difficile pour le modèle l'apprentissage des relations entre une variable codée en moyenne et une autre variable, rend plus difficile pour le modèle l'apprentissage des relations entre une variable codée en moyenne et une autre variable





3. Encodage

Encodeur	Description	+/-
** LeaveOneOutEncoder	Cette méthode est très similaire à l'encodage des cibles, mais exclut la cible de la rangée actuelle lors du calcul de la cible moyenne pour un niveau afin de réduire l'effet des valeurs aberrantes. Comme le modèle est exposé non seulement à la même valeur pour chaque classe encodée, mais aussi à une plage, il apprend à mieux généraliser.	- Puisque chaque valeur de la catégorie est remplacée par la même valeur numérique, le modèle peut avoir tendance à s'adapter de manière excessive aux valeurs encodées qu'il a vue
** CatBoostEncoder	effectue le codage CatBoost pour les caractéristiques catégorielles. Il prend en charge les types de cibles suivants : binaires et continues. Pour le support des cibles polynomiales, il utilise un PolynomialWrapper. Il s'agit d'un codage très similaire à celui du leave-one-out, mais qui calcule les valeurs "à la volée". Par conséquent, les valeurs varient naturellement pendant la phase d'apprentissage et il n'est pas nécessaire d'ajouter du bruit aléatoire.	
** PolynomialEncoder	Le codage polynomial est une forme d'analyse des tendances dans la mesure où il recherche les tendances linéaires, quadratiques et cubiques de la variable catégorielle. Ce type de système de codage ne doit être utilisé qu'avec une variable ordinale dans laquelle les niveaux sont également espacés.	
** JamesSteinEncoder	Comme TargetEncoder. L'encodeur James-Stein évalue la quantité de variation dans les exemples de cette catégorie et la compare à la variation sur l'ensemble des données.	- Distribution normale - S'il n'y a que quelques exemples par catégorie, cette technique ne sera pas particulièrement utile. Nous devons également être conscients que nous devons "sauvegarder" la moyenne du groupe pour chaque catégorie



* : sklearn.preprocessing - ** : category_encoders

3. Encodage

Encodeur	Description	+/-
** HelmertEncoder	Pour la valeur de la caractéristique, l'estimateur de James-Stein renvoie une moyenne pondérée de : La valeur cible moyenne pour la valeur de la caractéristique observée. La valeur cible moyenne (indépendamment de la valeur de la caractéristique). L'encodeur James-Stein rétrécit la moyenne vers la moyenne générale. Il s'agit d'un codeur basé sur la cible.	

* : sklearn.preprocessing - ** : category_encoders

4

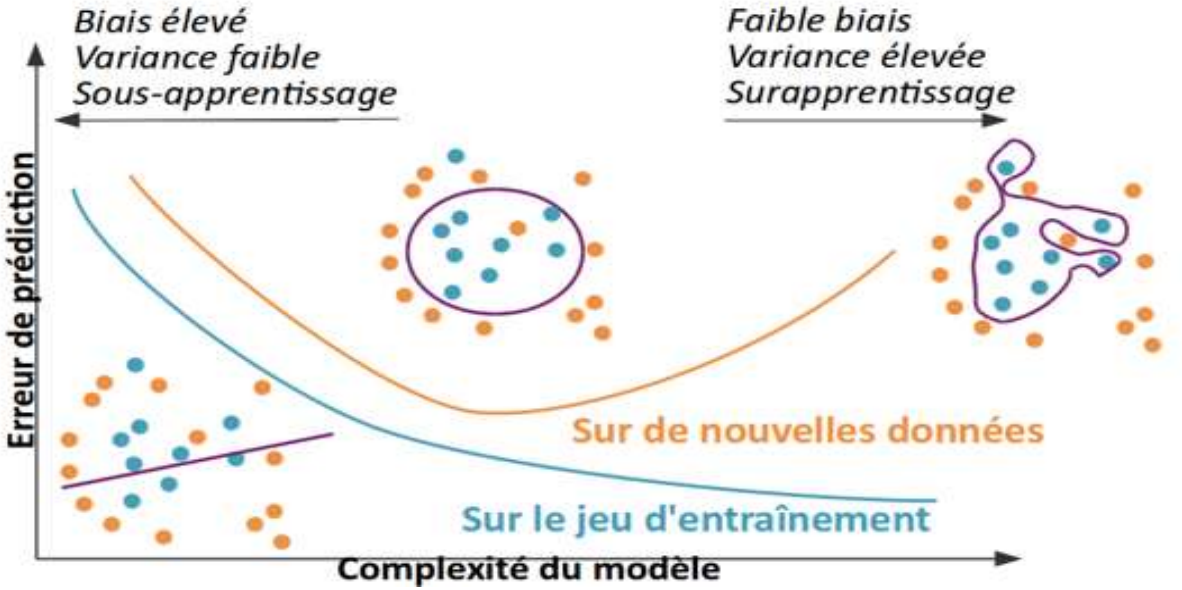
Bias-variance trade off

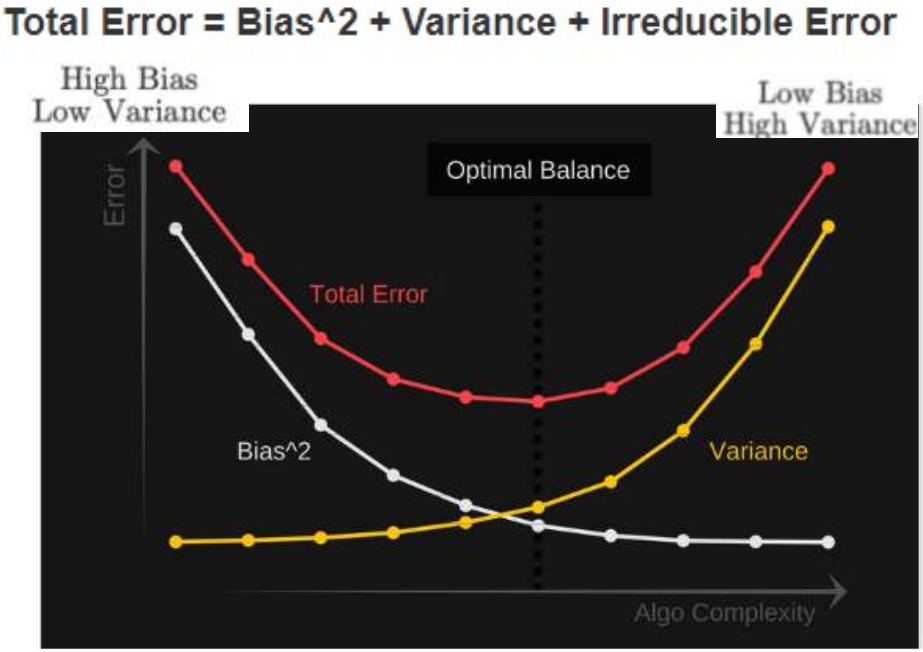
4. Bias-variance trade off

problème de **minimiser** simultanément **deux sources d'erreurs de prédiction** qui empêchent les algorithmes d'apprentissage supervisé de généraliser au-delà de leur échantillon d'apprentissage :

- Le **biais** = différence entre la prédiction moyenne de notre modèle et la valeur correcte que nous essayons de prédire.
- La **variance** = erreur due à la sensibilité aux petites fluctuations de l'échantillon d'apprentissage

➔ **compromis**



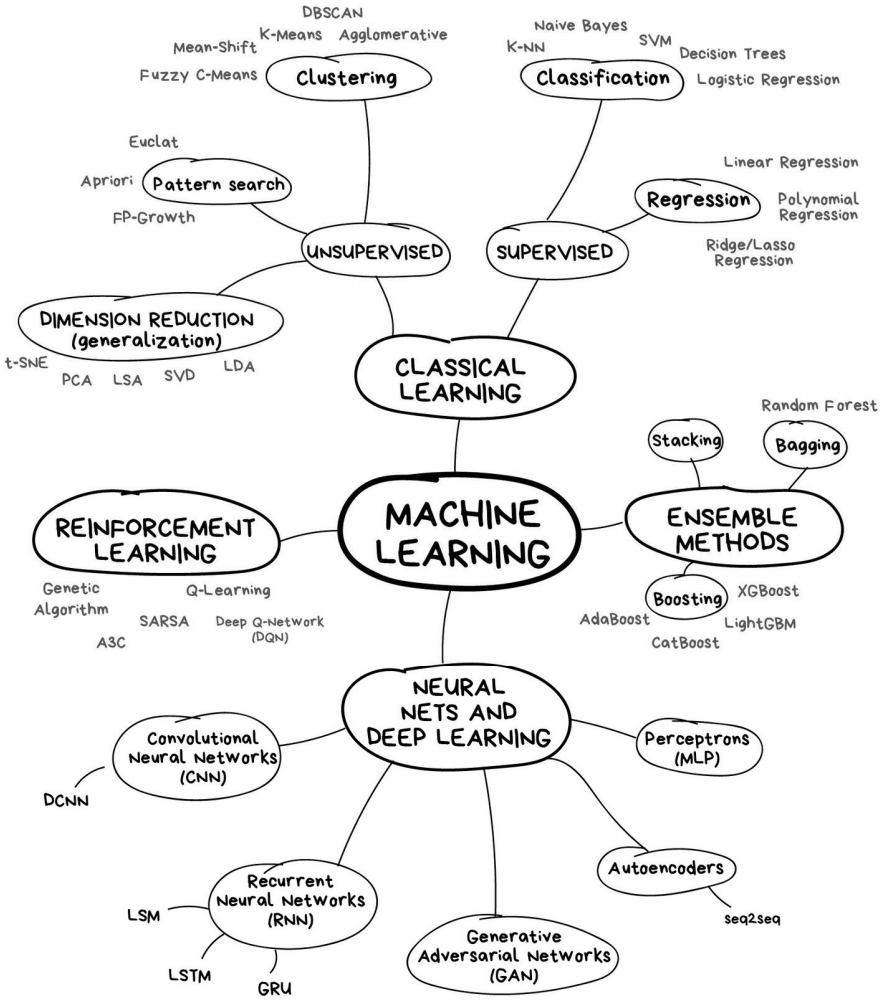


5

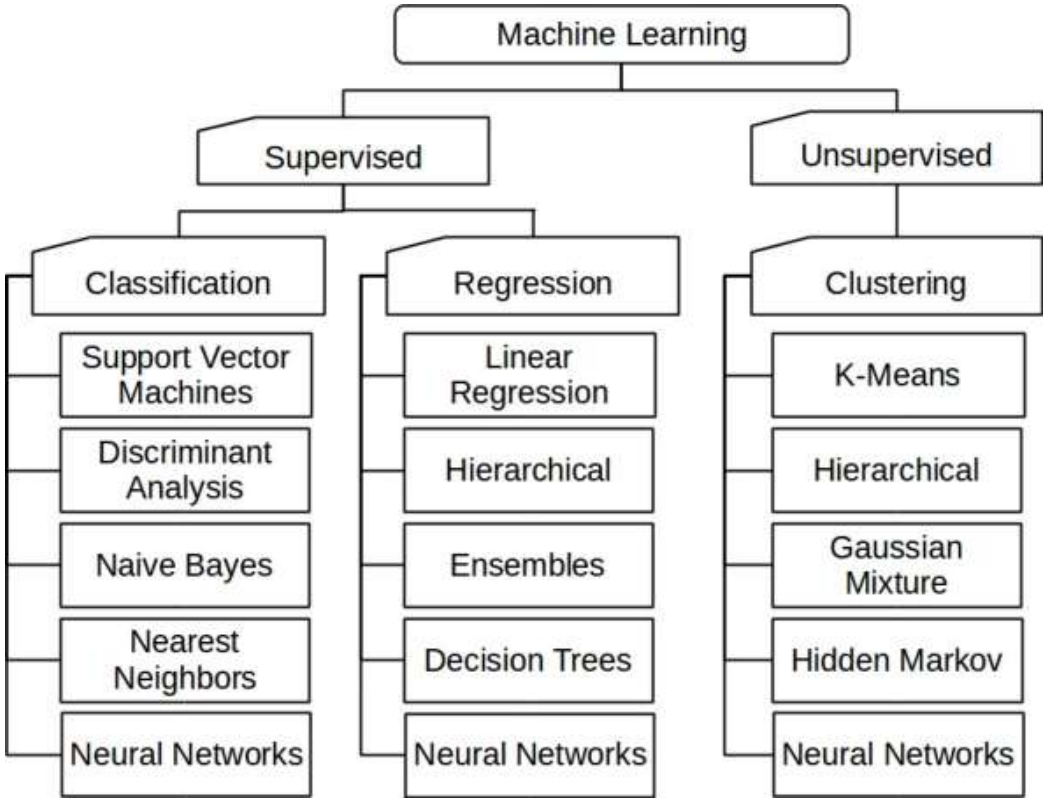
Modèles

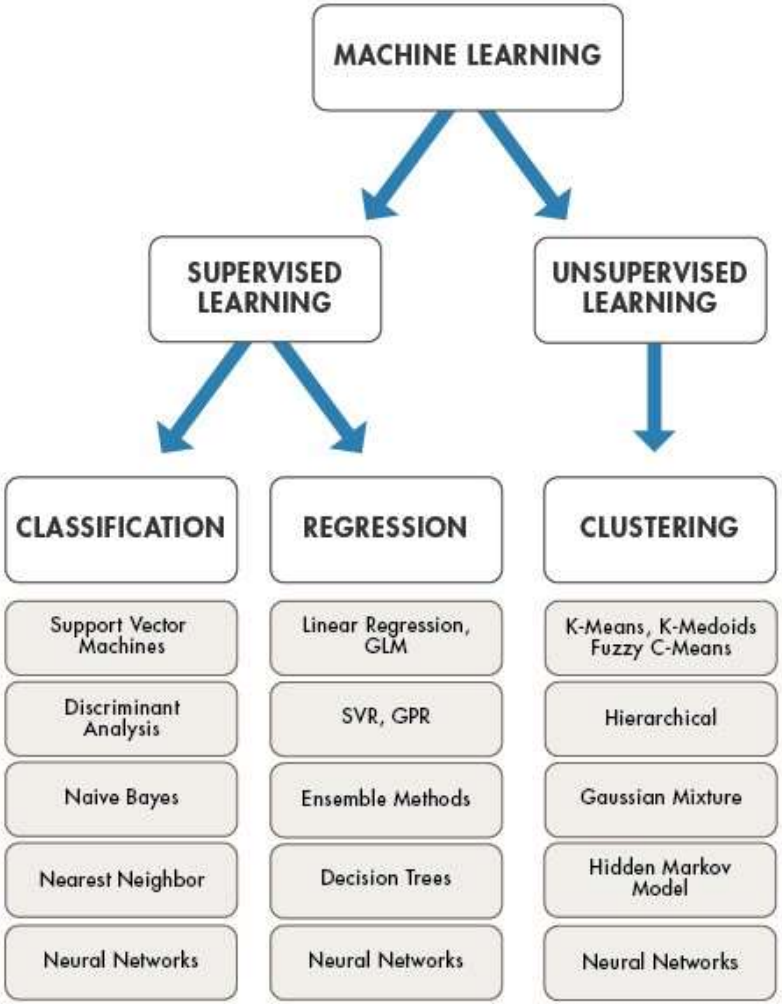
Modèles
Généralités

5. Modèles - Sheet



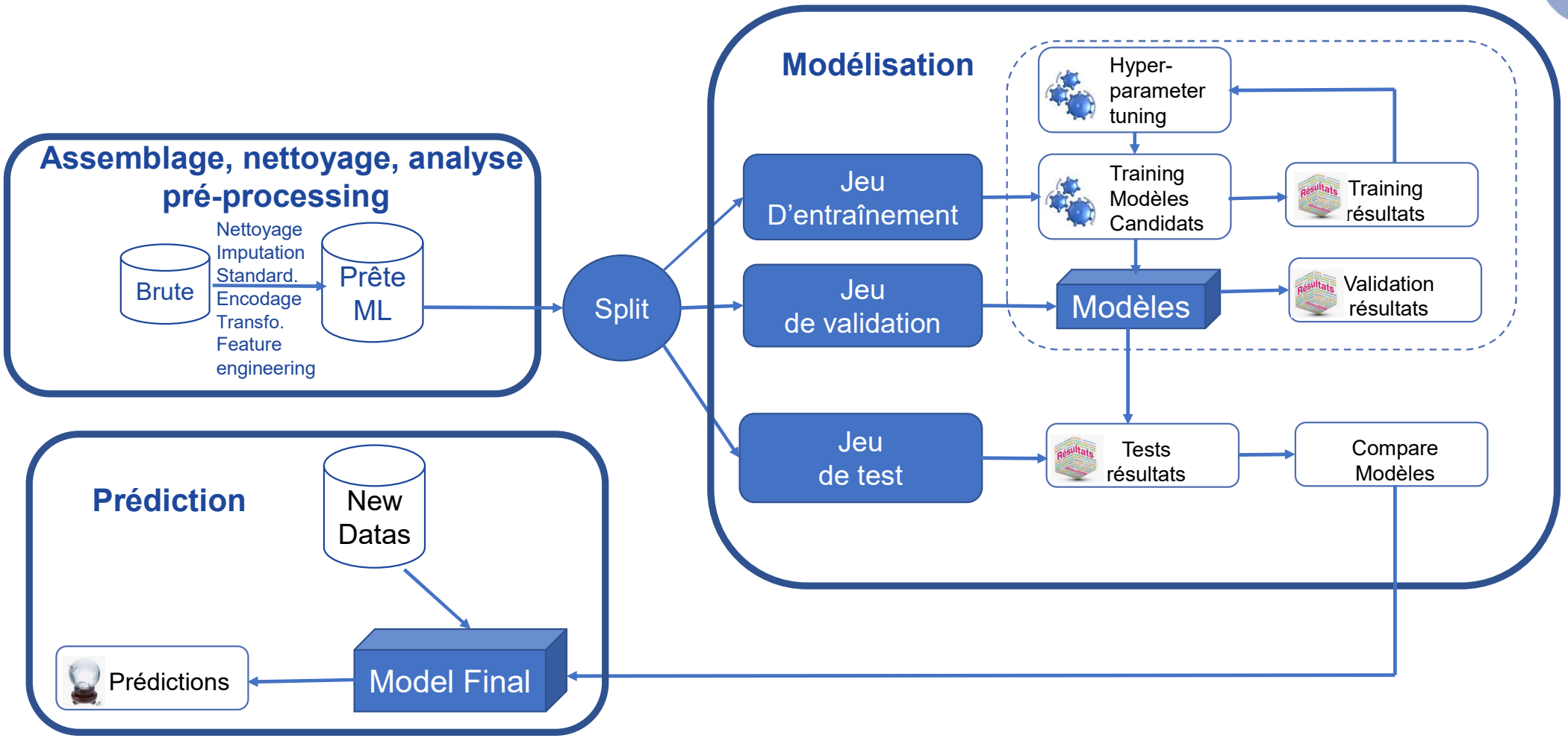
5. Modèles - Sheet





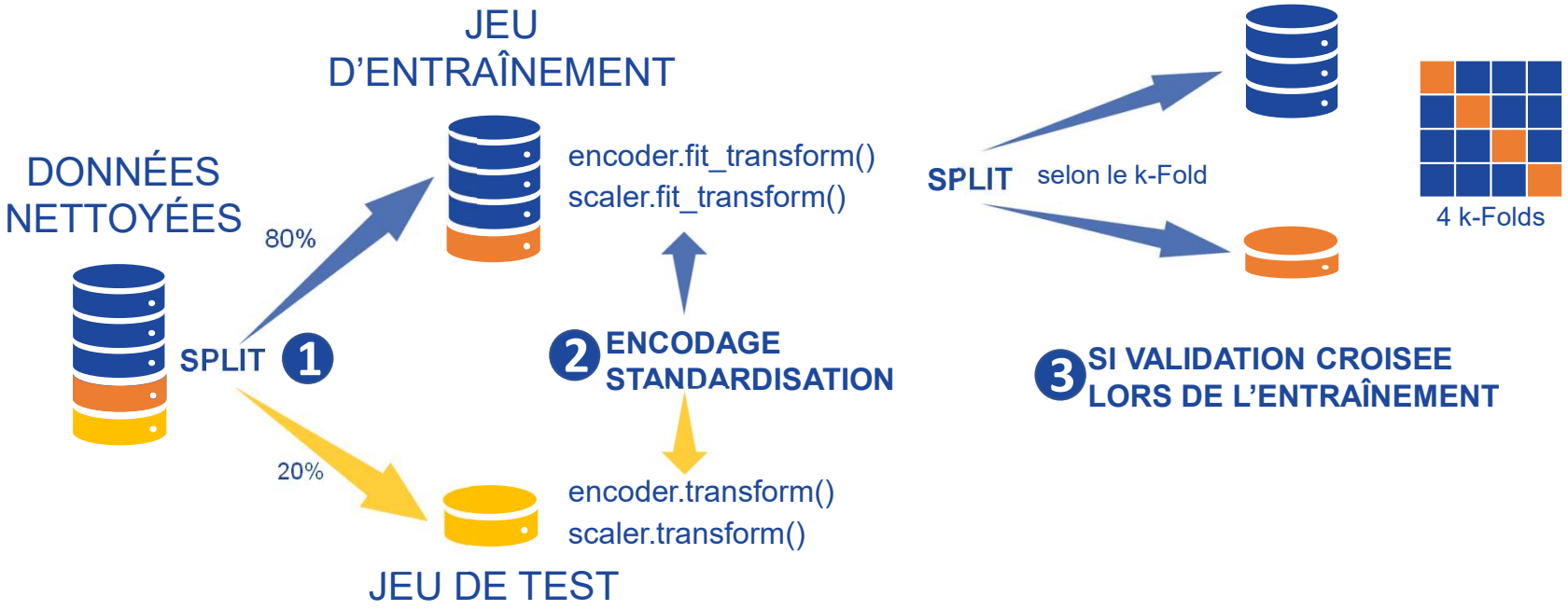
5. Modèles - Sheet

5. Modèles - Généralités

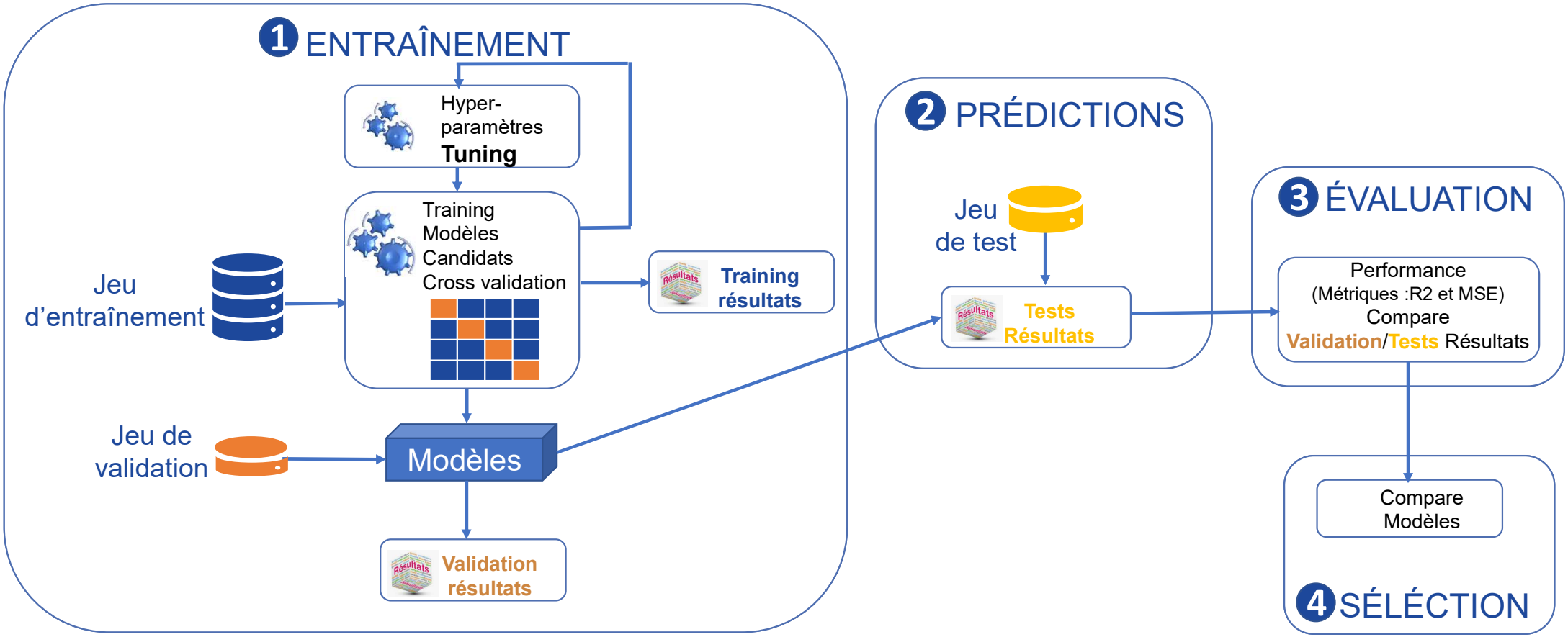


5. Modèles - Généralités

SPLIT - ENCODAGE/STANDARDISATION : préparation des données au machine learning



5. Modèles - Généralités



Modèles SKLEARN

5. Modèles - sklearn

Modèle	Package	Description	Gère Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularization	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
LinearRegression	linear	Algorithme des moindres carrés ordinaires							Hautement interprétable, aucun biais introduit	Les données sont composées de peu de valeurs aberrantes La variance entre les étiquettes de sortie est faible Toutes les caractéristiques d'entrée sont non seulement indépendantes mais aussi non corrélées.	Facilité d'interprétation des résultats Faible niveau de complexité	Risque de multicolinéarité si les caractéristiques d'entrée sont corrélées. De petites erreurs ou valeurs aberrantes dans les valeurs cibles peuvent avoir un impact considérable sur le modèle
Ridge	linear RidgeCV	Les moindres carrés ordinaires avec un terme de régularisation L2 ajouté. Le poids du terme de régularisation est contrôlé par un argument alpha supplémentaire.	Oui						Prévention de l'overfitting	Les données sont composées de quelques valeurs aberrantes Il peut y avoir une certaine corrélation entre les caractéristiques d'entrée. Éviter le surajustement On veut utiliser toutes les caractéristiques pour faire des prédictions.	Facile à interpréter les résultats Moins affecté par les caractéristiques corrélées Prévient le surajustement et diminue la variance des poids appris	En réduisant la variance, il incorpore un certain degré de biais dans le modèle. Il peut être difficile d'ajuster alpha pour atteindre un équilibre souhaitable entre les MCO et le terme de régularisation
Lasso	linear LassoCV lasso_path LassoLarsCV LassoLarsIC MultiTaskLasso	Algorithme des moindres carrés ordinaires avec ajout d'un terme de régularisation L1. Le poids du terme de régularisation est contrôlé par un argument alpha supplémentaire. Le poids du terme MCO est inversement proportionnel au nombre d'échantillons.	Oui		Oui				Seules quelques caractéristiques sont importantes, sélection des caractéristiques	Les données sont composées de quelques valeurs aberrantes Il peut y avoir une certaine corrélation entre les caractéristiques d'entrée. Vous souhaitez donner plus de poids à quelques caractéristiques importantes et ne pas ignorer les caractéristiques moins importantes. Sélection des caractéristiques	Réduit le nombre de caractéristiques que le modèle utilise Peut gérer de grandes quantités de caractéristiques Peut souvent être utilisé comme un test préliminaire pour effectuer une sélection de caractéristiques	En réduisant la variance, incorpore un certain degré de biais dans le modèle. Il peut être difficile de régler alpha pour atteindre un équilibre souhaitable entre les MCO et le terme de régularisation

5. Modèles - sklearn

Modèle	Package	Description	Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularization	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
ElasticNet	linear ElasticNetCV MultiTaskElasticNet	Moindres carrés ordinaires avec un terme de régularisation L1 et L2. Les poids des termes de régularisation L1 et L2 sont contrôlés par un paramètre l1_ratio.	Oui		Oui				Mélange de Ridge et Lasso	Les données sont composées de quelques valeurs aberrantes Il peut y avoir une certaine corrélation entre les caractéristiques d'entrée. Éviter le surajustement Sélection des caractéristiques	Incorporer les capacités de sélection de caractéristiques de Lasso avec les capacités de régularisation de Ridge.	En réduisant la variance, incorpore un certain degré de biais dans le modèle. Il peut être difficile de régler l'alpha pour atteindre un équilibre souhaitable entre les MCO et les termes de régularisation. Coût de calcul plus élevé que Ridge ou Lasso
Lars (Least Angle Regression)	linear lars_path lars_path_gram	Une régression pas à pas qui modifie les poids des caractéristiques les plus corrélées à l'étiquette de sortie à chaque étape. Au fur et à mesure des étapes, d'autres caractéristiques deviennent tout aussi corrélées. Lorsque cela se produit, l'algorithme se met à jour pour se déplacer dans la direction déterminée par les angles commutatifs des poids de ces caractéristiques.			Oui				Plus de caractéristiques que d'échantillons de données	Plus de caractéristiques d'entrée que d'échantillons de données Les données comportent peu de valeurs aberrantes Peu de variance entre les étiquettes de sortie	Les poids des caractéristiques également corrélées sont mis à jour à des taux similaires. Efficacité numérique et informatique pour de nombreuses caractéristiques. Résultats et étapes de formation interprétables	Le bruit ou de petites erreurs ou valeurs aberrantes dans les valeurs cibles peuvent avoir un impact considérable sur le modèle
OrthogonalMatchingPursuit	linear orthogonal_mp	Une régression par étapes vers l'avant qui permet à l'utilisateur de spécifier un nombre maximal de coefficients non nuls ou une valeur d'erreur cible à atteindre.			Oui				Sélection des caractéristiques	Vous cherchez à sélectionner les N caractéristiques les plus importantes pour représenter un modèle composé de plusieurs caractéristiques	Peut déterminer les caractéristiques les plus étroitement liées aux étiquettes. Peut spécifier le nombre exact de coefficients non nuls à retourner	Applications limitées Non utilisé très souvent (et donc pas beaucoup de ressources en ligne)

5. Modèles - sklearn

Modèle	Package	Description	Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularization	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
BayesianRidge	linear	Similaire à Ridge, mais le paramètre de régularisation est réglé pour s'adapter aux données pendant le processus de formation.	Oui			Oui			Ridge mais ne veut pas fixer de constante de régularisation	Vous recherchez des résultats similaires à ceux de Ridge, mais vous êtes prêt à sacrifier l'interprétabilité pour gagner du temps en évitant de tester différents poids de régularisation.	Aucun besoin de régler la valeur alpha s'adapte bien aux données disponibles	Des résultats moins interprétables
ARDRegression	linear	BayesianRidge avec des valeurs de poids plus espacées. Presque comme une version de BayesianLasso.	Oui		Oui	Oui			Lasso mais pas de constante de régularisation.	Vous recherchez des résultats similaires à ceux de Lasso, mais vous êtes prêt à sacrifier l'interprétabilité pour gagner du temps en évitant de tester différents poids de termes de régularisation.	Il n'est pas nécessaire de régler la valeur alpha. S'adapte bien aux données disponibles Réduit le poids des caractéristiques non importantes	Résultats moins interprétables Coûteux en termes de calcul (ne peut pas traiter de très grands ensembles de données)
SGDRegressor	linear	Ajusté en minimisant une perte empirique régularisée avec une descente de gradient stochastique (SGD). Le type de fonction de perte utilisé peut être changé en utilisant l'argument "loss".			Oui		Oui		Le nombre de caractéristiques et d'échantillons est très élevé	Avoir un très grand ensemble de données et vouloir obtenir rapidement des résultats. Beaucoup de flexibilité pour changer entre différentes fonctions de perte	Fonctionne bien sur de grandes quantités de données et de fonctions Fonction personnalisable qui offre différentes fonctions de perte	Résultats moins interprétables -Beaucoup de paramètres à régler
PassiveAggressiveRegressor	linear	Ajusté en utilisant une variante de la perte charnière. Efficace pour l'apprentissage à grande échelle avec plusieurs points de données et caractéristiques.			Oui		Oui		Le nombre de caractéristiques et d'échantillons est très élevé	Vous disposez d'un très grand ensemble de données et vous souhaitez obtenir rapidement des résultats.	Fonctionne bien sur de grandes quantités de données et de fonctionnalités	Résultats moins interprétables Ne sont pas très utilisés (et donc peu de ressources en ligne)
RANSACRegressor	linear	Un modèle linéaire conçu pour traiter les valeurs aberrantes dans les données et/ou les données corrompues. Ajuste de manière itérative un modèle linéaire à un sous-ensemble de données. Si le modèle s'adapte mieux aux données en ligne que le modèle précédent, il est enregistré comme le meilleur modèle. Cette opération est effectuée pendant un nombre spécifié d'itérations ou jusqu'à ce qu'un critère d'arrêt soit rempli.		Oui					Valeurs aberrantes dans les étiquettes de l'ensemble de données	Vous avez des valeurs aberrantes dans les étiquettes de votre ensemble de données.	Il s'adapte mieux aux valeurs aberrantes de grande taille dans la direction Y que les autres algorithmes de calcul des valeurs aberrantes. Plus rapide que TheilSen et s'adapte beaucoup mieux à un grand nombre d'échantillons	S'effondre avec un grand nombre de caractéristiques d'entrée. -Ignore toutes les données qu'il considère comme aberrantes

5. Modèles - sklearn

Modèle	Package	Description	Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularization	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
TheilSenRegressor	linear	Un modèle linéaire conçu pour traiter les valeurs aberrantes dans les données et/ou les données corrompues. Calcule les solutions des moindres carrés sur un certain nombre de sous-échantillons de données, puis détermine la médiane L1 de ces calculs pour choisir le meilleur modèle.		Oui					Valeurs aberrantes dans les caractéristiques de l'ensemble de données	Il y a des valeurs aberrantes dans les caractéristiques de votre ensemble de données.	mieux gérer les valeurs aberrantes de taille moyenne dans la direction X	S'effondre avec un grand nombre de caractéristiques d'entrée. Ignore toutes les données qu'il considère comme aberrantes
HuberRegressor	linear	Un modèle linéaire conçu pour traiter les valeurs aberrantes des données et/ou les données corrompues. Il n'ignore pas les valeurs aberrantes, mais leur accorde plutôt un poids plus faible.		Oui					Données aberrantes et recherche de l'algorithme le plus rapide	Vous voulez des analyses rapides des données en ignorant les valeurs aberrantes.	Plus rapide que RANSAC et TheilSen (tant que le nombre d'échantillons n'est pas trop grand) N'ignore pas complètement les points de données qu'il considère comme aberrants	S'effondrer avec un grand nombre de caractéristiques d'entrée
DecisionTreeRegressor	tree	Développe un arbre de décision en divisant sur les caractéristiques d'entrée. Très similaire à un arbre de décision typique, sauf que les nœuds de sortie correspondent à des sorties linéaires.						Oui	Données non linéaires regroupées dans des buckets	Les données ne sont pas linéaires et sont plutôt composées de ""buckets"". Nombre d'échantillons > nombre de caractéristiques Il y a des caractéristiques dépendantes dans les données d'entrée. DTR gère bien ces corrélations.	Peut exporter la structure de l'arbre pour voir sur quelles caractéristiques l'arbre se divise. Gère bien les données éparses et corrélées. Possibilité d'ajuster le modèle pour résoudre les problèmes d'overfitting	Risque d'ajustement excessif, surtout lorsque le nombre de caractéristiques est proche ou supérieur au nombre d'échantillons.
GaussianProcessRegressor	gaussian_process	Crée un modèle en prenant une distribution sur un certain nombre de fonctions ajustées aux données. Met à jour le poids de ces différentes fonctions en utilisant la règle de Baye.						Oui	Données non linéaires, pas sûr de la structure des données	Lorsque les données sont non linéaires ou que vous n'êtes pas sûr de la structure des données et que vous voulez un modèle qui s'adapte bien aux données d'entrée. Ne sont pas concernés par l'overfitting"	S'adapte très bien aux données de structures diverses	Très enclin à l'ajustement excessif et plus difficile à éviter que DecisionTreeRegressor. -Très coûteux en termes de calcul

5. Modèles - sklearn

Modèle	Package	Description	Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularization	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
MLPRegressor	neural_network	Fonctionne comme un réseau neuronal avec plusieurs neurones à chaque couche et des fonctions d'activation non linéaires entre chaque couche. Un certain nombre de paramètres peuvent être réglés pour obtenir des résultats optimaux.			Oui			Oui	Données non linéaires, beaucoup de caractéristiques importantes	Les données se composent de plusieurs caractéristiques et de modèles linéaires de lutte Toutes ou la plupart des caractéristiques d'entrée sont importantes pour faire des prédictions".	Peut apprendre des modèles en temps réel Les réseaux neuronaux peuvent obtenir des résultats impressionnants Peuvent traiter un grand nombre de caractéristiques importantes	Nécessité de mettre les données à l'échelle -Difficile d'interpréter les résultats Beaucoup de paramètres à régler
KNeighborsRegressor	neighbors	Crée un modèle basé sur les k plus proches voisins à un point donné. Où k est un argument d'entrée.						Oui	Données non linéaires, l'interprétabilité est importante, caractéristiques non importantes	Lorsque vous n'êtes pas sûr de la structure de vos données et que vous souhaitez un modèle qui s'ajuste bien. Non concerné par l'overfitting L'interprétabilité est importante	S'adapte très bien aux données de structures variées Plus facile à interpréter que les autres modèles non linéaires	Extrêmement impacté par les valeurs aberrantes et les données corrompues Il faut beaucoup plus d'échantillons que de caractéristiques pour obtenir des résultats de qualité. Difficulté à traiter un grand nombre de caractéristiques
RadiusNeighborsRegressor	neighbors	Crée un modèle basé sur tous les plus proches voisins dans un rayon r donné pour un point donné. Où r est un argument d'entrée.						Oui	Données non linéaires, l'interprétabilité est importante	Lorsque vous n'êtes pas sûr de la structure de vos données et que vous souhaitez un modèle qui s'ajuste bien Non concerné par l'overfitting L'interprétabilité est importante	S'adapte très bien aux données de structures variées Plus facile à interpréter que les autres modèles non linéaires	Extrêmement impacté par les valeurs aberrantes et les données corrompues Il faut beaucoup plus d'échantillons que de caractéristiques pour obtenir des résultats de qualité. Difficulté à traiter un grand nombre de caractéristiques

5. Modèles - sklearn

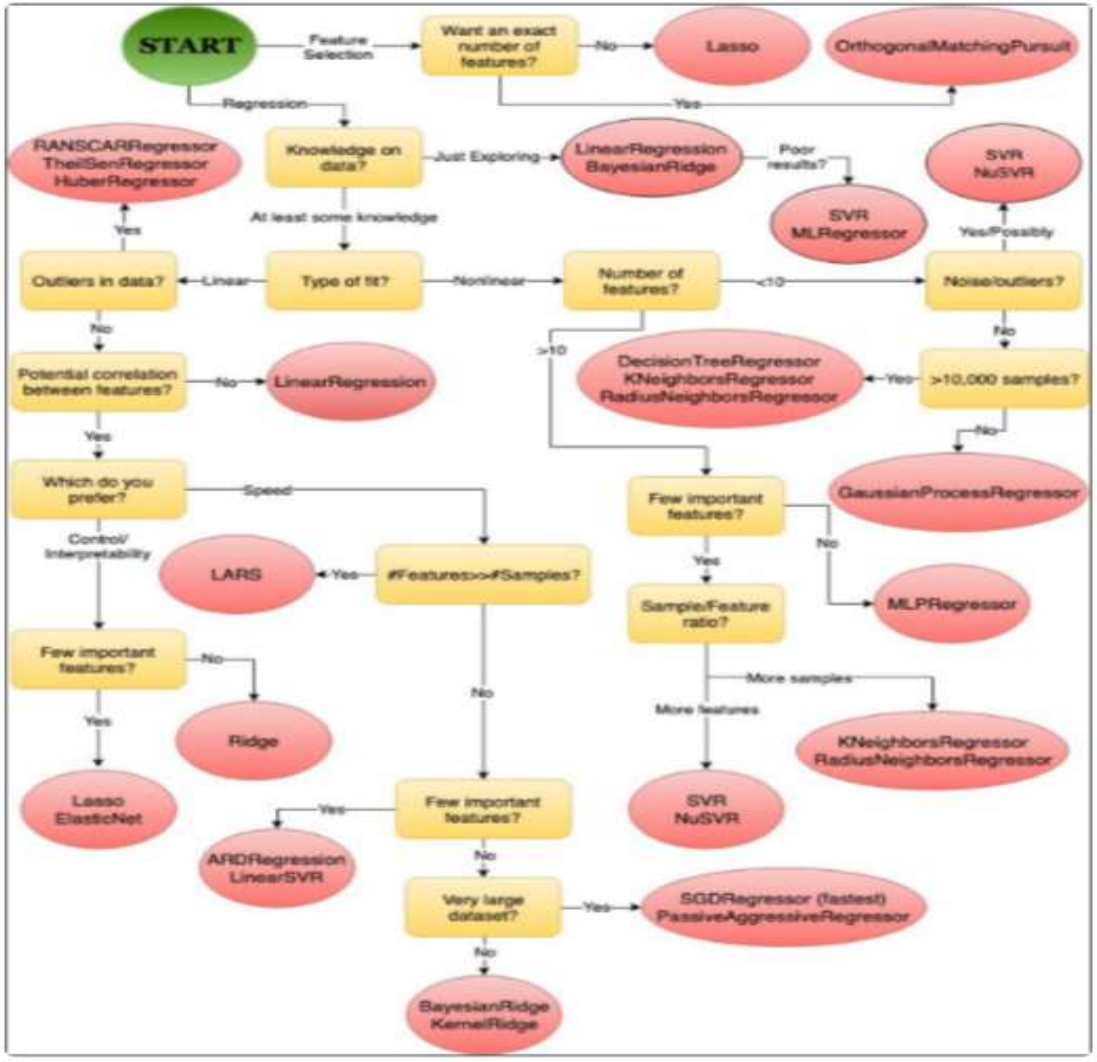
Modèle	Package	Description	Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularization	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
SVR	svm	Une implémentation de l'algorithme de la machine à vecteurs de support pour effectuer une régression. Le type de noyau peut être changé entre 'linéaire', 'poly', 'rbf', 'sigmoïde', ou une version personnalisée pour changer la façon dont le modèle apprend.			Oui			Oui	Données non linéaires, grand nombre de caractéristiques sans importance	Lorsque vous avez un jeu de données avec un grand nombre d'éléments Vous voulez que le modèle apprenne à partir d'un sous-ensemble de données qu'il juge le plus important. Vous n'êtes pas concerné par l'interprétabilité.	Efficace avec de nombreuses fonctionnalités Fournit de la flexibilité pour tester différents modèles avec des options d'algorithme de noyau	Très difficile d'interpréter les résultats Calculs coûteux pour les grands ensembles de données. Problèmes liés aux échelles variables des données d'entrée (peuvent être évités en mettant les données à l'échelle)
NuSVR	svm	Similaire à SVR avec les mêmes options de noyau. La différence réside dans la manière dont le modèle est implémenté.			Oui			Oui	Données non linéaires, grand nombre de caractéristiques non importantes.	Lorsque vous avez un jeu de données avec un grand nombre d'éléments Vous voulez que le modèle apprenne à partir d'un sous-ensemble de données qu'il juge le plus important. Vous n'êtes pas concerné par l'interprétabilité.	Efficace avec de nombreuses fonctionnalités Fournit de la flexibilité pour tester différents modèles avec des options d'algorithme de noyau	Très difficile d'interpréter les résultats Calculs coûteux pour les grands ensembles de données. Problèmes liés aux échelles variables des données d'entrée (peuvent être évités en mettant les données à l'échelle). Moins utilisé en pratique que le SVR (donc moins d'aide de la part de la communauté)
LinearSVR	svm	Une implémentation de plus bas niveau du SVR qui ne considère que les solutions linéaires de la machine à vecteurs de support. Mais ce faisant, elle est beaucoup plus efficace en termes de calcul pour les grands ensembles de données.			Oui				SVR, mais avec un très grand nombre d'échantillons et/ou de caractéristiques.	Vous souhaitez implémenter le SVR sur un très grand jeu de données (>100k).	Efficace avec beaucoup de caractéristiques S'adapte mieux aux grands ensembles de données que SVR et NuSVR	Difficile d'interpréter les résultats Problèmes liés aux échelles variables des données d'entrée (peuvent être évités par la mise à l'échelle des données)

5. Modèles - sklearn

Modèle	Package	Description	Overfit?	Gère outliers?	Gère gd nbre var?	adaptive learns regularizatio n	Large dataset	Non linear	Usage?	Usage+?	Avantages	Désavantage
KernalRidge	kernel_ridge	Une implémentation kernalisée de Ridge. Les performances sont très similaires à celles du SVR, mais les résultats sont moins épars.			Oui				Une autre déviation du SVR	Vous voulez tester un algorithme en plus du SVR qui optimise une fonction de perte différente ?	Efficace avec de nombreuses fonctionnalités	Utilisé moins que le SVR dans la pratique Très difficile d'interpréter les résultats Problèmes liés aux échelles variables des données d'entrée (peuvent être évités en mettant les données à l'échelle)
IsotonicRegression	isotonic	Ajuste une ligne par morceaux aux données qui n'est pas décroissante. Utilise la fonction d'erreur quadratique moyenne pour ajuster la ligne.						Oui	Le jeu de données présente des sauts importants dans les valeurs d'étiquettes.	Il existe de grands écarts entre les valeurs d'un ensemble de données qu'il serait difficile d'ajuster à une ligne continue.	Souvent utilisé pour approximer les résultats prédits par d'autres modèles moins interprétables. Peut gérer une grande variance dans les points de données Résultats interprétables	Utilisé peu dans la pratique et peu utile pour l'application est limité à l'augmentation (ne peut pas traiter des données qui ne sont pas en augmentation)

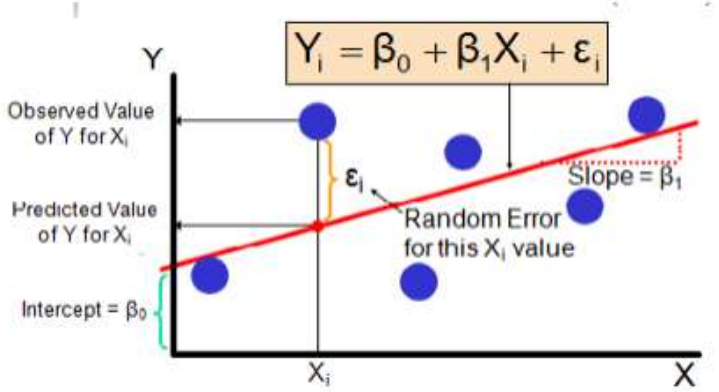
Source : https://towardsdatascience.com/choosing-a-scikit-learn-linear-regression-algorithm-dd96b48105f5

5. Modèles - sklearn



5. Modèles – Régression linéaire

La **régression linéaire** est un modèle linéaire qui suppose une relation linéaire entre les variables d'entrée (variables indépendantes "x") et la variable de sortie (variable dépendante "y"), de sorte que "y" peut être calculé à partir d'une combinaison linéaire des variables d'entrée (x).
Pour une variable à entrée unique, la méthode est appelée régression linéaire simple, tandis que pour des variables à entrées multiples, elle est appelée régression linéaire multiple.

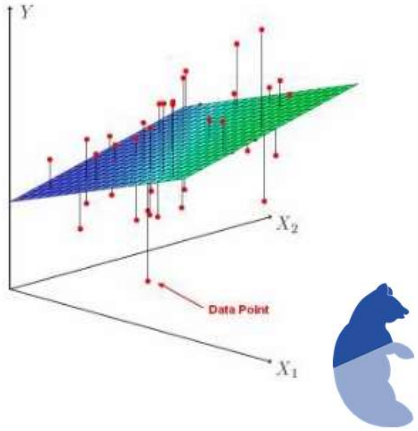


Les coefficients sont trouvés **en maximisant la vraisemblance** (la probabilité d'observer nos n observations étant donné β, si erreurs normalement distribuée et centrées en 0 et les variables iid (indépendantes et identiquement distribuées) et non corrélées) ⇔ **minimiser la somme des carrés des erreurs**

$$\min_{\beta \in \mathbb{R}^2} \left(\sum_{i=1}^{n} (y^i - f_\beta(x^i))^2 \right)$$

Avantage :

- Interprétable. Pas de paramètres à régler.
- SI plus de features que de données de tests. Gros jeux de données ou grandes dimensions (features).



Inconvénient :

- Si les variables sont CORRELEES : coefficient très variant et solution non unique, dur à interpréter ➔ et problème de sur apprentissage ➔ régulariser.
- Sujets sur-apprentissage (pas régularisation).
- Ne peut pas contrôler la complexité du modèle (pas de param).

5. Modèles – Régularisation

Régularisation : Pour limiter le sur-apprentissage, on peut utiliser une technique, la **régularisation**, qui consiste à contrôler simultanément **l'erreur du modèle sur le jeu d'entraînement** et **la complexité du modèle. ➔ minimiser une fonction objective** qui est la somme d'un terme d'erreur et d'un terme mesurant la complexité du modèle.

$$\arg \min_{\beta \in \mathbb{R}^{p+1}} (y - X\beta)^\top (y - X\beta) + \lambda \, \text{Regularisateur}(\beta)$$

Le **régularisateur**, qui mesure la complexité du modèle, est une fonction des poids β du modèle.
λ = hyperparamètre, le **coefficient de régularisation,** qui contrôle l'importance relative du terme d'erreur et du terme de régularisation.

- Plus **λ** est grand, plus le terme de régularisation est important.
- Plus il est petit, est plus l'erreur est importante
- s'il est suffisamment faible (et en particulier s'il est égal à zéro), on retrouvera la solution de la régression linéaire non-régularisée.

Quelle valeur donner à cet hyperparamètre ? En général, c'est une question que l'on réglera en utilisant une **validation croisée** .

5. Modèles – Ridge

Régularisation de Tykhonov : La régularisation de Tykhonov est un cas particulier de régularisation, dans lequel on utilise pour régulariser la régression linéaire **le carré de la norme euclidienne l2** du vecteur de poids β.

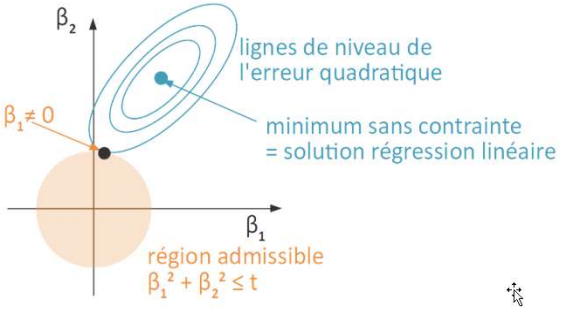
Il faut optimiser : $\arg\min_{\beta \in \mathbb{R}^{p+1}} ||y - X\beta||_2^2 + \lambda ||\beta||_2^2$ ou minimiser la somme des carrés des erreurs $||\beta||_2^2 = \sum_{j=0}^{p} \beta_j^2$

Erreur **Complexité**

Le modèle linéaire que l'on apprend en résolvant cette équation est appelé **régression ridge.**
Elle réduit les poids des coefficients de pondération (les plus petits possibles) associés à chacune des variables.

Comment la solution de la régression ridge évolue-t-elle en fonction du coefficient de régularisation **λ** ?

- Si **λ** est très grand, alors la régularisation prend le dessus, le terme d'erreur n'importe plus et la solution est **β=0**, on pénalise la compexité du modèle et plus on a des **β** petits..
- À l'inverse, si **λ** est très faible, le terme de régularisation n'est plus utilisé,
- et on retrouve la solution de la régression linéaire non régularisée.



Solution de la régression ridge.

Paramètre : alpha à régler

Avantage :

- La régression ridge admet *toujours* une solution *analytique unique*.
- La régression ridge permet **d'éviter le surapprentissage** en **restreignant l'amplitude des poids.**
- La régression ridge a un effet de **sélection groupée** : les variables corrélées ont le même coefficient.

Inconvénient :

- Standardiser les variables

5. Modèles – Lasso

Lasso (*Least Absolute Shrinkage and Selection Operator)*: modèle **parcimonieux** de **sélection de variables** et de **réduction de dimension supervisée** : les variables qui ne sont pas nécessaires à la prédiction de l'étiquette sont éliminées ➔ *annuler* certains coefficients.
Les variables qui auront un coefficient égal à zéro ne feront plus partie du **modèle,** qui en sera simplifié d'autant.

Régularisation = norme **ℓ1 :** $||\beta||_1 = \sum_{j=1}^{p} |\beta_j|$

Il faut optimiser : $\arg\min_{\beta \in \mathbb{R}^{p+1}} ||y - X\beta||_2^2 + \lambda||\beta||_1.$ mais pas de solution explicite donc algorithme de gradient pour le résoudre.

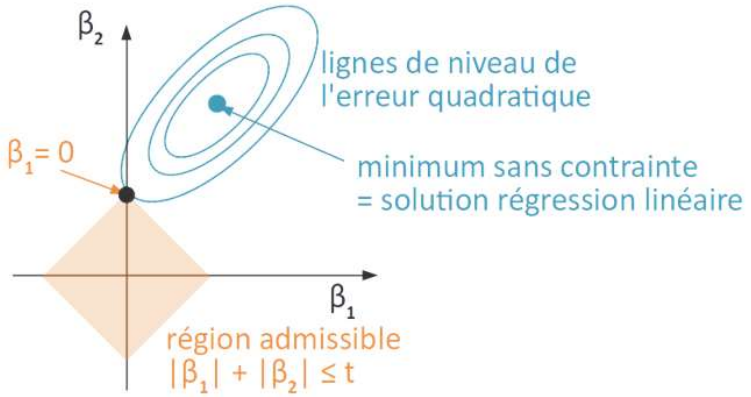
Paramètre : alpha à régler, contrôle l'impact sur la valeur des coefficients.
Basse : moins complexe et bon résultat, très basse : surapprentissage.

Avantage :

- Parcimonieux : **sélection de variables** et de **réduction de dimension supervisée** : les variables qui ne sont pas nécessaires à la prédiction de l'étiquette sont éliminées.

Inconvénient :

- Instable : Si plusieurs variables corrélées contribuent à la prédiction de l'étiquette, le lasso va avoir tendance à choisir *une seule* d'entre elles (affectant un poids de 0 aux autres), plutôt que de répartir les poids équitablement comme la régression ridge, mais aléatoire ➔ instable



Solution du lasso.

5. Modèles – Elastic Net

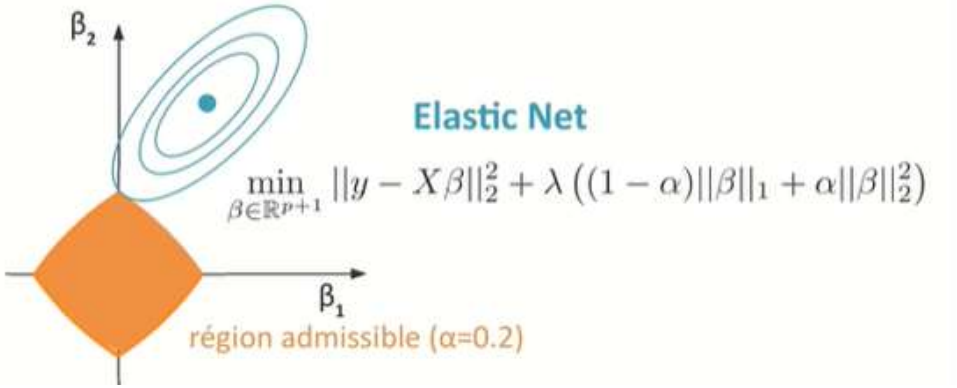
Elastic net, qui combine les deux termes de régularisation en un (norme **ℓ2** nous permet d'éviter le sur-apprentissage avec une solution unique, et l'utilisation de la norme **ℓ1** d'avoir un modèle parcimonieux mais instable).

$$\arg \min_{\beta \in \mathbb{R}^{p+1}} ||y - X\beta||_2^2 + \lambda \left((1-\alpha)||\beta||_1 + \alpha||\beta||_2^2 \right)$$

L'**Elastic net** combine les normes **ℓ1** et **ℓ2** pour obtenir une solution *moins parcimonieuse* que le lasso, mais plus stable et dans laquelle toutes les variables corrélées pertinentes pour la prédiction de l'étiquette sont sélectionnées et reçoivent un poids identique.

$0 \leq \alpha \leq 1$

Si alpha = 0 ➔ LASSO
Si alpha = 1 ➔ RIDGE



Avantage :

- Stabilité : plus stable et dans laquelle toutes les variables corrélées pertinentes pour la prédiction de l'étiquette sont sélectionnées et reçoivent un poids identique.

Inconvénient :

- 2 hyperparamètres, alpha et lambda ce qui demande plus de Ressources computationnelles.

5

Modèles Arbres de décision

5. Modèles – Arbres de décision

Un **DecisionTree** est un algorithme d'apprentissage automatique supervisé.

Ce système non paramétrique, contrairement aux modèles de régression linéaire (qui supposent la linéarité), ne fait aucune hypothèse sous-jacente sur la distribution des erreurs ou des données.

Il s'agit d'une structure de type organigramme, composée de plusieurs questions (nœud) et qui, en fonction des réponses (branche) données, conduira à une étiquette de classe ou à une valeur (feuille) lorsqu'elle sera appliquée à une observation quelconque.

Les arbres de décision permettent de classer des objets en effectuant des décisions successives sur la base de leurs variables.

L'algorithme de base pour la construction d'arbres de décision utilise une recherche descendante top-down, greedy search dans l'espace des branches possibles, sans retour en arrière.

Les noeuds de l'arbre représentent ces décisions alors que les feuilles représentent les valeurs de la variable cible (à prédire).

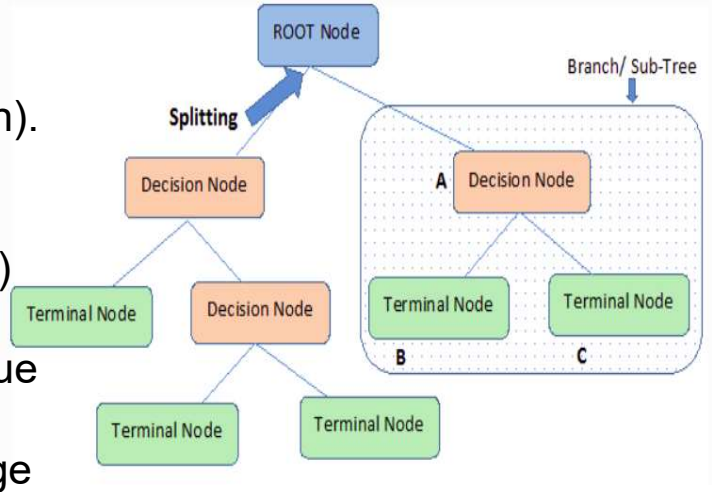
Plusieurs algorithmes existent permettant de générer des arbres de décision (*ID3*, *C4.5*, *CART*, etc.). Chaque algorithme utilise un critère pour choisir la variable à utiliser sur un noeud (gain d'information, entropie, écart-type, MSE (régression) etc.)

5. Modèles – Arbres de décision

Algorithme :
Conformément à l'analogie avec les arbres, les arbres de décision mettent en œuvre un processus de décision séquentiel.
En partant du nœud racine, une caractéristique est évaluée et l'un des deux nœuds (branches) est sélectionné.
Chaque nœud de l'arbre est fondamentalement une règle de décision.
Cette procédure est répétée jusqu'à ce qu'une feuille finale soit atteinte, qui représente normalement la cible.
Les arbres de décision sont également des modèles intéressants si l'on se soucie de l'interprétabilité.

Avantage :
- Interprétables. Pas besoin de norma/standardisation.
- Représentation graphique. Feature importances (importance dans la décison).
- Les arbres peuvent facilement traiter des prédicteurs qualitatifs sans avoir à créer des variables fictives.
- Régression : pas d'extrapolation possible (prédiction en dehors jeux de train)

Inconvénient :
- les arbres n'ont généralement pas le même niveau de précision prédictive que certaines des autres approches de régression et de classification.
- Surapprentisage si profond et feuille pure (100% apprentissage) : pré élagage : stopper en amont la création de l'arbre (max_depth, max_leaf_nodes, min_samples_leaf) ou post-élagage : supprimer ou regrouper les nœuds avec peu d'information). Pas très bon en généralisation



5. Modèles – Arbres de décision

Hyparamètres :

min_samples_split (nombre minimum d'échantillons qu'un nœud doit avoir avant d'être divisé),

min_sample_leaf (nombre minimum d'échantillons qu'une feuille doit avoir)

min_weight_fraction_leaf (identique à min_sample_leaf mais exprimé comme une fraction du nombre total d'instances pondérées)

max_leaf_nodes (nombre maximum de noeuds de feuilles)

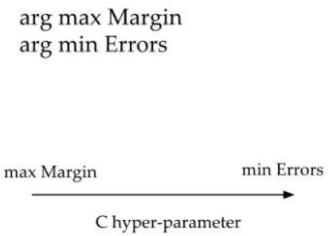
max_features (nombre maximum de caractéristiques qui sont évaluées pour le fractionnement dans chaque nœud). L'augmentation du min ou la diminution du max régularise le modèle.

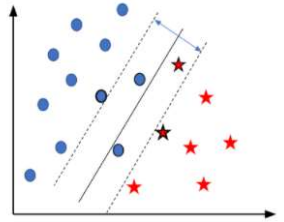
5

Modèles SVM

5. Modèles – SVM

Les **SVM** machines à vecteurs de support sont le plus souvent utilisées sur des données avec beaucoup de variables.
L'idée des SVMs est de chercher un hyperplan (un séparateur linéaire) qui sépare au mieux les objets de chaque catégorie. En effet, il peut y avoir une infinité de séparateurs possibles comme le montre la figure ci-dessous.
Le meilleur hyperplan selon les SVMs est celui qui maximise les marges avec les objets de chaque catégorie.
Ces objets sont d'ailleurs appelés "vecteurs de support" car ils supportent les hyper-plans parallèles.
Par contre, il est fréquent qu'il n'y ait pas d'hyperplan capable de séparer parfaitement les données.
Dans ce cas, on peut permettre qu'il y ait un certain nombre d'erreurs.
L'optimisation de l'hyper-paramètre C permet justement de fixer un compromis entre la maximisation des marges et la minimisation des erreurs. Prédiction = mesure de la distance du nouveau point aux vecteurs de support.





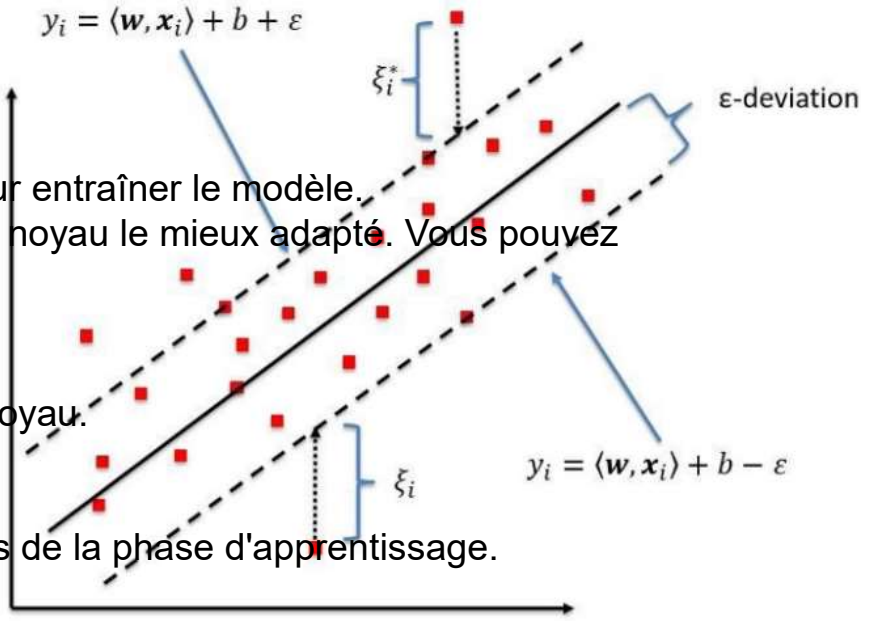
Astuce du noyau :entraîner un modèle dans un espace de grande dimension en calculant directement la distance (produit scalaire) des points de données pour la représentation étendue des caractéristiques sans jamais calculer réellement cette extension. Ex : noyau polynomial, noyau rbf radial gaussien.

Paramètres :

- Gamma :contrôle la largeur du noyau gaussien, détermine la portée de l'influence de certains échantillons (petite valeur : grande portée, grande : petite portée.
- C : paramètre de régularisation (limite l'importance de chaque point).

5. Modèles – SVR

L'objectif du vecteur de support est de trouver un hyperplan dans un espace à N dimensions qui peut classer les points de données.
Les points de données situés sur la marge et les plus proches de l'hyperplan sont appelés vecteurs de support.
Maintenant, lorsque la plupart des données se trouvent dans la meilleure marge vers chaque côté de l'hyperplan, alors le SVR ou la régression vectorielle de soutien peut être utilisé pour identifier et prédire les données dépendantes très clairement.
Sa formule mathématique pour les lignes marginales vers chaque côté de l'hyperplan est représentée par :
Yi = (w , xi) + b + déviation
Yi = (w, xi) + b- déviation



Étapes à suivre pour construire un modèle de régression de support :
1) Trouvez vos ensembles de données X et Y, indépendants et dépendants, pour entraîner le modèle.
2) Observez les données en un coup d'œil et essayez d'ajuster le paramètre de noyau le mieux adapté. Vous pouvez également essayer de tracer les points de données et de voir la corrélation.
Il peut être linéaire, gaussien ou polynomial, selon la complexité.
Le noyau le plus couramment utilisé est gaussien.
*Noyau polynomial : K(x,y) =(x .y +1)d, où d>0 est une constante qui définit le noyau.
Ordre.
*Noyau RBF gaussien : K(x,y) =exp(-| x-y|2/2σ2), où σ>0 est un paramètre qui
Définit la largeur du noyau. Les paramètres associés d et σ sont déterminés lors de la phase d'apprentissage.
3) Tracer des graphiques
4) Prédire pour toute valeur indépendante.

5. Modèles – SVR

Avantages :

- Puissants et efficaces sur une large variété de jeux de données (peu ou grandes dimensions des features).

Inconvénients :

- Pré-traitement des données : Standardiser avec même échelle : MinMaxScaler.
- Pas pour les grandes quantités de données (temps de consommation mémoire trop grand).
- Ajustement précis des paramètres.
- Peu interprétable.

Modèles KNeighbors

5. Modèles – KNeighbors

K- Nearest Neighbors est un algorithme d'apprentissage automatique supervisé car la variable cible est connue.
Non paramétrique car il ne fait pas d'hypothèse sur le modèle de distribution des données sous-jacentes.
Algorithme paresseux car KNN n'a pas d'étape de formation.
Tous les points de données ne seront utilisés qu'au moment de la prédiction.
Sans étape de formation, l'étape de prédiction est coûteuse.
Utilise la similarité des caractéristiques pour prédire le groupe dans lequel le nouveau point sera classé.

Algorithme :
Étape 1 : Choisissez une valeur pour K. K doit être un nombre impair.
Étape 2 : Trouver la distance du nouveau point par rapport à chacune des données d'apprentissage.
Étape 3 : Trouvez les K voisins les plus proches du nouveau point de données.
Étape 4 : Pour la classification, comptez le nombre de points de données dans chaque catégorie parmi les k voisins. Le nouveau point de données appartiendra à la classe qui a le plus de voisins.
Pour la régression, la valeur du nouveau point de données sera la moyenne des k voisins.

La distance peut être calculée en utilisant

- La distance euclidienne $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
- La distance de Manhattan $|x_1 - x_2| + |y_1 - y_2|$
- La distance de Hamming
- La distance de Minkowski $(|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$

5. Modèles – KNeighbors

Avantages :

- Algorithme simple et donc facile à interpréter la prédiction.
- Non paramétrique, donc ne fait aucune hypothèse sur le modèle de données sous-jacentes
- Utilisé à la fois pour la classification et la régression
- L'étape de formation est beaucoup plus rapide pour les plus proches voisins que pour les autres algorithmes d'apprentissage automatique, mais prédictions parfois longues si bcp features/données.

Inconvénients :

- KNN est coûteux en calcul car il recherche les plus proches voisins pour le nouveau point à l'étape de prédiction.
- Besoin élevé en mémoire car KNN doit stocker tous les points de données (pas plus de 100 features).
- L'étape de prédiction est très coûteuse
- Sensible aux valeurs aberrantes, la précision est affectée par le bruit ou les données non pertinentes.
- Sensible aux datasets sparses (jeux de données parsemés avec beaucoup de données nulles).
- Pour les petits jeux de données.

Paramètres importants :

- K : nombre de voisins (augmenter k lisse les prédictions, mais moins bons résultats apprentissage).
- Distance : mesurée entre les points, euclidienne, manhatan

5

Modèles Ensemblistes

5. Modèles – Ensemblistes

apprentissage automatique dans lequel plusieurs modèles (souvent appelés "apprenants faibles") sont formés pour résoudre le même problème et combinés pour obtenir de meilleurs résultats (plus précis et/ou robuste).

Attention : combinaison des modèles apprenants faibles à faibles bias et fortes variances ➔ réduire la variance (et vis et versa)

Méthodes **parallèles :**

entraîner plusieurs modèles apprenants faibles de manière indépendante (en parallèle) pour ensuite les regrouper afin de prendre une décision

Méthodes **séquentielles :**

entraîner plusieurs modèles apprenants faibles de manière séquentielle à la suite, premier modèle induit erreurs, on donne plus de poids aux données induisant erreur et entraîne deuxième modèle, puis nouveau modèle entraîné avec le jeu de données pondérés en augmentant le poids des observations induisant le plus d'erreur… Le dernier modèle est le plus performant.

Algorithme de combinaison ?

BAGGING:

Entraîne plusieurs modèles apprenants faibles homogènes, indépendamment les uns des autres en **parallèle** et les combine en suivant un processus déterministe de calcul de moyenne (moins de variance).

BOOSTING:

Entraîne des modèles apprenants faibles homogènes **séquentiellement** de manière très adaptative (un modèle de base dépend des précédents) et les combine en suivant une stratégie déterministe (modèles forts moins biaisés).

STACKING:

Entraîne des modèles apprenants faibles hétérogènes en **parallèle** et les combine en formant un méta-modèle pour produire une prédiction basée sur les prédictions des différents modèles faibles (modèles forts moins biaisés).

5

Modèles Ensemblistes Bagging

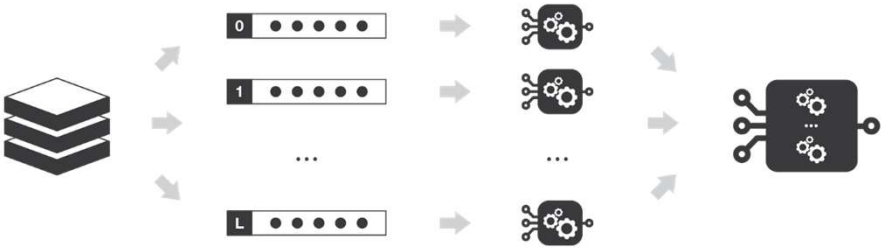
5. Modèles – Ensemblistes - Bagging

BAGGING (*bootstrap aggregation)* :

- Basée sur bootstrap (nouveaux datasets de taille identique à l'original avec remise, représentatifs et indépendants et identiquement distribués),
- entraînement de plusieurs modèles apprenants faibles presque indépendants en parallèle,
- moyenne de leurs prédictions afin d'obtenir un modèle avec une variance plus faible.

Il permet de réduire la variance des estimateurs individuels et offre une prédiction plus performante et plus stable, dépendante du nombre d'apprenants faibles utilisés.

Avantage : parallèlisation.



$\{z_1^1, z_2^1, ..., z_B^1\}, \{z_1^2, z_2^2, ..., z_B^2\}, ..., \{z_1^L, z_2^L, ..., z_B^L\}$

$z_b^l \equiv b$-th observation of the l-th bootstrap sample

$w_1(.), w_2(.), ..., w_L(.)$

$$s_L(.) = \frac{1}{L}\sum_{l=1}^{L} w_l(.)$$ (simple average, for regression problem)

$$s_L(.) = \arg\max_k [card(l|w_l(.) = k)]$$ (simple majority vote, for classification problem)

Algorithme :

- Des sous-ensembles aléatoires sont créés à partir de l'ensemble de données original (Bootstrapping). Le sous-ensemble de l'ensemble de données comprend toutes les caractéristiques.
- Un estimateur de base spécifié par l'utilisateur est ajusté sur chacun de ces petits ensembles.
- Les prédictions de chaque modèle sont combinées pour obtenir le résultat final.

5. Modèles – Ensemblistes - Bagging

Hyperparamètres :

base_estimator : Il définit l'estimateur de base à adapter sur des sous-ensembles aléatoires de l'ensemble de données.
Si rien n'est spécifié, l'estimateur de base est un arbre de décision.

n_estimators : C'est le nombre d'estimateurs de base à créer. Le nombre d'estimateurs doit être soigneusement ajusté car un grand nombre prendrait beaucoup de temps à exécuter, tandis qu'un très petit nombre pourrait ne pas fournir les meilleurs résultats.

max_samples : Ce paramètre contrôle la taille des sous-ensembles. Il s'agit du nombre maximum d'échantillons pour entraîner chaque estimateur de base.

max_features : Ce paramètre contrôle le nombre de caractéristiques à tirer de l'ensemble des données. Il définit le nombre maximum de caractéristiques requises pour entraîner chaque estimateur de base.

n_jobs : Le nombre de tâches à exécuter en parallèle. Définissez cette valeur comme étant égale au nombre de cœurs dans votre système. Si la valeur est -1, le nombre de tâches est égal au nombre de cœurs.

random_state : Indique la méthode de répartition aléatoire. Lorsque la valeur de random state est la même pour deux modèles, la sélection aléatoire est la même pour les deux modèles. Ce paramètre est utile lorsque vous souhaitez comparer différents modèles.

Modèles Random Forest

5. Modèles – Bagging - RandomForest

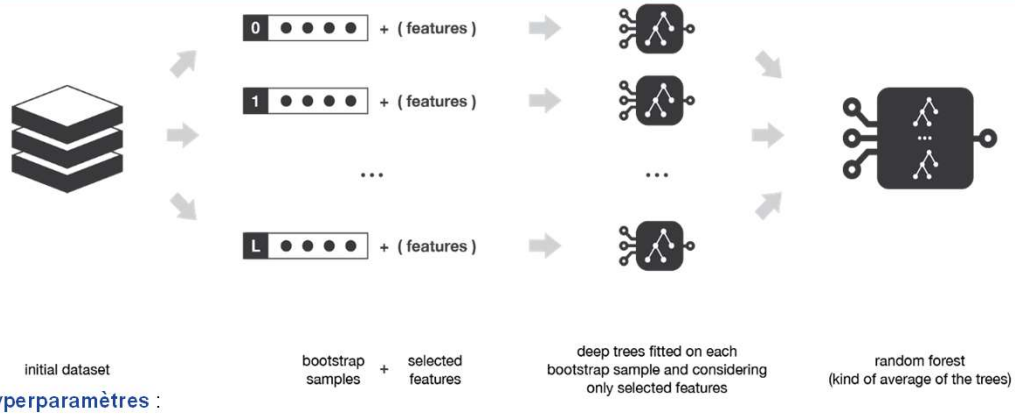
RandomForest :
Méthode de bagging où les arbres profonds (low bias, high variance), ajustés sur des échantillons bootstrap de données et des sous-ensembles aléatoires de features, sont combinés (moyenne des prédictions pour tous les arbres pour la régression) pour produire une sortie avec une variance plus faible.

1. *greedy, top-down recursive partitioning algorithm* : faire grandir l'arbre (feature à utiliser et point de séparation pour minimiser l'erreur quadratique)
2. Elaguer l'arbre : réduire l'over-fitting en prenant en compte le coût de complexité du modèle.

Avantages *:* amélioration du bagging puisqu'elles créent des arbres plus différents structurellement en échantillonnant aléatoirement seulement un sous-ensemble des features disponibles, ce qui permet de réduire la corrélation entre les arbres et donc d'obtenir un meilleur modèle final. Gros datasets. Interprétable. Estimation de l'importance des features. Rapide. Pas d'over-fitting. Plus robuste aux valeurs manquantes. Hyperparamètres de base efficaces. Parallèlisable.

Désavantages : taille mémoire pour stockage gros jeux. Un nombre élevé d'arbres peut rendre le processus de calcul beaucoup plus lent et inefficace pour les prédictions en temps réel. Datasets éparses. Pas données textuelles (grandes dimensions).

5. Modèles – Bagging - RandomForest



Algorithme :

- Des sous-ensembles aléatoires sont créés à partir de l'ensemble de données original (bootstrapping).
- À chaque nœud de l'arbre de décision, seul un ensemble aléatoire de caractéristiques est pris en compte pour décider de la meilleure répartition (algo de calcul la combinaison caractéristique/séparation localement optimale).
- Un modèle d'arbre de décision est ajusté sur chacun des sous-ensembles.
- La prédiction finale est calculée en faisant la moyenne des prédictions de tous les arbres de décision.

Hyperparamètres :

Augmenter le pouvoir prédictif :

n_estimators : Nombre d'arbres que l'algorithme fait croître avant de faire la prédiction. Un nombre plus élevé d'arbres devrait augmenter les performances et rendre la prédiction plus stable, avec l'inconvénient de ralentir le calcul (élevée mieux)

max_features : correspond au maximum de caractéristiques que Random Forest est autorisé à essayer dans un arbre individuel, lors de la recherche de la meilleure division ; (=sqrt(n_feature) classification, n_features : régression)

min_sample_leaf : nombre minimum requis de feuilles pour effectuer la division sur un noeud interne.

max_depth : La forêt aléatoire comporte plusieurs arbres de décision. Ce paramètre définit la profondeur maximale des arbres.

criterion: Il définit la fonction qui doit être utilisée pour le fractionnement. La fonction mesure la qualité d'un fractionnement pour chaque caractéristique et choisit le meilleur fractionnement.

min_samples_leaf : Ceci définit le nombre minimum d'échantillons requis pour être à un nœud feuille. Une taille de feuille plus petite rend le modèle plus enclin à capturer le bruit dans les données de train.

max_leaf_nodes : Ce paramètre spécifie le nombre maximal de nœuds de feuille pour chaque arbre. L'arbre cesse de se diviser lorsque le nombre de nœuds de feuilles devient égal au nombre maximal de nœuds de feuilles (peu aider à meilleur résultat.

Augmentation de la vitesse de traitement :

n_jobs : indique au système le nombre de processeurs qu'il est autorisé à utiliser. La valeur '-1' signifie qu'il n'y a pas de limite ;

random_state : rend la sortie du modèle reproductible. Il produira toujours les mêmes résultats si vous lui donnez une valeur fixe ainsi que les mêmes paramètres et données d'entraînement.

oob_score : méthode de validation croisée de Random Forest. On appelle cela les échantillons hors-sac. Elle est très similaire à la méthode de validation croisée leave-one-out mais sans charge de calcul.

Modèles ExtraTrees

5. Modèles – Bagging - ExtraTrees

L'algorithme **Extremely Randomized Trees**, ou **Extra Trees** en abrégé, est un algorithme d'apprentissage automatique d'ensemble d'arbres de décision, lié à d'autres ensembles d'algorithmes d'arbres de décision tels que l'agrégation bootstrap (bagging) et la forêt aléatoire.
L'algorithme Extra Trees fonctionne en créant un grand nombre d'arbres de décision non élagués à partir de l'ensemble de données d'apprentissage (pas bootstrap, échantillonne sans remplacement, réduit biais).
Les prédictions sont faites en faisant la moyenne des prédictions des arbres de décision dans le cas de la régression.

L'algorithme Extra-Trees construit un ensemble d'arbres de décision ou de régression non élagués selon la procédure descendante classique. Ses deux principales différences avec les autres méthodes d'ensemble basées sur les arbres sont qu'il divise les nœuds en choisissant les caractéristiques des points de coupe de manière totalement aléatoire (pas un algorithme gourmand pour sélectionner un point de division optimal, réduit la variance) et qu'il utilise l'échantillon d'apprentissage entier (plutôt qu'une réplique bootstrap) pour faire croître les arbres.

Avantages :

- sélection aléatoire des points de fractionnement rend les arbres de décision de l'ensemble moins corrélés (mais augmente la variance donc augmenter le nombre d'arbres).
- Coût de calcul plus faible que RF ➔ plus rapide.

Inconvénients :

- sélection aléatoire ➔ jamais le même résultat lorsque l'algorithme est exécuté.
- Évaluation : faire une validation croisée, ajustement : augmenter les arbres jusqu'à variance stabilisée

5. Modèles – Bagging - ExtraTrees

Hyperparamètres :

n_estimators : le nombre d'arbres de décision dans l'ensemble.

max_features : le nombre de caractéristiques d'entrée à sélectionner aléatoirement et à prendre en compte pour chaque point de séparation. Les bonnes valeurs empiriques par défaut sont max_features=None (toujours considérer toutes les caractéristiques au lieu d'un sous-ensemble aléatoire) pour les problèmes de régression, et max_features="sqrt" (utiliser un sous-ensemble aléatoire de taille sqrt(n_features)) pour les tâches de classification (où n_features est le nombre de caractéristiques dans les données).

min_samples_split :le nombre minimum d'échantillons requis dans un nœud pour créer un nouveau point de séparation.

min_samples_leaf : Le nombre minimum d'échantillons requis pour être à un noeud feuille. Un point de séparation à n'importe quelle profondeur ne sera considéré que s'il laisse au moins min_samples_leaf échantillons d'entraînement dans chacune des branches gauche et droite. Cela peut avoir pour effet de lisser le modèle, en particulier dans la régression.

max_depth : profondeur , de bons résultats sont souvent obtenus en définissant max_depth=None (tout déployé)

bootstrap : un paramètre optionnel qui permet aux utilisateurs d'utiliser des répliques bootstrap, mais par défaut, il utilise l'échantillon d'entrée entier. Cela peut augmenter la variance car le bootstrap la rend plus diversifiée.

n_jobs : construction parallèle des arbres et calcul parallèle des prédictions

`min_impurity_decrease : Un nœud sera divisé si cette division induit une diminution de l'impureté supérieure ou égale à cette valeur.`

`random_state` : valeur aléatoire

`warm_start` : Lorsqu'elle est définie sur True, la solution de l'appel précédent à l'ajustement est réutilisée et d'autres estimateurs sont ajoutés à l'ensemble, sinon, une nouvelle forêt est ajustée.

Verbose : Contrôle la verbosité lors de l'ajustement et de la prédiction.

Criterion : fonction permettant de mesurer la qualité d'un fractionnement. Les critères pris en charge sont "mse" pour l'erreur quadratique moyenne, qui est égale à la réduction de la variance comme critère de sélection des caractéristiques, et "mae" pour l'erreur absolue moyenne.

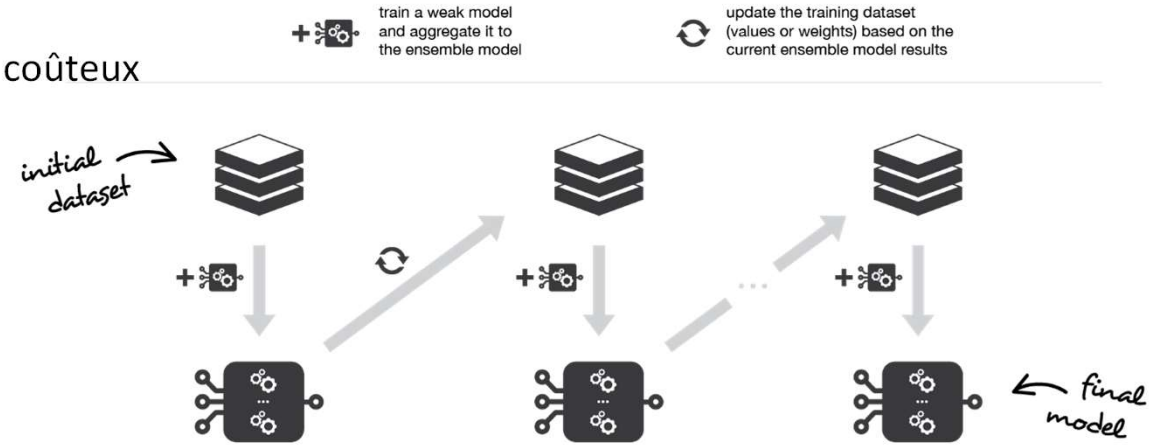
Modèles Ensemblistes Boosting

5. Modèles – Ensemblistes - Boosting

BOOSTING :
Entraînement de modèles apprenants faibles (peu profonds high bias, low variance) qui sont agrégés séquentiellement pour obtenir un apprenant fort plus performant.
Chaque modèle de la séquence est ajusté en donnant plus d'importance aux observations du jeu de données qui ont été mal traitées par les modèles précédents de la séquence.
Le modèle final est un apprenant fort avec un biais plus faible (et parfois aussi réduction de la variance).

Réglage :
- informations prises pour ajuster séquentiellement?
- comment agréger le modèle courant aux précédents ?

Inconvénient : pas de parallélisation, ajustement séquentiel coûteux



5

Modèles AdaBoost

5. Modèles – Boosting - AdaBoost

AdaBoost :

Le boosting adaptatif met à jour les poids attachés à chacune des observations de l'ensemble de données d'apprentissage.

Le modèle d'ensemble est défini comme une somme pondérée de L apprenants faibles (stumps c'est à dire des arbres qui ne font qu'une seule séparation, avec un seul noeud) construits séquentiellement et une agrégation en une simple combinaison linéaire pondérée par des coefficients exprimant la performance de chaque apprenant.

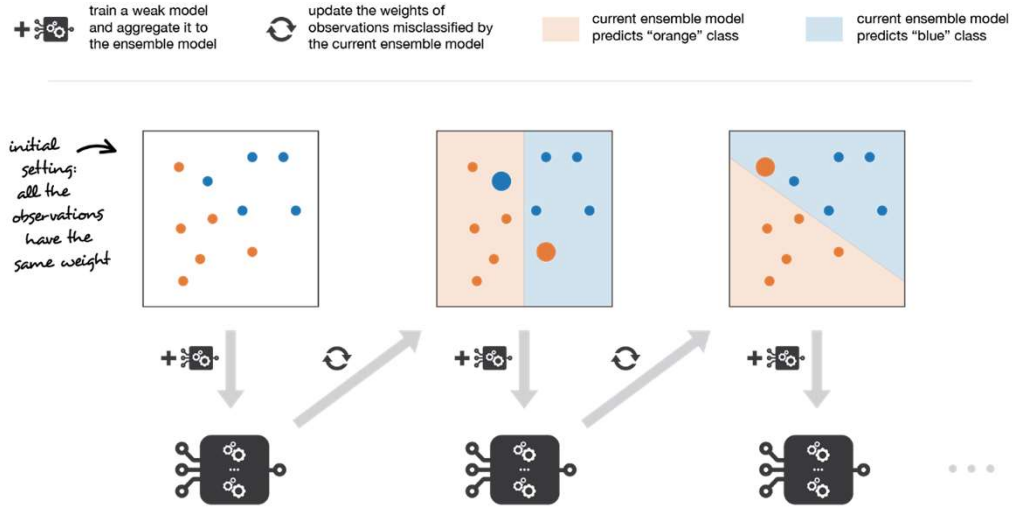
Optimisation globale difficile ➔ optimisation itérative locale en ajoutant les apprenants faibles un par un, en recherchant à chaque itération la meilleure paire possible (coefficient, apprenant faible) à ajouter au modèle d'ensemble actuel

$$(c_l, w_l(.)) = \arg\min_{c,w(.)} E(s_{l-1}(.) + c \times w(.)) = \arg\min_{c,w(.)} \sum_{n=1}^{N} e(y_n, s_{l-1}(x_n) + c \times w(x_n))$$

où E(.) est l'erreur d'ajustement du modèle donné et e(.,.) est la fonction de perte/erreur.

L itération :

- ajuster le meilleur modèle faible possible avec les poids des observations courantes
- calculer la valeur du coefficient de mise à jour qui est une sorte de métrique d'évaluation scalaire de l'apprenant faible qui indique dans quelle mesure cet apprenant faible doit être pris en compte dans le modèle d'ensemble
- mettre à jour l'apprenant fort en ajoutant le nouvel apprenant faible multiplié par son coefficient de mise à jour
- calculer les nouveaux poids des observations qui expriment les observations sur lesquelles nous souhaitons nous concentrer à la prochaine itération (les poids des observations mal prédites par le modèle agrégé augmentent et les poids des observations correctement prédites diminuent).



5. Modèles – Boosting - AdaBoost

Algorithme :

- Au départ, toutes les observations ont le même poids.
- Un modèle est construit sur un sous-ensemble de données.
- En utilisant ce modèle, des prédictions sont faites sur l'ensemble des données.
- Les erreurs sont calculées en comparant les prédictions et les valeurs réelles.
- Lors de la création du modèle suivant, des poids plus élevés sont attribués aux points de données qui ont été prédits de manière incorrecte.
- Les pondérations peuvent être déterminées en utilisant la valeur de l'erreur. Par exemple, plus l'erreur est élevée, plus le poids attribué à l'observation est important.

Ce processus est répété jusqu'à ce que la fonction d'erreur ne change pas, ou que la limite maximale du nombre d'estimateurs soit atteinte.

Hyperparamètres :

base_estimators : spécifie le type d'estimateur de base, c'est-à-dire l'algorithme à utiliser comme apprenant de base.

n_estimators : Il définit le nombre d'estimateurs de base, la valeur par défaut est 10 mais vous pouvez l'augmenter afin d'obtenir une meilleure performance.

learning_rate : même impact que dans l'algorithme de descente de gradient. Ce paramètre contrôle la contribution des estimateurs dans la combinaison finale. Il existe un compromis entre le taux d'apprentissage et les n_estimateurs.

max_depth : profondeur maximale de l'estimateur individuel. Réglez ce paramètre pour obtenir les meilleures performances.

n_jobs : indique au système le nombre de processeurs qu'il est autorisé à utiliser. La valeur '-1' signifie qu'il n'y a pas de limite

random_state : rend la sortie du modèle reproductible. Il produira toujours les mêmes résultats si vous lui donnez une valeur fixe ainsi que les mêmes paramètres et données d'entraînement.

5

Modèles GradientBoosting

5. Modèles – Boosting – Gradient Boosting

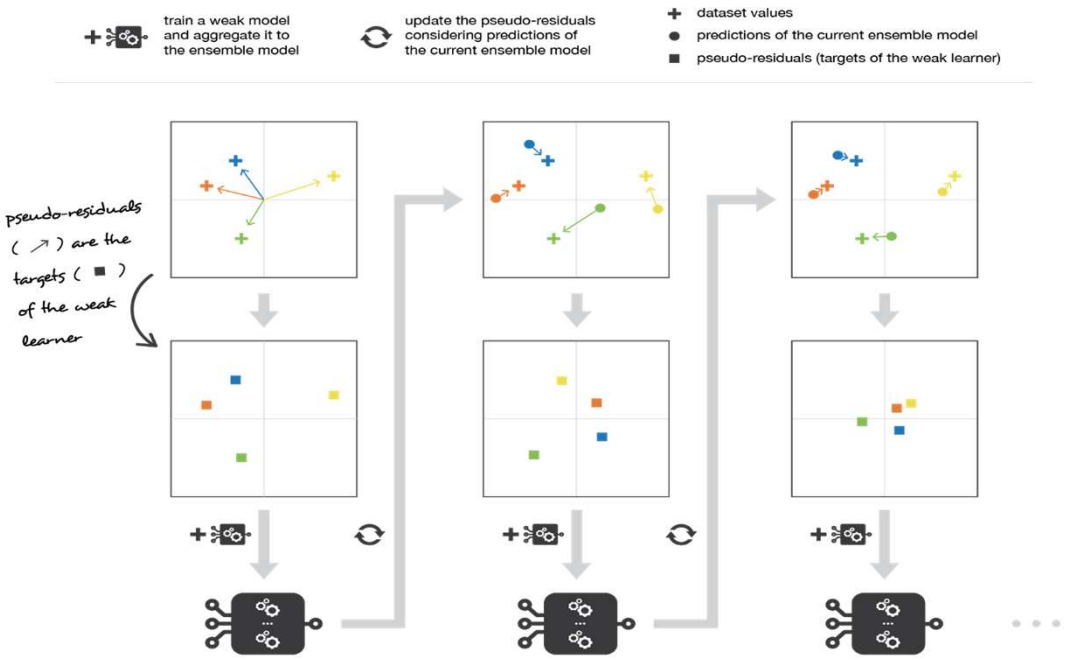
Gradient Boosting :
boosting gradient met à jour la valeur de ces observations.
Le modèle d'ensemble que nous essayons de construire est également une somme pondérée d'apprenants faibles.
L'optimisation globale est difficile ➔ optimisation séquentielle, le boosting de gradient transforme le problème en une descente de gradient : à chaque itération, nous ajustons un apprenant faible à l'opposé du gradient de l'erreur d'ajustement actuelle par rapport au modèle d'ensemble actuel.

$$s_l(.) = s_{l-1}(.) - c_l \times \nabla_{s_{l-1}} E(s_{l-1})(.)$$

où E(.) est l'erreur d'ajustement du modèle donné, c_l est un coefficient correspondant à la taille du pas et

$$-\nabla_{s_{l-1}} E(s_{l-1})(.)$$

est l'opposé du gradient de l'erreur d'ajustement par rapport au modèle d'ensemble au pas l-1.



L itérations :

- ajuster le meilleur apprenant faible possible aux pseudo-résidus (approximer l'opposé du gradient par rapport à l'apprenant fort actuel)
- calculer la valeur de la taille de pas optimale qui définit de combien nous mettons à jour le modèle d'ensemble dans la direction du nouvel apprenant faible
- mettre à jour le modèle d'ensemble en ajoutant le nouvel apprenant faible multiplié par la taille du pas (faire un pas de descente de gradient)
- calculer de nouveaux pseudo-résidus qui indiquent, pour chaque observation, dans quelle direction nous souhaitons mettre à jour les prédictions du modèle d'ensemble.

5. Modèles – Boosting – Gradient Boosting

ID	Married	Gender	Current City	Monthly Income	Age (target)	Mean Age (prediction 1)	Residual 1
1	Y	M	A	51,000	35	32	3
2	N	F	B	25,000	24	32	-8
3	Y	M	A	74,000	38	32	6
4	N	F	A	29,000	30	32	-2
5	N	F	B	37,000	33	32	1

ID	Age (target)	Mean Age (prediction 1)	Residual 1 (new target)	Prediction 2	Combine (mean+pred2)
1	35	32	3	3	35
2	24	32	-8	-5	27
3	38	32	6	3	35
4	30	32	-2	-5	27
5	33	32	1	3	35

ID	Age (target)	Mean Age (prediction 1)	Residual 1 (new target)	Prediction 2	Combine (mean+pred2)	Residual 2 (latest target)
1	35	32	3	3	35	0
2	24	32	-8	-5	27	-3
3	38	32	6	3	35	-3
4	30	32	-2	-5	27	3
5	33	32	1	3	35	-2

Algorithme :

- Un modèle est construit sur un sous-ensemble de données.
- En utilisant ce modèle, des prédictions sont faites sur l'ensemble des données.
- Les erreurs sont calculées en comparant la moyenne des prédictions et les valeurs réelles.
- Un nouveau modèle est créé en utilisant les erreurs calculées comme variable cible. Notre objectif est de trouver la meilleure répartition pour minimiser l'erreur.
- Les prédictions faites par ce nouveau modèle sont combinées avec les prédictions du modèle précédent.
- De nouvelles erreurs sont calculées en utilisant cette valeur prédite et la valeur réelle.

Ce processus est répété jusqu'à ce que la fonction d'erreur ne change pas, ou que la limite maximale du nombre d'estimateurs soit atteinte.

Hyperparamètres spécifiques à l'arbre : Ils affectent chaque arbre individuel du modèle.

min_samples_split : Définit le nombre minimum d'échantillons (ou d'observations) requis dans un nœud pour être pris en compte pour le fractionnement.
Utilisé pour contrôler le surajustement. Des valeurs plus élevées empêchent un modèle d'apprendre des relations qui pourraient être très spécifiques à l'échantillon particulier sélectionné pour un arbre.
Des valeurs trop élevées peuvent conduire à un sous-ajustement et doivent donc être réglées à l'aide de CV.

min_samples_leaf : Définit les échantillons (ou observations) minimums requis dans un nœud terminal ou une feuille.
Utilisé pour contrôler l'excès d'ajustement de manière similaire à min_samples_split.
En général, des valeurs plus faibles doivent être choisies pour les problèmes de classes déséquilibrées car les régions dans lesquelles la classe minoritaire sera majoritaire seront très petites.

min_weight_fraction_leaf : Similaire à min_samples_leaf mais défini comme une fraction du nombre total d'observations au lieu d'un nombre entier.
Une seule des options #2 et #3 doit être définie.

max_depth : La profondeur maximale d'un arbre. Utilisé pour contrôler l'excès d'ajustement car une profondeur plus élevée permettra au modèle d'apprendre des relations très spécifiques à un échantillon particulier.
Doit être réglé à l'aide de CV.

max_leaf_nodes : Le nombre maximum de nœuds terminaux ou de feuilles dans un arbre.
Peut être défini à la place de max_depth. Comme les arbres binaires sont créés, une profondeur de 'n' produirait un maximum de 2^n feuilles.
Si cette valeur est définie, GBM ignorera max_depth.

max_features : Le nombre de caractéristiques à prendre en compte lors de la recherche de la meilleure répartition. Elles seront sélectionnées aléatoirement.
En règle générale, la racine carrée du nombre total de caractéristiques fonctionne bien mais nous devrions vérifier jusqu'à 30-40% du nombre total de caractéristiques.
Des valeurs plus élevées peuvent conduire à un ajustement excessif, mais cela dépend de chaque cas.

5. Modèles – Boosting – Gradient Boosting

Paramètres de boosting : Ils affectent l'opération de boosting dans le modèle.

learning_rate : cela détermine l'impact de chaque arbre sur le résultat final (étape 2.4). Le GBM fonctionne en commençant par une estimation initiale qui est mise à jour en utilisant la sortie de chaque arbre. Le paramètre d'apprentissage contrôle l'ampleur de ce changement dans les estimations. Contrôle le dégré 'intensité avec lequel chaque arbre essaie de corriger les erreurs de l'arbre précédent. Valeurs élevées ➔ modèle plus complexe.
Des valeurs faibles sont généralement préférées car elles rendent le modèle robuste aux caractéristiques spécifiques de l'arbre et lui permettent ainsi de bien se généraliser.
Des valeurs plus faibles nécessiteraient un plus grand nombre d'arbres pour modéliser toutes les relations et seraient coûteuses en termes de calcul.

n_estimators : Le nombre d'arbres séquentiels à modéliser (étape 2). Valeur élevée ➔ modèle plus complexe.
Bien que le GBM soit assez robuste pour un nombre élevé d'arbres, il peut toujours être surajusté à un moment donné. Par conséquent, ce nombre doit être ajusté en utilisant CV pour un taux d'apprentissage particulier.

subsample : La fraction d'observations à sélectionner pour chaque arbre. La sélection se fait par échantillonnage aléatoire.
Des valeurs légèrement inférieures à 1 rendent le modèle robuste en réduisant la variance.
Les valeurs typiques ~0.8 fonctionnent généralement bien mais peuvent être affinées davantage.

Paramètres divers : Autres paramètres pour le fonctionnement global

loss : Il s'agit de la fonction de perte à minimiser dans chaque fractionnement. Elle peut avoir différentes valeurs pour la classification et la régression. En général, les valeurs par défaut fonctionnent bien. D'autres valeurs ne devraient être choisies que si vous comprenez leur impact sur le modèle.

init : Ceci affecte l'initialisation de la sortie. Ceci peut être utilisé si nous avons créé un autre modèle dont le résultat doit être utilisé comme estimations initiales pour GBM.

random_state : La graine de nombre aléatoire afin que les mêmes nombres aléatoires soient générés à chaque fois.
Ceci est important pour le réglage des paramètres. Si nous ne fixons pas le nombre aléatoire, nous aurons des résultats différents pour les exécutions suivantes sur les mêmes paramètres et il devient difficile de comparer les modèles.
Cela peut potentiellement entraîner un surajustement à un échantillon aléatoire particulier sélectionné. Nous pouvons essayer d'exécuter des modèles pour différents échantillons aléatoires, ce qui est coûteux en termes de calcul et n'est généralement pas utilisé.

verbose : Le type de sortie à imprimer lorsque le modèle s'ajuste. Les différentes valeurs peuvent être :
0 : aucune sortie générée (par défaut)
1 : sortie générée pour les arbres dans certains intervalles
>1 : sortie générée pour tous les arbres

warm_start : Ce paramètre a une application intéressante et peut être d'une grande aide s'il est utilisé judicieusement.
Il permet d'ajuster des arbres supplémentaires sur les ajustements précédents d'un modèle. Cela peut faire gagner beaucoup de temps et vous devriez explorer cette option pour les applications avancées.

presort : Sélectionnez si vous voulez présélectionner les données pour des fractionnements plus rapides.
La sélection est automatique par défaut, mais elle peut être modifiée si nécessaire.

Modèles XGBoost

5. Modèles – Boosting – XGBoost

Extreme Gradient Boosting (2014) est une implémentation avancée du Gradient Boosting.

XGBoost est un algorithme d'arbre de décision boosté par le gradient. Il existe depuis 2014 et a fini par dominer Kaggle et la communauté des sciences des données. XGB a introduit le boosting par gradient, où de nouveaux modèles sont adaptés aux résidus des modèles précédents, puis ajoutés ensemble, en utilisant un algorithme de descente de gradient pour minimiser la perte.

Paramètres généraux : Ils définissent la fonctionnalité globale de XGBoost.

booster [default=gbtree] Sélectionnez le type de modèle à exécuter à chaque itération. Il y a 2 options : gbtree : modèles basés sur des arbres gblinear : modèles linéaires

silent [default=0] : Le mode silencieux est activé est fixé à 1, c'est-à-dire qu'aucun message d'exécution ne sera imprimé. Il est généralement bon de garder la valeur 0 car les messages peuvent aider à comprendre le modèle.

nthread [par défaut, le nombre maximum de threads disponibles si non défini]. Ceci est utilisé pour le traitement parallèle et le nombre de cœurs dans le système doit être entré.Si vous souhaitez exécuter sur tous les cœurs, la valeur ne doit pas être saisie et l'algorithme le détectera automatiquement.

Paramètres de la tâche d'apprentissage : ces paramètres sont utilisés pour définir l'objectif d'optimisation et la métrique à calculer à chaque étape.

objectif [default=reg:linear] : Ceci définit la fonction de perte à minimiser. Les valeurs les plus utilisées sont :
binary:logistic -régression logistique pour la classification binaire, renvoie la probabilité prédite (et non la classe).
multi:softmax -classification multiclasse utilisant l'objectif softmax, renvoie la classe prédite (pas les probabilités)
Vous devez également définir un paramètre supplémentaire num_class (nombre de classes) définissant le nombre de classes uniques.

5. Modèles – Boosting – XGBoost

Paramètres du Booster :

eta [default=0.3] / Analogue au taux d'apprentissage dans GBM Rend le modèle plus robuste en réduisant les poids à chaque étape. Valeurs finales typiques à utiliser : 0.01-0.2

min_child_weight [default=1]/ Définit la somme minimale des poids de toutes les observations requises dans un enfant.
Ceci est similaire à min_child_leaf dans GBM mais pas exactement. Cela fait référence à la "somme des poids" minimale des observations alors que GBM a un "nombre d'observations" minimal.
Utilisé pour contrôler l'over-fitting. Des valeurs plus élevées empêchent un modèle d'apprendre des relations qui pourraient être très spécifiques à l'échantillon particulier sélectionné pour un arbre.
Des valeurs trop élevées peuvent conduire à un sous-ajustement et doivent donc être réglées à l'aide de CV.

max_depth [default=6]/ La profondeur maximale d'un arbre, comme GBM. Utilisé pour contrôler l'excès d'ajustement car une profondeur plus élevée permettra au modèle d'apprendre des relations très spécifiques à un échantillon particulier.
Doit être réglé à l'aide de CV. Valeurs typiques : 3-10

max_leaf_nodes : Le nombre maximum de nœuds terminaux ou de feuilles dans un arbre. Peut être défini à la place de max_depth. Comme les arbres binaires sont créés, une profondeur de 'n' produirait un maximum de 2^n feuilles.
Si cette valeur est définie, GBM ignorera max_depth.

gamma [default=0] : Un nœud est divisé uniquement lorsque la division qui en résulte donne une réduction positive de la fonction de perte. Gamma spécifie la réduction minimale des pertes requise pour effectuer un fractionnement.
Rend l'algorithme conservateur. Les valeurs peuvent varier en fonction de la fonction de perte et doivent être ajustées.

max_delta_step [default=0] : Le pas delta maximum que nous autorisons pour l'estimation du poids de chaque arbre. Si la valeur est fixée à 0, cela signifie qu'il n'y a pas de contrainte. S'il est défini à une valeur positive, cela peut aider à rendre l'étape de mise à jour plus conservatrice.
Habituellement, ce paramètre n'est pas nécessaire, mais il peut être utile dans la régression logistique lorsque la classe est extrêmement déséquilibrée.
Il n'est généralement pas utilisé mais vous pouvez l'explorer davantage si vous le souhaitez.

subsample [default=1] : Identique au sous-échantillon de GBM. Dénote la fraction d'observations à échantillonner de manière aléatoire pour chaque arbre.
Des valeurs faibles rendent l'algorithme plus conservateur et empêchent l'ajustement excessif, mais des valeurs trop faibles peuvent conduire à un ajustement insuffisant.
Valeurs typiques : 0.5-1

colsample_bytree [default=1] (en anglais) : Similaire à max_features dans GBM. Indique la fraction de colonnes à échantillonner de manière aléatoire pour chaque arbre.
Valeurs typiques : 0.5-1

colsample_bylevel [default=1] : Indique le rapport de sous-échantillonnage des colonnes pour chaque division, dans chaque niveau.
Je ne l'utilise pas souvent car subsample et colsample_bytree feront le travail pour vous. Mais vous pouvez explorer davantage si vous le souhaitez.

lambda [default=1] : Terme de régularisation L2 sur les poids (analogue à la régression Ridge)
Ceci est utilisé pour gérer la partie régularisation de XGBoost. Bien que de nombreux data scientists ne l'utilisent pas souvent, il devrait être exploré pour réduire l'overfitting.

alpha [default=0] : Terme de régularisation L1 sur le poids (analogue à la régression Lasso)
Peut être utilisé en cas de dimensionnalité très élevée afin que l'algorithme s'exécute plus rapidement lorsqu'il est mis en œuvre.

scale_pos_weight [par défaut=1] : Une valeur supérieure à 0 devrait être utilisée en cas de fort déséquilibre des classes, car elle permet une convergence plus rapide.

5. Modèles – Boosting – XGBoost

Avantages (%GBM) :

- Régularisation : L'implémentation GBM standard n'a pas de régularisation comme XGBoost, ce qui permet de réduire l'overfitting. En fait, XGBoost est également connu comme une technique de "boosting régularisé".
- Traitement parallèle : XGBoost met en œuvre un traitement parallèle et est incroyablement plus rapide que GBM.

Mais attendez, nous savons que le boosting est un processus séquentiel, alors comment peut-il être parallélisé ? Nous savons que chaque arbre ne peut être construit qu'après le précédent, alors qu'est-ce qui nous empêche de faire un arbre en utilisant tous les cœurs ? J'espère que vous comprenez où je veux en venir. Consultez ce lien pour aller plus loin.

- XGBoost supporte également l'implémentation sur Hadoop.
- Grande flexibilité : XGBoost permet aux utilisateurs de définir des objectifs d'optimisation et des critères d'évaluation personnalisés. Cela ajoute une toute nouvelle dimension au modèle et il n'y a aucune limite à ce que nous pouvons faire.
- Traitement des valeurs manquantes : XGBoost dispose d'une routine intégrée pour gérer les valeurs manquantes. L'utilisateur doit fournir une valeur différente des autres observations et la passer comme paramètre. XGBoost essaie différentes choses lorsqu'il rencontre une valeur manquante sur chaque nœud et apprend quel chemin prendre pour les valeurs manquantes à l'avenir.
- Élagage de l'arbre : Un GBM arrête de diviser un nœud lorsqu'il rencontre une perte négative dans la division. Il s'agit donc plutôt d'un algorithme avide. XGBoost, quant à lui, effectue des scissions jusqu'à la profondeur maximale spécifiée, puis commence à élaguer l'arbre en arrière et supprime les scissions au-delà desquelles il n'y a pas de gain positif. Un autre avantage est que parfois, une division de perte négative, par exemple -2, peut être suivie d'une division de perte positive +10. GBM s'arrêtera lorsqu'il rencontrera -2. Mais XGBoost ira plus loin et verra un effet combiné de +8 de la division et gardera les deux.
- Validation croisée intégrée : XGBoost permet à l'utilisateur d'exécuter une validation croisée à chaque itération du processus de boosting et il est donc facile d'obtenir le nombre optimal exact d'itérations de boosting en une seule exécution. Contrairement à GBM, où nous devons effectuer une recherche sur grille et où seules quelques valeurs peuvent être testées.
- Continuer sur le modèle existant : L'utilisateur peut commencer à former un modèle XGBoost à partir de la dernière itération de l'exécution précédente. Cela peut être un avantage significatif dans certaines applications spécifiques. L'implémentation GBM de sklearn possède également cette fonctionnalité, ils sont donc à égalité sur ce point.
- Ancien : documentation et beaucoup d’exemples

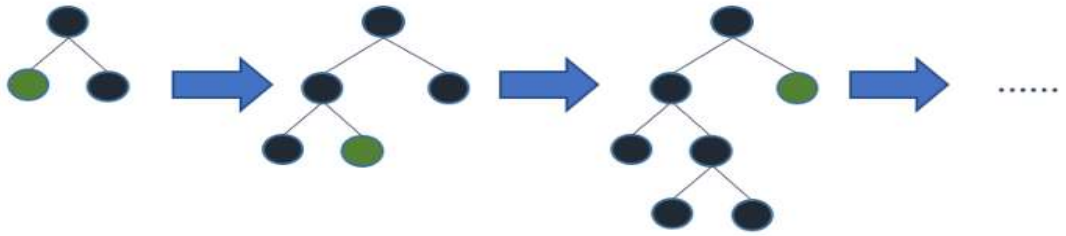
Modèles LightGBM

5. Modèles – Boosting - LightGBM

LightGBM :

LightGBM est un cadre GBDT open-source créé par Microsoft (2017) comme une alternative rapide et évolutive à XGB et GBM. Par défaut, LightGBM entraîne un arbre de décision à boost de gradient (GBDT), mais il prend également en charge les forêts aléatoires, les arbres de régression additifs multiples (DART) et l'échantillonnage unilatéral basé sur le gradient (Goss).

Light GBM est un cadre rapide, distribué et très performant de boosting de gradient basé sur l'algorithme de l'arbre de décision, basé sur la croissance de l'arbre par feuilles, contrairement aux autres algorithmes qui fonctionnent selon une approche par niveaux. Il divise l'arbre **par feuille** avec le meilleur ajustement alors que d'autres algorithmes de boosting divisent l'arbre par profondeur ou par niveau plutôt que par feuille. Ainsi, lors de la croissance sur la même feuille dans Light GBM, l'algorithme par feuille peut réduire plus de pertes que l'algorithme par niveau, ce qui donne une bien meilleure précision, rarement atteinte par les algorithmes de boosting existants. En outre, il est étonnamment très rapide, d'où le terme "léger".



Leaf-wise tree growth

5. Modèles – Boosting - LightGBM

Avantages :

- Vitesse de formation plus rapide et efficacité accrue : Light GBM utilise un algorithme basé sur l'histogramme, c'est-à-dire qu'il classe les valeurs de caractéristiques continues dans des cases discrètes, ce qui accélère la procédure de formation.
- Utilisation réduite de la mémoire : Remplace les valeurs continues par des cases discrètes, ce qui réduit l'utilisation de la mémoire.
- Meilleure précision que tout autre algorithme de boosting : Il produit des arbres beaucoup plus complexes en suivant une approche de division par feuille plutôt qu'une approche par niveau, ce qui est le principal facteur permettant d'obtenir une meilleure précision. Cependant, il peut parfois conduire à un surajustement qui peut être évité en définissant le paramètre max_depth.
- Compatibilité avec les grands ensembles de données : Il est capable de réaliser d'aussi bonnes performances avec de grands ensembles de données avec une réduction significative du temps de formation par rapport à XGBOOST.
- Prise en charge de l'apprentissage parallèle.

Inconvénients :

- Over-fitting : Les divisions en feuilles entraînent une augmentation de la complexité et peuvent conduire à un surajustement, ce qui peut être résolu en spécifiant un autre paramètre, max-depth, qui spécifie la profondeur à laquelle la division aura lieu.
- Récent : moins de documentation mais en progression

Hyperparamètres :

max_depth : Spécifie la profondeur maximale à laquelle l'arbre se développera. Ce paramètre est utilisé pour gérer l'overfitting.

min_data_in_leaf : Nombre minimum de données dans une feuille.

feature_fraction : default=1 ; spécifie la fraction de caractéristiques à prendre pour chaque itération.

bagging_fraction : default=1 ; spécifie la fraction de données à utiliser pour chaque itération et est généralement utilisé pour accélérer l'apprentissage et éviter l'overfitting.

task : valeur par défaut = train ; options = train , prediction ; Spécifie la tâche que nous souhaitons effectuer qui est soit train soit prediction.

application : valeur par défaut=regression, type=enum, options= options :

regression : exécuter la tâche de régression

binary : classification binaire

multiclass : Classification multiclasse

lambdarank : application lambdarank

data : type=string ; données d'entraînement, LightGBM s'entraînera à partir de ces données

num_iterations : nombre d'itérations de boosting à effectuer ; default=100 ; type=int

num_leaves : nombre de feuilles dans un arbre ; default = 31 ; type =int

device : default= cpu ; options = gpu,cpu. Dispositif sur lequel nous voulons entraîner notre modèle. Choisissez GPU pour un entraînement plus rapide.

min_gain_to_split : default=.1 ; gain min pour effectuer le fractionnement

max_bin : nombre maximum de bacs pour mettre les valeurs des caractéristiques dans des cases.

min_data_in_bin : nombre minimum de données dans un bin.

num_threads : default=OpenMP_default, type=int ;Nombre de threads pour Light GBM.

label : type=string ; spécifier la colonne label

categorical_feature : type=string ; spécifie les caractéristiques catégorielles que nous voulons utiliser pour l'entraînement de notre modèle

num_class : default=1 ; type=int ; utilisé uniquement pour la classification multi-classes

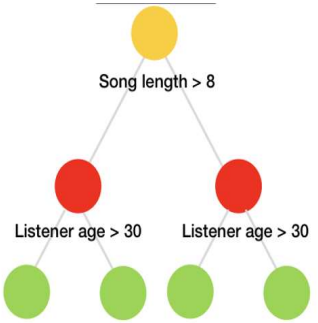
5

Modèles CatBoost

5. Modèles – Boosting - CatBoost

CatBoost est un algorithme d'apprentissage automatique récemment (2017) mis en libre accès par Yandex (russes) qui s'appuie sur la théorie des arbres de décision et du boosting de gradient.
Category : la bibliothèque fonctionne bien avec plusieurs catégories de données, telles que l'audio, le texte, l'image, y compris les données historiques.
Boosting : "Boost" vient de l'algorithme d'apprentissage automatique Gradient Boosting, car cette bibliothèque est basée sur la bibliothèque Gradient Boosting.

Dans la procédure de croissance des arbres de décision, CatBoost ne suit pas les modèles similaires de boosting de gradient. Au lieu de cela, CatBoost fait croître des arbres oblivious, ce qui signifie que les arbres sont cultivés en imposant la règle selon laquelle tous les nœuds au même niveau, testent le même prédicteur avec la même condition, et donc l'index d'une feuille peut être calculé avec des opérations bitwise. La procédure d'arbre oblivious permet un schéma d'ajustement simple et efficace sur les CPU, tandis que la structure de l'arbre fonctionne comme une régularisation pour trouver une solution optimale et éviter l'overfitting.



Avantage :

CatBoost peut traiter automatiquement les variables catégorielles, les données audio, texte.., ne nécessite pas de prétraitement approfondi des données comme d'autres algorithmes d'apprentissage automatique. Performant .Robuste (peu de réglage d'hyperparamètres). Feature importances. Utilisation GPU. Gère valeurs manquantes. Régression/Classification.

Inconvénient :

- Long si pas GPU. Récent : pas de documentation et peu d'exemples.

5. Modèles – Boosting - CatBoost

Hyperparamètres :

loss_function : alias as objective - Métrique utilisée pour l'apprentissage. Il s'agit de métriques de régression telles que l'erreur quadratique moyenne pour la régression et le logloss pour la classification.

Iterations : Le nombre maximum d'arbres qui peuvent être construits. Le nombre final d'arbres peut être inférieur ou égal à ce nombre (1000 par défaut).

learning_rate : - Le taux d'apprentissage qui détermine la rapidité ou la lenteur de l'apprentissage du modèle. La valeur par défaut est généralement de 0,03. Utilisé pour réduire le pas du gradient.

border_count : Il spécifie le nombre de splits pour les caractéristiques numériques. Il est similaire au paramètre max_bin.

depth : Définit la profondeur des arbres.

eval_metric - Métrique utilisée pour détecter le surajustement.

random_seed : La graine aléatoire utilisée pour l'apprentissage.

l2_leaf_reg alias reg_lambda - Coefficient du terme de régularisation L2 de la fonction de coût. La valeur par défaut est 3,0.

min_data_in_leaf alias min_child_samples - Il s'agit du nombre minimum d'échantillons d'entraînement dans une feuille. Ce paramètre est uniquement utilisé avec les politiques Lossguide et Depthwise growing.

max_leaves alias num_leaves - Ce paramètre est utilisé uniquement avec la politique Lossguide et détermine le nombre de feuilles dans l'arbre.

ignored_features - Indique les caractéristiques qui doivent être ignorées dans le processus de formation.

bootstrap_type - Détermine la méthode d'échantillonnage pour les poids des objets, par exemple Bayesian, Bernoulli, MVS et Poisson.

grow_policy - Détermine la façon dont l'algorithme de recherche gloutonne sera appliqué. Il peut être soit SymmetricTree, Depthwise, ou Lossguide. SymmetricTree est la valeur par défaut. Dans SymmetricTree, l'arbre est construit niveau par niveau jusqu'à ce que la profondeur soit atteinte. À chaque étape, les feuilles de l'arbre précédent sont divisées avec la même condition. Lorsque l'option Depthwise est choisie, un arbre est construit étape par étape jusqu'à ce que la profondeur spécifiée soit atteinte. À chaque étape, toutes les feuilles non terminales du dernier niveau de l'arbre sont séparées. Les feuilles sont divisées en utilisant la condition qui entraîne la meilleure amélioration des pertes. Dans Lossguide, l'arbre est construit feuille par feuille jusqu'à ce que le nombre de feuilles spécifié soit atteint. À chaque étape, la feuille non terminale présentant la meilleure amélioration des pertes est divisée.

nan_mode - Méthode de traitement des valeurs manquantes. Les options sont Interdit, Min et Max. La valeur par défaut est Min. Lorsque l'option Forbidden est utilisée, la présence de valeurs manquantes entraîne des erreurs. Avec Min, les valeurs manquantes sont considérées comme les valeurs minimales pour cette caractéristique. Avec Max, les valeurs manquantes sont traitées comme la valeur maximale pour cette caractéristique.

leaf_estimation_method - La méthode utilisée pour calculer les valeurs dans les feuilles. Dans la classification, 10 itérations de Newton sont utilisées. Les problèmes de régression utilisant la perte quantile ou MAE utilisent une itération Exact. La classification multiple utilise une itération Netwon.

leaf_estimation_backtracking - Le type de backtracking à utiliser pendant la descente de gradient. La valeur par défaut est AnyImprovement. AnyImprovement réduit le pas de descente, jusqu'à ce que la valeur de la fonction de perte soit plus petite que lors de la dernière itération. Armijo réduit le pas de descente jusqu'à ce que la condition Armijo soit remplie.

5. Modèles – Boosting - CatBoost

Hyperparamètres :

boosting_type - Le schéma de boosting. Il peut être simple pour le schéma classique de boosting par gradient, ou ordonné, qui offre une meilleure qualité sur les petits ensembles de données.

score_function - Le type de score utilisé pour sélectionner la prochaine division pendant la construction de l'arbre. Cosinus est l'option par défaut. Les autres options disponibles sont L2, NewtonL2 et NewtonCosine.

early_stopping_rounds - Lorsque True, définit le type de détecteur d'overfitting à Iter et arrête l'apprentissage lorsque la métrique optimale est atteinte.

classes_count - Le nombre de classes pour les problèmes de multi-classification.

task_type - Si vous utilisez un CPU ou un GPU. CPU est la valeur par défaut.

devices - Les ID des périphériques GPU à utiliser pour la formation.

cat_features - Le tableau avec les colonnes catégoriques.

text_features - Utilisé pour déclarer les colonnes de texte dans les problèmes de classification.

Avantage :

- CatBoost peut traiter automatiquement les variables catégorielles et ne nécessite pas de prétraitement approfondi des données comme d'autres algorithmes d'apprentissage automatique. Performant .
- Robuste (peu de réglage d’hyperparamètres).
- Feature importances.
- Utilisation GPU.
- Gère valeurs manquantes.
- Elle produit des résultats de pointe sans la formation intensive des données généralement requise par les autres méthodes d'apprentissage automatique, et
- Elle offre une prise en charge puissante et prête à l'emploi des formats de données plus descriptifs qui accompagnent de nombreux problèmes commerciaux.

Modèles Boosting Comparaison

5. Modèles – Boosting - Comparaison

Function	XGBoost	CatBoost	Light GBM
Important parameters which control overfitting	1. **learning_rate or eta** – optimal values lie between 0.01-0.2 2. **max_depth** 3. **min_child_weight:** similar to min_child leaf; default is 1	1. **Learning_rate** 2. **Depth** - value can be any integer up to 16. Recommended - [1 to 10] 3. No such feature like min_child_weight 4. **l2-leaf-reg**: L2 regularization coefficient. Used for leaf value calculation (any positive integer allowed)	1. **learning_rate** 2. **max_depth**: default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune **num_leaves** (number of leaves in a tree) which should be smaller than 2^(max_depth). It is a very important parameter for LGBM 3. **min_data_in_leaf**: default=20, alias= min_data, min_child_samples
Parameters for categorical values	Not Available	1. **cat_features**: It denotes the index of categorical features 2. **one_hot_max_size**: Use one-hot encoding for all features with number of different values less than or equal to the given parameter value (max – 255)	1. **categorical_feature**: specify the categorical features we want to use for training our model
Parameters for controlling speed	1. **colsample_bytree**: subsample ratio of columns 2. **subsample**: subsample ratio of the training instance 3. **n_estimators**: maximum number of decision trees; high value can lead to overfitting	1. **rsm**: Random subspace method. The percentage of features to use at each split selection 2. No such parameter to subset data 3. **iterations**: maximum number of trees that can be built; high value can lead to overfitting	1. **feature_fraction:** fraction of features to be taken for each iteration 2. **bagging_fraction:** data to be used for each iteration and is generally used to speed up the training and avoid overfitting 3. **num_iterations:** number of boosting iterations to be performed; default=100

Modèles Ensemblistes Stacking

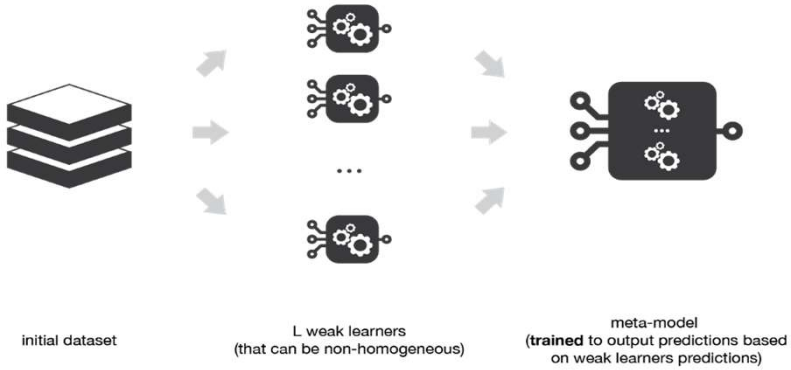
5. Modèles – Ensemblistes - Stacking

Stacking :

Apprendre plusieurs apprenants faibles différents et les combiner en formant un méta-modèle pour produire des prédictions basées sur les multiples prédictions renvoyées par ces modèles faibles.

Ainsi, nous devons définir deux choses afin de construire notre modèle de stacking : les L learners que nous voulons adapter et le méta-modèle qui les combine.

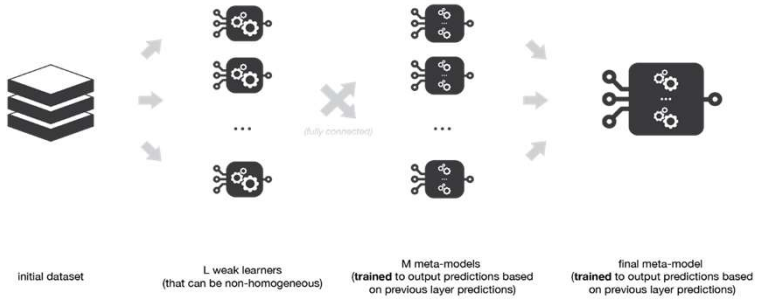
- diviser les données d'apprentissage en deux plis
- choisir L apprenants faibles et les adapter aux données du premier pli
- pour chacun des L apprenants faibles, faire des prédictions pour les observations du second pli
- ajuster le méta-modèle sur le second pli, en utilisant les prédictions faites par les apprenants faibles comme entrées.



Une extension possible du stacking est le stacking multi-niveaux.
Elle consiste à faire un empilage avec plusieurs couches.

Désavantage :

l'ajout de niveaux peut être soit coûteux en données (si la technique des k-folds n'est pas utilisée et que, par conséquent, davantage de données sont nécessaires), soit coûteux en temps (si la technique des k-folds est utilisée et que, par conséquent, de nombreux modèles doivent être ajustés).



Comparer Modèles

6. Comparer modèles

La qualité d'une fonction est généralement évaluée à l'aide d'une fonction de perte.
Compte tenu d'un problème, d'un type de modèle et d'une fonction de perte choisis, la tâche de recherche du meilleur modèle est exprimée en un problème d'optimisation.
Ici, nous essayons de minimiser une fonction de coût par rapport aux paramètres du modèle (pour un modèle linéaire, les coefficients (ou poids) et le biais). Une fonction de coût est la moyenne des pertes calculées sur tous les ensembles de données d'apprentissage.
Fonctions de perte de régression :

1. Mean Squared Error (Erreur quadratique moyenne)/ perte quadratique (MSE)
2. Erreur absolue moyenne (MAE)
3. Smooth MAE
4. Log cosh Loss
5. sPerte quantile

Problème d'optimisation/ Minimisation des coûts :
l'objectif est de trouver les paramètres du modèle (poids et biais) qui minimisent la fonction de coût.
L'optimisation est réalisée à l'aide d'algorithmes comme la descente de gradient, la descente de gradient stochastique, etc.

Partial derivative of the cost function w.r.t. j^{th} parameter

$$\frac{\delta}{\delta\theta_j} C_{MSE}(\theta) \qquad \frac{\delta}{\delta\theta_j} C_{LogLoss}(\theta)$$

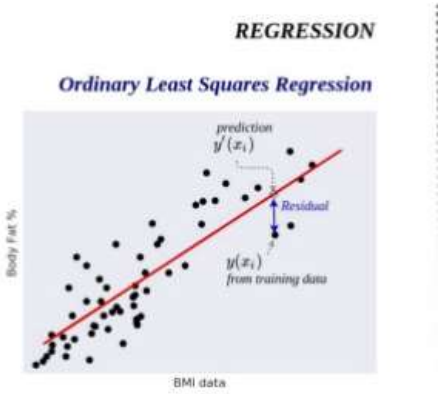
Gradient vector of cost function: Partial derivative of the cost function all parameters

$$\nabla_\theta C_{MSE}(\theta) \qquad \nabla_\theta C_{LogLoss}(\theta)$$

Gradient descent Step

$$\theta^{\text{next step}} = \theta - \eta\nabla_\theta C_{MSE}(\theta) \qquad \theta^{\text{next step}} = \theta - \eta\nabla_\theta C_{LogLoss}(\theta)$$

Learning rate/ Step size



Residual/ error $= \text{training target value - predicted value} = y - y'$

Loss function $L(y, y') = \frac{1}{2}(y - y')^2$

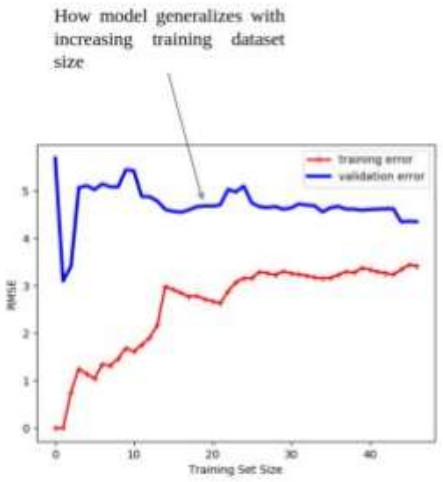
Cost Function Mean Squared Error (**MSE**) $C(\theta) = \frac{1}{N}\Sigma_{i=1}^{N} L(y_i, y_i')$

6. Comparer modèles

Mesure de la performance du modèle :

Regression performance measure & model evaluation	*Mathematical expression/ definition*
Mean Squared Error	$MSE(y, y') = \frac{1}{n_{samples}} \Sigma_{i=0}^{n_{samples}-1} (y_i - y'_i)^2$
Root Mean Squared Error	$RMSE(y, y') = \sqrt{MSE(y, y')}$
Mean Absolute Error	$MSE(y, y') = \frac{1}{n_{samples}} \Sigma_{i=0}^{n_{samples}-1} \lvert y_i - y'_i \rvert$
Explained Variance score Best possible score = 1.0, worsens with lower values	$exp.\ var(y, y') = 1 - \frac{Variance\ \{y - y'\}}{Variance\ \{y\}}$
R2 Score It provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model.	$R^2(y, y') = 1 - \frac{\Sigma_{i=1}^{n} (y_i - y'_i)^2}{\Sigma_{i=1}^{n} (y - (\frac{1}{n} \Sigma_{i=1}^{n} (y_i)))^2}$
Cross Validation Evaluation *using Scikit learn's K fold cross validation* Estimates Model's generalization performance	Randomly split the training dataset into *K* distinct subsets called *folds* and it trains and evaluates the model *K* times, picking a different fold for evaluation every time and training on the other *K-1* folds.
Learning Curves	Plots a model's performance on the training dataset and validation dataset as a function of training set size.

Learning curves for our regression model



Mean Square Error (MSE) L2 est la fonction de perte de régression la plus couramment utilisée. MSE est la somme des distances au carré entre notre variable cible et les valeurs prédites.

Mean Absolute Error (MAE) L1 est une autre fonction de perte utilisée pour les modèles de régression. MAE est la somme des différences absolues entre nos variables cibles et prédites. Elle mesure donc la magnitude moyenne des erreurs dans un ensemble de prédictions, sans tenir compte de leurs directions. Plus robustes aux outliers. PB son gradient est le même partout, ce qui signifie que le gradient sera important même pour de petites valeurs de perte. Ce n'est pas bon pour l'apprentissage. Pas utiliser pour les réseaux de neurones. Si les valeurs aberrantes représentent des anomalies importantes pour l'entreprise et doivent être détectées, alors nous devrions utiliser MSE. En revanche, si nous pensons que les valeurs aberrantes représentent simplement des données corrompues, nous devrions choisir MAE comme perte.

6. Comparer modèles

Mesure de la performance du modèle :

Choix de la métrique:

1. **MAE : Mean Average Error**
   Mesure de l'erreur moyenne réalisée

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

2. **RMSE : Root Mean Squared Error**
   Mesure de l'erreur pénalisant les grandes erreurs

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

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, ...

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Le R^2 ajusté est une version modifiée du R^2, et il est ajusté pour le nombre de variables indépendantes dans le modèle, et il sera toujours inférieur ou égal au R^2. Dans la formule ci-dessous, n est le nombre d'observations dans les données et k est le nombre de variables indépendantes dans les données.

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n-1)}{n-k-1}\right]$$

MAE: l'erreur absolue moyenne représente la moyenne de la différence absolue entre les valeurs réelles et prédites dans l'ensemble de données. Elle mesure la moyenne des résidus dans l'ensemble de données.

MSE : l'erreur quadratique moyenne représente la moyenne de la différence au carré entre les valeurs originales et prédites dans l'ensemble de données. Elle mesure la variance des résidus.

RMSE : l'erreur quadratique moyenne est la racine carrée de l'erreur quadratique moyenne. Elle mesure l'écart-type des résidus.

R2 : Le coefficient de détermination ou R-carré représente la proportion de la variance de la variable dépendante qui est expliquée par le modèle de régression linéaire. Il s'agit d'un score sans échelle, c'est-à-dire qu'indépendamment du fait que les valeurs soient petites ou grandes, la valeur du carré R sera inférieure à un.

6. Comparer modèles

Différences entre ces mesures d'évaluation :

- MSE et RMSE pénalisent les erreurs de prédiction importantes par rapport à la MAE. Cependant, RMSE est plus largement utilisée que MSE pour évaluer la performance du modèle de régression avec d'autres modèles aléatoires car elle a les mêmes unités que la variable dépendante (axe Y).

- La MSE est une fonction différentiable qui facilite les opérations mathématiques par rapport à une fonction non différentiable comme la MAE. Par conséquent, dans de nombreux modèles, lRMSE est utilisée comme métrique par défaut pour calculer la fonction de perte, bien qu'elle soit plus difficile à interpréter que la MAE.

- MAE est plus robuste aux données comportant des valeurs aberrantes.

- Une valeur plus faible de MAE, MSE et RMSE implique une plus grande précision d'un modèle de régression. Cependant, une valeur plus élevée du R carré est considérée comme souhaitable.

- Le R au carré et le R au carré ajusté sont utilisés pour expliquer dans quelle mesure les variables indépendantes du modèle de régression linéaire expliquent la variabilité de la variable dépendante. La valeur du R carré augmente toujours avec l'ajout des variables indépendantes, ce qui pourrait conduire à l'ajout de variables redondantes dans notre modèle. Cependant, le R au carré ajusté résout ce problème.

- Le R au carré ajusté prend en compte le nombre de variables prédictives, et il est utilisé pour déterminer le nombre de variables indépendantes dans notre modèle. La valeur du R au carré ajusté diminue si l'augmentation du R au carré par la variable supplémentaire n'est pas assez significative.

- Pour comparer la précision de différents modèles de régression linéaire, la RMSE est un meilleur choix que le R au carré.

- Par conséquent, si l'on compare la précision de prédiction entre différents modèles de régression linéaire (LR), la RMSE est une meilleure option car elle est simple à calculer et différentiable. Toutefois, si votre ensemble de données comporte des valeurs aberrantes, choisissez MAE plutôt que RMSE.

- En outre, le nombre de variables prédicteurs dans un modèle de régression linéaire est déterminé par le R au carré ajusté, et choisissez RMSE plutôt que R au carré ajusté si vous vous souciez d'évaluer la précision de la prédiction parmi différents modèles LR.

Coefficient of Determination (R Square)

$$R^2 = \frac{SSR}{SST}$$

$$SSR = \sum_i (\hat{y}_i - \bar{y})^2$$

$$SST = \sum_i (y_i - \bar{y})^2$$

Where,

- SSR is Sum of Squared Regression also known as variation explained by the model
- SST is Total variation in the data also known as sum of squared total
- y_i is the y value for observation i
- y_bar is the mean of y value
- y_bar_hat is predicted value of y for observation i