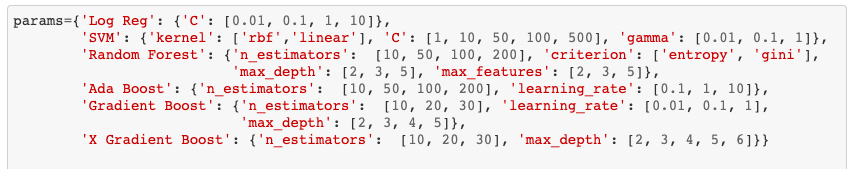
**Explication des différents modèles utilisés**

Valeurs des hyperparamètres choisis pour le projet

****

1. **Régression logistique**

Principe : Elle estime la probabilité qu’une observation appartienne à une classe particulière.

Hyperparamètre : C

C contrôle l’importance de la régularisation du modèle. Plus la valeur de C est élevée, moins le modèle est régularisé.

Rappel : La régularisation consiste à imposer des contraintes à un modèle pour le simplifier et réduire le risque de surajustement (overfitting).

1. **SVM**

Principe : Ajuste un chemin aussi large que possible entre les classes. L’objectif est d’avoir la marge la plus large possible entre la frontière de décision qui sépare les deux classes et les observations d’entraînement. Une fois le SVM entraîné, on appelle vecteur de support toute observation se trouvant sur le chemin y compris sa bordure.

Hyperparamètres : C et gamma.

Gamma est un hyperparamètre de régularisation. Si le modèle surajuste, on doit réduire gamma. S’il sous-ajuste, on doit l’augmenter (de manière similaire à C).

1. **Random Forest (ensemble d’arbres de décision)**

Algorithme d’apprentissage d’ensemble (ou méthode ensembliste).

Principe d’une forêt aléatoire : On entraîne un ensemble d’arbres de décision, chacun sur un sous-ensemble aléatoire différent du jeu d’entraînement. Pour obtenir des prédictions, il suffit d’obtenir les prédictions de chacun des arbres, puis de choisir la classe obtenant le plus de votes.

Pour éviter le surajustement, on doit restreindre la liberté des arbres durant l’entraînement.

n\_estimators : This is the number of trees you want to build before taking the maximum voting or averages of predictions. Higher number of trees give you better performance but makes your code slower. You should choose as high value as your processor can handle because this makes your predictions stronger and more stable.

Mesure d’impureté entropie: l’entropie d’un jeu de donnée est nulle lorsque toutes ses observations appartiennent à une seule classe. On souhaite donc que l’entropie soit la plus faible possible à la fin des arbres (donc que les données soient les mieux classées possibles).

Impureté gini : Mesure l’impureté d’un nœud (ou feuille). Un nœud est pur (gini=0) si toutes les observations d’entraînement qui y aboutissent appartiennent à la même classe.

max\_depth : Permet de limiter la profondeur maximale d’un arbre. Une profondeur correspond à un étage de l’arbre. En réduisant max\_depth on régularise le modèle et on réduit par conséquent le risque de surajustement.

max\_features : maximum de variables que l’algorithme d’entraînement va examiner pour toutes les observations de chaque nœud.

-----------------------------------------------------------------------------------------------------------------

**BOOSTING**: Méthode d’ensemble qui peut combiner plusieurs mauvais élèves en un bon élève.

Idée générale des méthodes de boosting : Entraîner des prédicteurs (une régression linéaire est par exemple un prédicteur) l’un après l’autre, chacun s’efforçant de corriger son prédécesseur.

Les deux méthodes de boosting les plus connues sont : AdaBoost et gradient boosting.

1. **AdaBoost**

Principe: AdaBoost helps you **combine multiple “weak classifiers” into a single “strong classifier”**.

The weak learners in AdaBoost are decision trees with a single split, called decision stumps.

Technique utilisée : Pour un nouveau prédicteur, l’un des moyens de corriger son prédécesseur consiste à prêter plus d’attention aux observations d’entraînement que ce prédécesseur a sous-ajustées. De cette façon, les nouveaux prédicteurs se concentrent de plus en plus sur les cas difficiles.

Pour construire un classificateur AdaBoost, un premier classificateur de base (tel qu’un arbre de décision) est entraîné puis utilisé pour effectuer des prédictions sur le jeu d’entraînement. Le poids relatif des observations d’entraînement mal classées est alors accru. Un second classificateur utilisant les poids modifiés est alors entraîné puis effectue des prédictions sur le jeu d’entraînement, les poids sont modifiés, et ainsi de suite.

**C’est une technique d’apprentissage séquentielle** qui présente des similitudes avec la descente de gradient. Au lieu d’ajuster les paramètres d’un seul prédicteur pour minimiser une fonction de coût, AdaBoost ajoute des prédicteurs à l’ensemble en les améliorant progressivement.

Une fois tous les prédicteurs entraînés, l’ensemble effectue des prédictions de manière similaire au bagging, à ceci près que les prédicteurs ont des poids différents compte tenu de leur exactitude globale sur le jeu d’entraînement pondéré.

1. **Gradient Boost (GB)**

**Principe :** *combiner les résultats d’un ensemble de modèles plus simple et plus faibles afin de fournir une meilleure prédiction.*

On parle d’ailleurs de méthode d’agrégation de modèles. L’idée est donc simple : au lieu d’utiliser un seul modèle, l’algorithme va en utiliser plusieurs qui seront ensuite combinés pour obtenir un seul résultat.

Tout comme AdaBoost, le GB travaille par ajout séquentiel de prédicteurs à un ensemble, chacun d’eux corrigeant son prédécesseur. Cependant, au lieu de modifier légèrement les poids des observations à chaque itération comme le fait AdaBoost, cette méthode tente d’ajuster un nouveau prédicteur aux erreurs résiduelles du prédicteur précédent.

Les prédictions d’ensemble s’améliorent au fur et à mesure de l’ajout d’arbres à l’ensemble.

**Hyperparamètre learning rate** : limite la contribution de chaque arbre. Si on lui donne une valeur faible, il faudra davantage d’arbres dans l’ensemble pour ajuster le jeu d’entraînement, mais d’ordinaire les prédictions se généraliseront mieux. C’est une technique de régularisation appelée *rétrécissement*.

1. **X Gradient Boost:** e**X**treme **G**radient **B**oosting.

C’est une implémentation open source optimisée de l’algorithme d’arbres de boosting de gradient.

Principe : l’algorithme travaille de manière séquentielle. Contrairement au Random Forest. Cette façon de faire va le rendre plus lent bien sûr mais il va surtout permettre à l’algorithme de s’améliorer par capitalisation par rapport aux exécutions précédentes. Il commence donc par construire un premier modèle qu’il va bien sur évaluer (on est bien sur de l’apprentissage supervisé). A partir de cette première évaluation, chaque individu va être alors pondéré en fonction de la performance de la prédiction. Etc.

XGBoost is similar to gradient boosting algorithm but it has a few tricks up its sleeve which makes it stand out from the rest. Features of XGBoost are: Clever Penalisation of Trees, A Proportional shrinking of leaf, extra Randomisation Parameter.

XGBoost is a specific **implementation of the Gradient Boosting** method which delivers more accurate approximations by using the strengths of **second order derivative of the loss function**.

Vidéo explicative : <https://www.youtube.com/watch?v=YABWwCLPfZs>