XG BOOST

Extreme Gradient Boost

- **XGBoost**

une méthode d’ensembling du gradient boost ,en combinant plusieurs modèles simple(arbre de décision) pour former un model prédictif plus avancée et plus performant dans le cadre de l’apprentissage supervisée
Elle est utilisée pour résoudre des problèmes de classification insi de régression .

-L’Algorithm de Xgboost est consideree parmis les Meilleur algorithms , car il permet un equilibre entre la variance et le biais du modele en introduisant la regularisation .
Car introduire une penalitee va reduire la complexitee de l’arbre, puis eviter le surajustement (diminuer la variance .
Par consequent, empecher le model de s’adapter au donne d’entrainement ,ce qui va diminuer de ce tromper

Une des particularite de XGboost c’est qu’il possede la possibilitee d’ajuster les parametres tels que :
-la profondeur des arbres de decision
- parametre de regularisation λ appliquer sur le nœud de l’arbre

- **Régularisation :**

  La régularisation est réalisée a travers l'ajustement de ces deux hyperparamètres

  Le but de cette dernière est de limiter la complexité du modèle , éviter le surajustement en contrôlant la variance , meilleur généralisation du modèle (sur l'ensemble de test)

- <u>Régularisation a travers shrinkidge :</u>
  Le taux d'apprentissage control la vitesse vers la quel le model atterrie vers la solution optimale avec le minimum d'erreur d'entrainement ,c'est a travers que les poids des arbres dans chaque itération est ajustée.
  Taux d'apprentissage élevée implique la construction des arbres de décision plus profond pour minimiser justement les erreurs pendant l'entrainement , qui veut dire surajustement .
  Taux d'apprentissage faible conduit vers sousajustement du model .
  « shrinkidge » consiste a réduire progressivement le taux d'apprentissage durant l'entrainement a fin d' atterire plus lentement vers la solution optimale
  insi , la profondeur des arbres est ajustée d'une manière régulière

- <u>Régularisation a travers la pénalité :</u>
  En ajoutant un terme de pénalité a la fonction de couts de cette algorithme lors de l'entrainement a fin de réduire les poids .

  Régularisation L1(λ) : lasso ; pénalité absolu sur les poids , implique une sélection des variables indépendant et l'annulation de certain .
  Régularisation L2(λ) : Ridge; pénalité quadratique sur les poids.

- **Modèle initiale :**

  Initialiser la valeur y_pred =0.5
  (quelque soit le probleme)

- **La construction de l’arbre :**

(pour classification binaire)

<u>exemple</u> :

x: dosage d’un médicament

y: médicament efficace ou pas

comme on a initialiser y_pred = 0.5 , veut dire le médicament est a 50% efficace

représentées par le trait en gras

Les points vert: dosage de médicament avec proba

qu’ils sont efficace =1

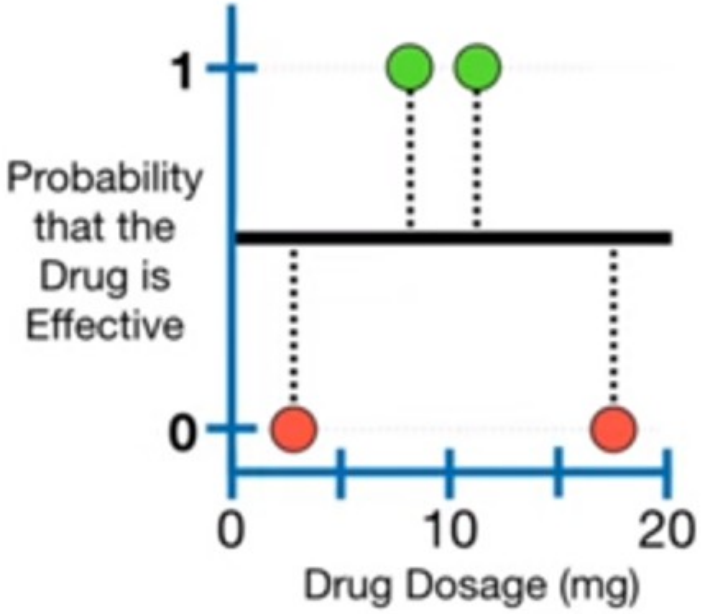
Les points rouge: dosage de médicament avec proba

qu’ils sont efficace =0

on calcule les résidus : y_reel - y_pred

A la création de l’arbre , on va ajuster les variab

résidus (y_reel-y_pred)



- *Fonction de cout :*

  La fonction logloss ou cross entropy loss permet de meusuer la qualitee de prediction d'un model de classification.

$$L(y_i, p_i) = -\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right] + \frac{1}{2}\lambda O_{value}^2$$

function de perte + function de regularisation

Fonction de perte : (log_loss)

$$L(y_i, p_i) = -\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right]$$

Gradien de la log_loss: (premiere derrivee)

Gradien_logloss = $-(y_i - p_i)$ = -(residu)

Hessien de la log_loss: (deuxieme derrivee)

Hess_logloss= $p_i \times (1 - p_i)$

- Score de similarité:

Dans un premier temps on va mettre tt les residus dans le noeud :

-0.5, 0.5, 0.5, -0.5

Puis,calculer le score de similarité :

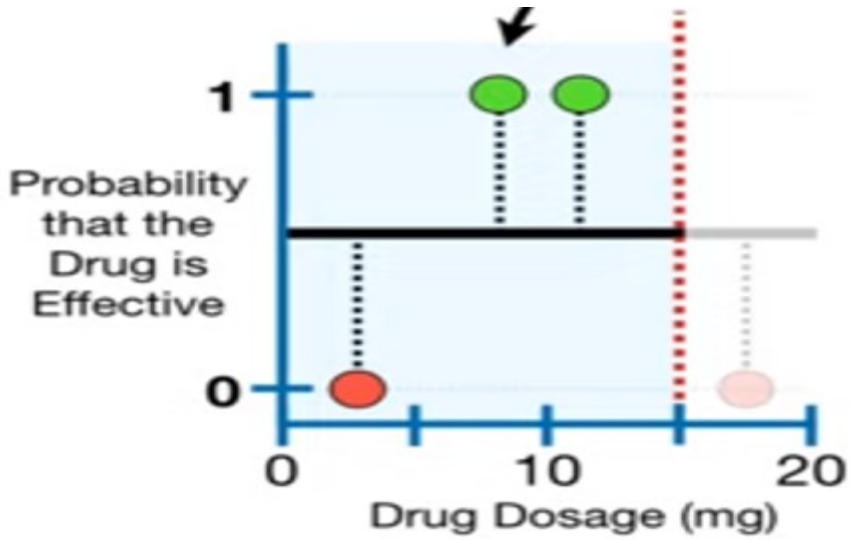
$$\frac{\left(\sum \text{Residual}_i\right)^2}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right] + \lambda}$$

Similarity=0

- Choix des variables :
  On va devoir essayer tout les cas possible et calculer le gain et choisir celui le moins elevee

  1)- dosage <15:
  on a choisit 15 car c'est la Moyenne entre les deux point (10 et 20)

  A gauche :

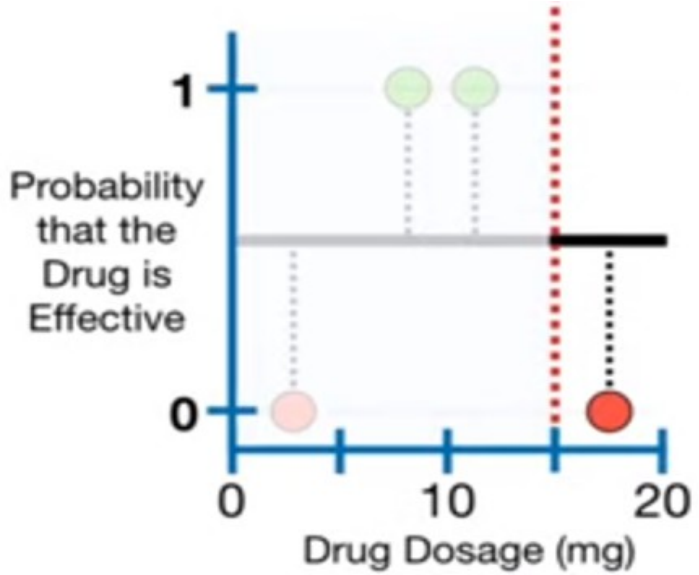


  calculee si

$$\textbf{similarity} = \frac{(-0.5 + 0.5 + 0.5)^2}{(0.5 \times (1-0.5)) + (0.5 \times (1-0.5)) + (0.5 \times (1-0.5)) + 0} = \mathbf{0.33}$$

  avec $\lambda = 0$

A droite :



calculer score de similaritee de la brache a droite :

$$\textbf{similarity} = \frac{(-0.5)^2}{0.5 \times (1 - 0.5) + \lambda} = 1$$

- Calculer le gain :

$$\text{Gain} = \text{Left}_{\text{Similarity}} + \text{Right}_{\text{Similarity}} - \text{Root}_{\text{Similarity}}$$

$$\text{Gain} = 0.33 + 1 - 0 = 1.33$$

Donc , Quand on a devisee les observation par rapport au dosage<15
le Gain = 1.33

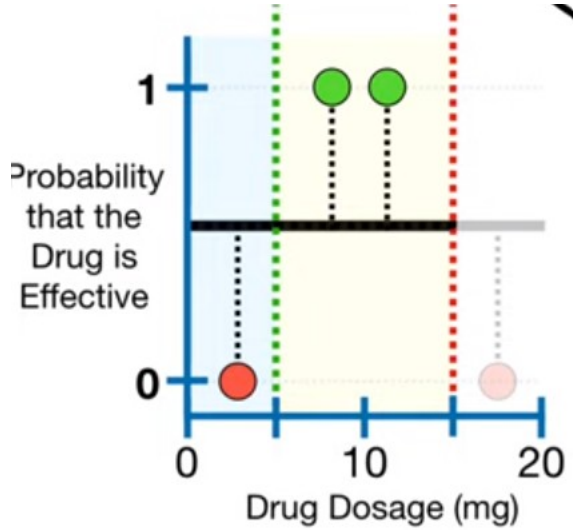
On repete la procedure , on va essayer de splitee par rapport a d'autres points et calculer les scores de similaritee

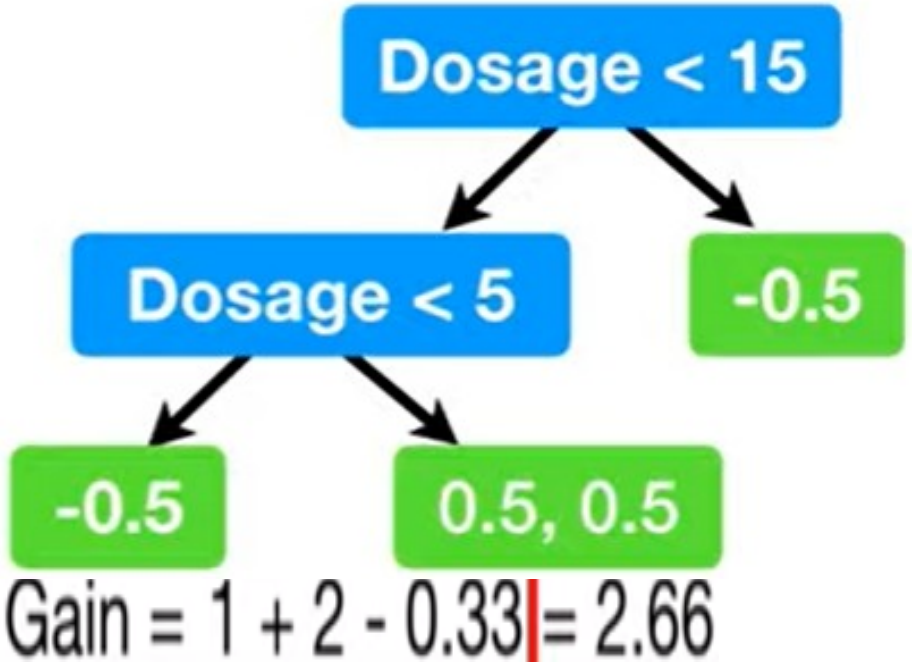
#on gardera l'arbre avec le gain le plus elevee

On va s’occuper de la branche :

De la meme manière on suppose une manière de split , et choisi celle avec le plus grand gain

2)dosage<5:





➢ L’elagage de l’arbre :

on va definer un hyper parameter γ
pour le comparer entre le Gain

l’elagage aura lieu : Gain < γ
en d’autre terme si Gain- γ <0 on va coupper l’arbre
exemple
prenant γ=3
2,66-3<0 alors on va couper la branche
Indirectement : en augementant λ dans le score de similaritee on obtien un gain plus elevee ,ce qui rond l’arbre plus secceptible aetre laggee

➢ output in each leaf :

Une fois tt les arbres sont construites , on va calculer la valeur de y_pred dans chaque branche

$$\frac{(\sum \text{Residual}_i)}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

dosage<5

output de la brache a gauche

$$\frac{-0.5}{0.5 \times (1 - 0.5) + \lambda} = -2$$

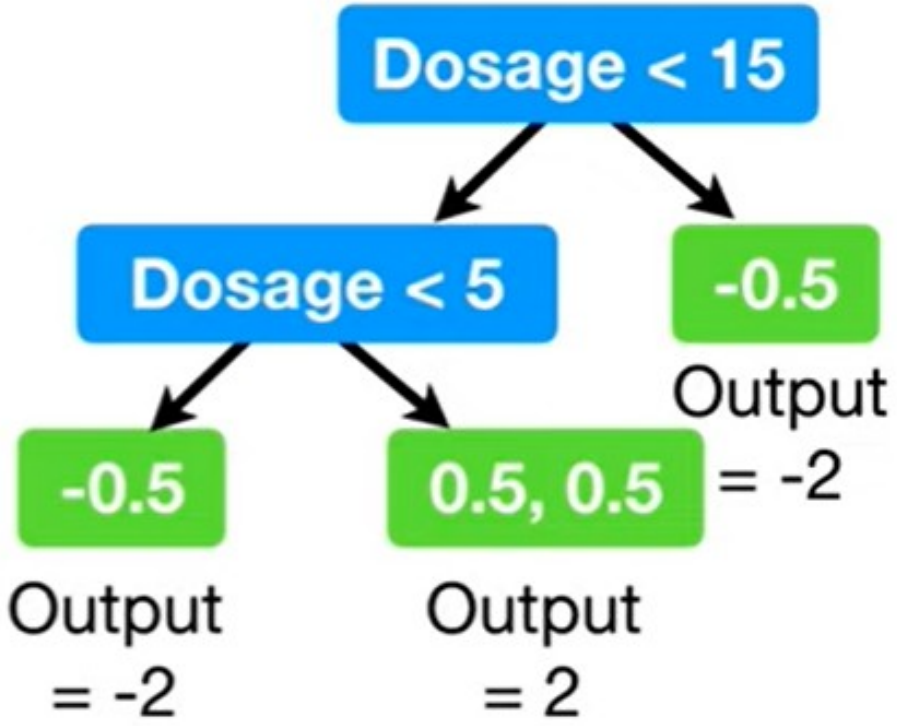
output de la brache a droite

$$\frac{0.5 + 0.5}{0.5 \times (1 - 0.5) + 0.5 \times (1 - 0.5) + \lambda} = 2$$

Pour la brahe dosage <15:

$$\frac{-0.5}{0.5 \times (1 - 0.5) + 0} = -2$$

maintenant qu'on a construit l'arbre :



➢ Les Nouvelles prediction :

une fois l'arbre est construit , on utilise les prediction de cet arbre pour calculer les Nouvelles predictions du models , en utilisant un taux d'apprentissage

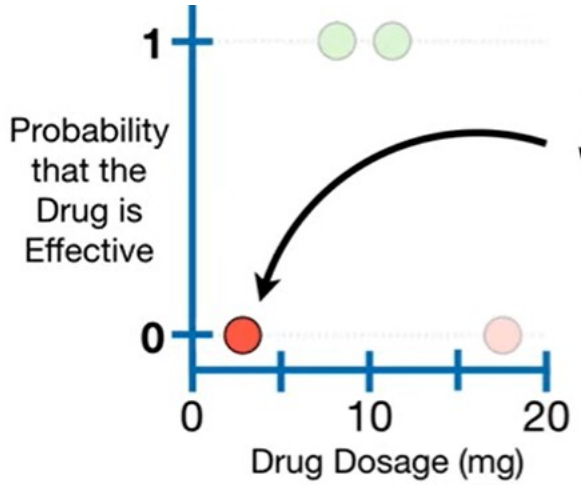
$$\log(\text{odds})_{\text{initial}} + \text{Learning rate} * \text{output(tree)}$$

Comme pour n'import quell cas de classification , on doit convertir les proba en log(odds)

on remplace $\log(\frac{p}{1-p}) = \log(\text{odds})$

- la premiere valeur initialiser : $\log(\frac{0.5}{1-0.5}) = 0$

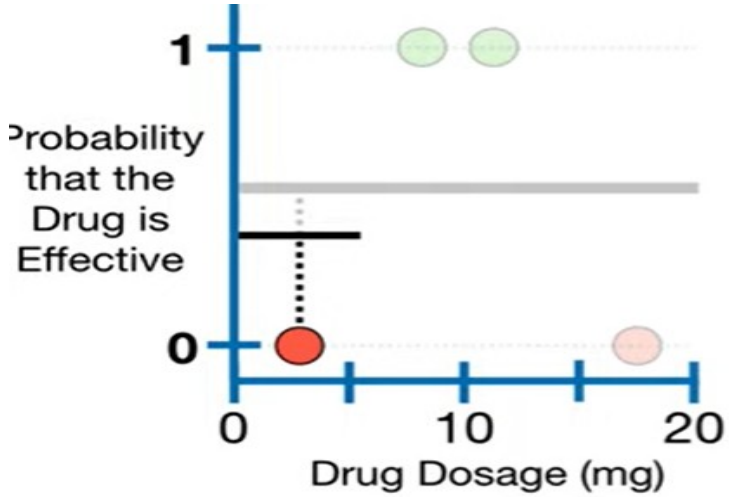
-Taux d’apprentissage =0.3
-La nouvelle prediction pour dosage=2
dosage=2<5 (-2)
 log(odds)prediction=0+0.3*-2= -0.6



pour convertir log(odds) en proba , on utilise la fonction logistique (sigmoid)

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} \quad \text{Probability} = \frac{e^{-0.6}}{1 + e^{-0.6}} = 0.35$$

ps: la prediction initial etait a 0.5
tandis que maintenant est a 0.3 elle a diminue
le residu a diminue



Nouvelle prediction pour dosage=8

D’apres l’arbre , dosage=8 > 5 (2)
log(odds)prediction=0+0.3*2= 6
convertir en proba (en introduisant la function logostoque)
proba=0.65

on remarque que les residues des Nouvelles prediction diminue

➢ Construction d'un novel arbre :

on peut construire une deuxieme arbre en se basant sur les residu de la premiere
ou le dominateur est different et n'est plus le meme

$$\text{Similarity Score} = \frac{(\sum \text{Residual}_i)^2}{(0.35 \times (1-0.35)) + (0.65 \times (1-0.65)) + (0.65 \times (1-0.65)) + (0.35 \times (1-0.35)) + \lambda}$$

Une fois touts les arbres sont construites , on va combiner leur prédiction pondérée pou trouver la prédiction finale du modèle