Revue-IRS

Revue Internationale de la Recherche Scientifique
(Revue-IRS)
ISSN: 2958-8413
Vol. 4, No. 5, Septembre 2026

MODÉLISATION ET OPTIMISATION D'UN MODÈLE HYBRIDE D'APPRENTISSAGE AUTOMATIQUE BASÉ SUR LA COMBINAISON DES CLASSIFIEURS VIA BASSAMOD. (Application à analyse des Données Épidémiologique)

Par : Chef de Travaux **BASA KUTUPU Faustin** (UNIKAN),
Professeur Ordinaire : TSHIBAKA KATUMONGANI Emmanuel (UPN)
Professeur Ordinaire : NTUMBA BADIBANGA Simon (UNIKIN)
Professeur : BATUBENGA MUAMBA NZAMBI Jean de Dieu (UNIKIN)
Assistante: TUDIMUENE MUNDA BOMBESHA Marie- Jeanne (UPKAN)
Assistant : BANANTAMBA KALUNDE Trésor (UNIKAN)

Digital Object Identifier (DOI): https://doi.org/10.5281/zenodo.**22706693**

Abstract: Ce travail porte sur la modélisation et l'optimisation de Bassamod, un modèle hybride d'apprentissage automatique appliqué à des données épidémiologiques. Il part du constat que les classificateurs individuels présentent certaines limites lorsqu'ils sont confrontés à des données complexes, bruitées, hétérogènes et non linéaires. La recherche vise donc à concevoir un modèle capable d'améliorer la précision, la robustesse et la capacité de généralisation des prédictions.

L'hypothèse principale stipule que la combinaison hiérarchique et dynamique de plusieurs classificateurs complémentaires permet d'améliorer les performances globales. Le modèle repose sur une architecture série-parallèle-série. La première couche utilise un arbre de décision afin d'orienter et de structurer les données. La deuxième couche met en parallèle plusieurs classificateurs hétérogènes : la régression logistique, les K plus proches voisins (KNN), la forêt aléatoire (Random Forest) et le Gradient Boosting. La troisième couche repose sur une fusion adaptative assurée par un méta-classificateur fondé sur le Gradient Boosting.

Cette structure permet d'exploiter la diversité, la complémentarité et l'adaptabilité des différents modèles. Sur le plan théorique, Bassamod s'inspire des approches de bagging, de boosting et de stacking, tout en les enrichissant par une orientation initiale des données. Son objectif est de réduire le biais, la variance ainsi que les erreurs de classification.

L'implémentation a été réalisée en Python, à l'aide d'outils tels que Pandas, NumPy et Scikit-learn. Les données ont été prétraitées, normalisées, puis divisées en ensembles d'entraînement et de test. Les résultats montrent que le KNN atteint une précision de 89,4 %, tandis que l'arbre de décision et la forêt aléatoire atteignent une précision de 100 % sur les données évaluées. Le modèle hybride Bassamod obtient une précision de 1,00, ainsi qu'une précision (precision), un rappel (recall) et un score F1 également égaux à 1,00. Les 337 observations de la classe 0 et les 163 observations de la classe 1 ont toutes été correctement classifiées. Une nouvelle observation a également été correctement prédite, confirmant ainsi la capacité opérationnelle du modèle.

Ces résultats valident les hypothèses initiales et démontrent l'intérêt d'une combinaison intelligente de plusieurs classificateurs. Toutefois, une performance parfaite doit être interprétée avec prudence, car elle peut être révélatrice d'un surapprentissage (overfitting).

En conclusion, Bassamod apparaît comme une solution innovante, robuste et prometteuse pour l'analyse prédictive des données épidémiologiques.

Keywords: Modélisation; Optimisation; Apprentissage automatique; Combinaison; Claissifieur; Bassamod…

Introduction

Au cours des dernières décennies, le développement des technologies numériques, de la science des données et de l'intelligence artificielle a profondément transformé les méthodes d'analyse et de prise de décision dans de nombreux domaines scientifiques. Parmi ces domaines, l'épidémiologie occupe une place particulièrement importante, dans la mesure où elle vise à étudier la fréquence, la distribution et les déterminants des phénomènes de santé au sein des populations. Dans un contexte marqué par l'augmentation des maladies

émergentes, la nécessité d'une surveillance sanitaire continue et la production croissante de données médicales, cliniques, démographiques et environnementales, la maîtrise de ces informations devient un enjeu scientifique et stratégique majeur.

Toutefois, l'exploitation efficace des données épidémiologiques demeure une problématique complexe. En effet, ces données présentent généralement plusieurs caractéristiques qui compliquent leur traitement : hétérogénéité des variables, présence de bruit, déséquilibre entre classes, relations non linéaires, dépendances multiples et évolution temporelle. Dans de telles conditions, les approches classiques d'analyse statistique et même certains algorithmes d'apprentissage automatique utilisés isolément montrent souvent des limites en termes de précision, de robustesse et de capacité de généralisation. Or, dans le domaine de la santé, où la qualité des prédictions peut avoir des conséquences importantes sur la prévention, le diagnostic et l'orientation des politiques sanitaires, l'amélioration des performances des modèles de classification constitue une exigence fondamentale.

C'est dans cette perspective que s'inscrit la présente recherche intitulée : « Modélisation et Optimisation d'un Modèle Hybride d'Apprentissage Automatique basé sur la Combinaison des Classifieurs via Bassamod : Application à l'Analyse des Données Épidémiologiques ». Ce travail part du principe qu'aucun classifieur pris isolément ne peut garantir, à lui seul, une performance optimale dans tous les contextes de données complexes. Certains modèles sont performants dans la capture de relations linéaires, d'autres excellent dans la prise en compte de structures non linéaires, tandis que d'autres encore se distinguent par leur robustesse ou leur capacité d'adaptation. Dès lors, il devient pertinent d'envisager une approche de combinaison de classifieurs, afin d'exploiter leur complémentarité et d'aboutir à une décision plus fiable, plus stable et plus performante.

La problématique centrale de cette étude peut ainsi être formulée de la manière suivante : comment concevoir un modèle hybride d'apprentissage automatique capable d'optimiser les performances de classification et d'améliorer la précision, la robustesse ainsi que la capacité de généralisation des systèmes appliqués aux données épidémiologiques ? Cette interrogation constitue le point de départ de notre réflexion. Elle traduit la volonté de dépasser les insuffisances des méthodes classiques de classification en proposant une architecture plus structurée, plus souple et mieux adaptée à la complexité des données sanitaires.

Pour répondre à cette question, nous avons formulé l'hypothèse selon laquelle la combinaison hiérarchique et dynamique de plusieurs classifieurs complémentaires, intégrés dans une architecture hybride multi-niveaux, permettrait d'améliorer significativement la performance prédictive d'un système d'apprentissage automatique. Plus spécifiquement, nous considérons que l'organisation série–parallèle–série, associée à une fusion adaptative des prédictions, favoriserait une meilleure exploitation des informations issues des différents modèles, tout en réduisant les erreurs liées au compromis biais–variance. Cette hypothèse repose sur l'idée que la diversité, la complémentarité et l'adaptativité constituent des leviers majeurs d'optimisation dans les systèmes d'apprentissage automatique.

L'objectif général de cette recherche est de concevoir, formaliser, implémenter et optimiser un modèle hybride de combinaison de classifieurs, dénommé Bassamod, en vue de son application à l'analyse des données épidémiologiques. De manière spécifique, il s'agit de présenter la motivation du modèle, d'en proposer une définition conceptuelle, formelle et mathématique, d'en décrire l'architecture fonctionnelle, d'en établir la modélisation théorique et algorithmique, puis d'en évaluer expérimentalement les performances à travers des indicateurs de classification appropriés.

L'originalité de cette recherche réside dans la proposition d'une architecture hybride de combinaison de classifieurs fondée sur trois niveaux complémentaires. Le premier niveau assure une orientation initiale des données à l'aide d'un arbre de décision, afin de structurer le problème de classification. Le deuxième niveau met en œuvre plusieurs classifieurs hétérogènes en parallèle, notamment la régression logistique, le KNN, la forêt aléatoire et le Gradient Boosting, dans le but de capter différentes formes de relations présentes dans les données. Enfin, le troisième niveau repose sur un méta-classifieur adaptatif chargé d'assurer la fusion optimisée des prédictions produites par les modèles de base. Cette organisation confère au modèle Bassamod une dimension innovante, à la croisée du stacking, du boosting et de l'apprentissage adaptatif.

Sur le plan scientifique, cette étude présente un triple intérêt. D'abord, elle contribue à l'enrichissement théorique des approches hybrides en apprentissage automatique, en proposant une nouvelle structure de combinaison des classifieurs. Ensuite, elle apporte une contribution méthodologique, en mettant en évidence une stratégie de fusion hiérarchique et adaptative mieux adaptée aux données complexes. Enfin, elle revêt un intérêt pratique important, en montrant comment les techniques modernes d'intelligence artificielle peuvent

être mobilisées pour renforcer l'analyse prédictive dans le domaine épidémiologique, avec des perspectives en matière d'aide à la décision et de surveillance sanitaire.

La démarche méthodologique adoptée dans ce travail repose à la fois sur une approche théorique et expérimentale. Elle mobilise les fondements de l'apprentissage automatique, de la théorie des ensembles de classifieurs, du méta-apprentissage et de l'optimisation des systèmes prédictifs. L'implémentation pratique du modèle à l'aide d'outils tels que Python, Pandas, NumPy et Scikit-learn permet de confronter les hypothèses théoriques à des résultats concrets, afin d'apprécier la pertinence réelle du modèle proposé.

En définitive, ce travail entend démontrer que l'amélioration de la performance d'un système de classification ne dépend pas uniquement de la qualité d'un algorithme isolé, mais de la capacité à organiser, articuler et fusionner intelligemment plusieurs modèles complémentaires au sein d'une architecture cohérente. À travers Bassamod, notre ambition est de proposer une solution robuste, flexible et performante, susceptible d'ouvrir de nouvelles perspectives dans l'exploitation de l'intelligence artificielle appliquée aux données épidémiologiques.

CHAPITRE I : MODELISATION DU MODELE BASSAMOD

Malgré les avancées significatives de ces approches, certaines limites subsistent, notamment en termes de performance, de robustesse et d'adaptabilité face à des données complexes telles que les données épidémiologiques, souvent caractérisées par le bruit, le déséquilibre et la non-linéarité. Ces insuffisances justifient la nécessité de proposer des modèles plus performants, capables d'exploiter efficacement la complémentarité des classifieurs.

C'est dans ce contexte que s'inscrit notre contribution, à travers la proposition d'un modèle hybride de combinaison de classifieurs, dénommé BassaMode. Ce modèle vise à améliorer la qualité de la classification en s'appuyant sur une architecture multi-niveaux intégrant plusieurs classifieurs et une stratégie de fusion optimisée.

L'objectif de cette section est donc de présenter la modélisation du modèle BassaMode. À cet effet, nous décrirons successivement la motivation du modèle, sa définition formelle, son architecture, sa modélisation mathématique, ainsi que son algorithme de fonctionnement. Une attention particulière sera également accordée aux mécanismes d'optimisation intégrés afin d'améliorer les performances globales du système.

Section1 : Nouvelles Approche *de Combinaison de Classifieurs*

Le développement du modèle BassaMode s'inscrit dans la volonté d'améliorer les performances des systèmes d'apprentissage automatique, notamment dans des contextes complexes tels que l'analyse des données épidémiologiques. Malgré les progrès réalisés dans le domaine de la classification, plusieurs limites persistent et justifient la nécessité de proposer une nouvelle approche.

1. Limites des classifieurs individuels : Les classifieurs traditionnels tels que la régression logistique, les arbres de décision, les machines à vecteurs de support (SVM) ou encore les méthodes basées sur les plus proches voisins (KNN), présentent des performances variables selon la nature des données.

Cependant, aucun classifieur ne peut, à lui seul, garantir une performance optimale dans tous les contextes. En particulier :

- certains modèles sont sensibles au bruit ;
- d'autres présentent un fort biais ou une forte variance ;
- certains échouent à capturer des relations complexes non linéaires.

Cette limitation constitue un premier facteur motivant la combinaison de plusieurs classifieurs.

2. Complexité des données épidémiologiques : Les données épidémiologiques possèdent des caractéristiques spécifiques :

- présence de bruit et d'incertitude ;
- déséquilibre entre classes (cas rares vs fréquents) ;
- hétérogénéité des variables (cliniques, démographiques, environnementales) ;
- évolution dynamique dans le temps.

Ces propriétés rendent difficile l'utilisation efficace d'un modèle unique. Il devient donc nécessaire de concevoir des modèles capables de s'adapter à cette complexité.

3. Limites des méthodes classiques de combinaison : Les techniques existantes telles que : le bagging, le boosting et le stacking. Ont permis d'améliorer les performances des systèmes de classification.

Toutefois, elles présentent encore certaines limites :

- manque de flexibilité dans certaines architectures ;
- absence d'adaptation dynamique aux données ;
- difficulté à exploiter pleinement la complémentarité entre classifieurs ;
- performances variables selon les contextes d'application.

Ces insuffisances ouvrent la voie à de nouvelles approches hybrides.

4. Besoin d'un modèle hybride optimisé

Face à ces limites, il apparaît nécessaire de concevoir un modèle capable de :

- combiner efficacement plusieurs classifieurs;
- exploiter leurs complémentarités ;
- améliorer la robustesse du système ;
- réduire les erreurs de classification ;
- s'adapter aux données complexes.

5. Proposition du modèle BassaMode

C'est dans ce contexte que nous proposons le modèle BassaMode, un modèle hybride de combinaison de classifieurs basé sur :

- une architecture multi-niveaux ;
- une organisation série–parallèle–série ;
- une stratégie de fusion optimisée ;
- un mécanisme d'amélioration des performances.

Le modèle BassaMode vise à :

- améliorer la précision de classification ; renforcer la stabilité des prédictions ;
- réduire l'impact du bruit ;
- offrir une meilleure capacité de généralisation.

6. Positionnement du modèle Bassamod : Le modèle BassaMode se positionne comme :

- une extension des méthodes d'ensemble existantes ; une approche hybride innovante ;
- une solution adaptée aux données épidémiologiques.

Il constitue ainsi une contribution à la fois :

- théorique (nouvelle architecture), méthodologique (nouvelle stratégie de combinaison), et
- appliquée à la santé publique.

La motivation du modèle BassaMode repose sur la nécessité de dépasser les limites des classifieurs individuels et des méthodes classiques de combinaison, en proposant une approche hybride optimisée capable de traiter efficacement des données complexes telles que les données épidémiologiques.

I.1.2. Définition

1. Définition conceptuelle

Le modèle BassaMode est un modèle hybride de combinaison de classifieurs conçu pour améliorer les performances des systèmes d'apprentissage automatique. Il repose sur l'intégration de plusieurs classifieurs hétérogènes organisés selon une architecture multi-niveaux, combinés à l'aide d'une fonction de fusion optimisée.

2. Définition mathématique

Soit :

- $X \subseteq R^d$: l'espace des caractéristiques (features),
- Y : l'ensemble des classes,
- $D=\{(xi,yi)\}_{i=1}^{n}$: l'ensemble de données d'apprentissage,
- $fj:X \rightarrow Y$, j =1, 2,...,k : un ensemble de k classifieurs de base,
- $w_j \in R$: le poids associé à chaque classifieur,
- F : une fonction de combinaison (fusion).

3. Définition formelle

Le modèle BassaMode est une fonction composée définie par : $BassaMode(x)=F(w_1f_1(x), w_2f_2(x), \ldots, w_kf_k(x))$ où :

- fj(x) : représente la sortie du classifieur j,
- wj : pondère l'importance de chaque classifieur,
- F est une fonction de décision (vote, moyenne, ou méta-classifieur).

3. Cas probabiliste (plus avancé)

Si les classifieurs produisent des probabilités : $f_j(x)=P_j(y|x)$; Alors le modèle devient : $BassaMode(x)=argmax_{y \in Y} \sum_{j=1}^{k} wjPj(y \mid x)$, Cela correspond à une fusion pondérée probabiliste.

4. Intégration du méta-classifieur (Stacking) : Dans le cas d'un modèle plus avancé (stacking), on définit : $Z(x)=(f_1(x),f_2(x),...,f_k(x))$ et : $BassaMode(x)=g(Z(x))$ où : g est un méta-classifieur (ex : régression logistique, XGBoost).

5. Définition fonctionnelle globale : Le modèle BassaMode peut être vu comme une composition de fonctions : $BassaMode=g \circ F \circ (f1,f2,...,fk)$

6. Interprétation : Cette définition signifie que :

- chaque classifieur apprend une représentation des données ;
- leurs sorties sont combinées ;
- une décision finale est produite de manière optimisée.

7. Propriétés du modèle

Le modèle BassaMode possède les propriétés suivantes :

- Modularité : ajout ou suppression de classifieurs possible ;
- Robustesse : réduction de l'impact du bruit ;
- Adaptabilité : ajustement des poids et du méta-modèle ;
- Généralisation : meilleure performance sur données inconnues.

Ainsi, le modèle BassaMode se définit formellement comme une fonction de combinaison pondérée ou apprise de plusieurs classifieurs, visant à produire une prédiction plus fiable et plus précise que celle obtenue par un classifieur unique.

I.1.3. Architecture du Modèle Bassamod

Le modèle BassaMode repose sur une architecture hybride en trois couches, combinant une phase d'orientation, une phase d'apprentissage parallèle et une phase de fusion adaptative.

Cette organisation permet d'exploiter efficacement la complémentarité des classifieurs tout en optimisant la décision finale.

1. Première couche : Orientation initiale par arbre de décision

La première couche du modèle est constituée d'un arbre de décision.

Rôle **:**

- effectuer une segmentation initiale des données ;
- orienter les observations vers des régions décisionnelles pertinentes ;
- simplifier la structure du problème.

Intérêt :

- réduction de la complexité ;
- meilleure séparation des classes ;
- amélioration des performances des classifieurs en aval.

Cette étape joue un rôle de filtrage intelligent des données.

2. Deuxième couche : Classifieurs hétérogènes en parallèle

La deuxième couche repose sur un ensemble de classifieurs fonctionnant en parallèle :

- Régression Logistique
- K-Nearest Neighbors (KNN)
- Random Forest
- Gradient Boosting

Rôle **:** exploiter différentes représentations des données ; capturer des relations linéaires et non linéaires ;

- produire des prédictions diversifiées.

Spécificité de chaque classifieur:

- Régression Logistique : modélisation linéaire, interprétable
- KNN : apprentissage basé sur la proximité
- Random Forest : robustesse et réduction de variance
- Gradient Boosting : correction des erreurs et forte performance

Cette diversité garantit une complémentarité informationnelle.

3. Troisième couche : Fusion adaptative (ADE)

La troisième couche correspond à un méta-classifieur de fusion adaptative (ADE) basé sur Gradient Boosting.

Rôle : combiner les sorties des classifieurs de base ; apprendre dynamiquement les meilleures combinaisons ;

- corriger les erreurs des modèles précédents.

Principe :

- les prédictions des classifieurs deviennent les entrées du méta-modèle ;
- le Gradient Boosting ajuste les poids de manière itérative ;
- la décision finale est optimisée.

Cette étape constitue le cœur de l'optimisation du modèle.

III.1.4.1. Représentation fonctionnelle

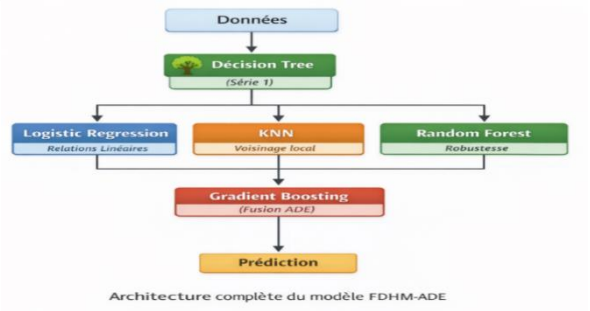

Figure I.1.1.2. Bassamod fonctionnel

I.1.4.2. Avantages de l'architecture

- meilleure précision globale
- réduction des erreurs individuelles
- robustesse face au bruit
- meilleure généralisation
- adaptation aux données épidémiologiques

I.1.4.3. Originalité du modèle Bassamod

Notre architecture est originale car elle :

- introduit une phase d'orientation (arbre de décision) avant la combinaison
- combine plusieurs classifieurs hétérogènes ;
- utilise une fusion adaptative basée sur Gradient Boosting (ADE) ;
- propose une structure série–parallèle–série optimisée.

L'architecture du modèle BassaMode repose sur une combinaison structurée de techniques d'apprentissage automatique permettant d'améliorer significativement la qualité de la classification. L'intégration d'une phase d'orientation, suivie d'un apprentissage parallèle et d'une fusion adaptative, constitue une approche innovante adaptée aux données complexes

Section 2. Analyse comparative de trois approches

Les trois méthodes relèvent des techniques d'apprentissage en ensemble, mais elles diffèrent fortement dans leur logique.

Selon le mode de construction

1. Bagging : construction parallèle de modèles homogènes ;
2. Boosting : construction séquentielle, chaque modèle corrigeant le précédent ;
3. Stacking : construction parallèle de modèles souvent hétérogènes, suivie d'un méta-apprentissage.

Fusion Générale: $H(x)=\Phi(h_1(x),h_2(x),...,h_n(x))$, où Φ est une fonction de fusion : Moyenne; vote majoritaire; ponderation; et méta-apprentissage

1.2 Objectif principal

- Bagging : réduire la variance ;
- Boosting : réduire le biais et améliorer la précision ;
- Stacking : apprendre une combinaison optimale de plusieurs prédictions.

1.3 Type de diversités

- Bagging : diversité créée par les échantillons bootstrap ;
- Boosting : diversité créée par la focalisation successive sur les erreurs ;
- Stacking : diversité créée par l'usage de modèles différents.

C'est – à- dire selon les Diversité des classifieurs : Diversity=1−Corr(h_i,h_j)

1.4 Fusion

- Bagging : vote ou moyenne simple ;
- Boosting : somme pondérée séquentielle ;
- Stacking : méta-classifieur.

D'après la Pondération adaptative **: $H(x)= \sum_{i=1}^{N} wi(x) \cdot hi(x)$** , On constate que dans notre modèle Bassamod, les poids sont dynamiques

Comportement Global (BIAS-VARIANCE)

$Erreur_{totale}$ =Bias2+Variance+BruitErreur

- Bagging → ↓ Variance
- Boosting → ↓ Bias
- Stacking → équilibre global

Analyse critique de la comparaison: L'analyse comparative montre que le Bagging est particulièrement utile lorsque le modèle de base est instable et que la priorité est la réduction de la variance. Le Boosting, quant à lui, est plus adapté lorsque l'on cherche à corriger progressivement les erreurs et à construire un modèle très performant, bien qu'il puisse être sensible au bruit. Le Stacking apparaît comme une approche plus souple et plus riche, dans la mesure où il combine plusieurs modèles différents à travers un méta-classifieur.

Cependant, aucune de ces méthodes ne répond parfaitement à toutes les exigences des données épidémiologiques complexes. Le Bagging reste souvent trop homogène. Le Boosting est puissant, mais sa logique séquentielle ne permet pas toujours d'exploiter pleinement des expertises multiples. Le Stacking est plus avancé, mais il ne prévoit pas nécessairement une orientation initiale structurée des données avant la phase de combinaison. C'est précisément dans cet espace que s'inscrit le modèle BassaMode, qui peut être vu comme une extension structurée du Stacking, enrichie par :

- une première couche d'orientation par arbre de décision ;

- une deuxième couche de classifieurs hétérogènes en parallèle ;
- une troisième couche de fusion adaptative fondée sur Gradient Boosting.

III.3.4 : Position du modèle BassaMode

Le modèle BassaMode ne se confond pas avec les approches classiques, même s'il emprunte certains de leurs principes.

- Il se distingue du Bagging parce qu'il ne repose pas sur des échantillons bootstrap ni sur une simple agrégation homogène.
- Il se distingue du Boosting parce qu'il n'organise pas ses classifieurs de base selon une chaîne séquentielle de correction.
- Il se rapproche du Stacking par l'usage d'un méta-classifieur, mais il s'en différencie par l'ajout d'une couche d'orientation initiale et par une fusion adaptative plus structurée.

Ainsi, BassaMode peut être présenté comme une architecture hybride inspirée du stacking, enrichie par une phase de guidage décisionnel et une fusion adaptative optimisée.

Section 3 : Construction du Modèle Bassamod

1 Principe général de la modélisation

Le modèle BassaMode est modélisé comme une composition de fonctions d'apprentissage organisées en trois niveaux :

1. Orientation des données (Arbre de décision)
2. Apprentissage parallèle (classifieurs hétérogènes)
3. Fusion adaptative (méta-classifieur ADE)

Ainsi, le modèle global est une fonction composée :

$$\text{BassaMode} = g \circ H \circ T$$

où :

- T: fonction d'orientation (arbre de décision) ;
- H : ensemble des classifieurs parallèles;
- g : fonction de fusion adaptative (ADE)

1. Modélisation de la première couche : Orientation

Soit :

- $x \in R^d$ une observation;
- T(x) la fonction arbre de décision.

Alors : $z=T(x)$

L'arbre de décision transforme l'espace initial en sous-espaces décisionnels.

2. Modélisation de la deuxième couche : Classifieurs parallèles

Soit un ensemble de classifieurs : $H=\{f_1,f_2,f_3,f_4\}$

avec :

- f_1: Régression logistique
- f_2 : KNN
- f_3 : Random Forest
- f_4 : Gradient Boosting

Chaque classifieur produit : $y_i=f_i(z)$

On obtient alors un vecteur de sortie : $Y=(y_1,y_2,y_3,y_4)$.

3. Modélisation de la troisième couche : Fusion adaptative (ADE)

$G(x)=Y(h_1(x),h_{2(}x),h_{3(}x),h_4(x))$

où G est un modèle Gradient Boosting (méta-classifieur)

Le méta-classifieur (Gradient Boosting) est défini comme : $\hat{y}=g(Y)$

où :

- g apprend à combiner les sorties des classifieurs ;
- Y est le vecteur des prédictions.

Forme explicite (Gradient Boosting)

Le modèle peut être écrit : $g(Y)= \sum_{m=1}^{M} M\ Ymhm(Y)$

où :

- h_m : arbres faibles (weak learners);
- γ_m: poids appris;
- M: nombre d'itérations

4. Modèle global du BassaMode

En combinant les trois niveaux : $\hat{y}=g(f_1(T(x)),f_2(T(x)),f_3(T(x)),f_4(T(x)))$

Ou plus détaillé :

$\mathbf{Y(x)= \sum_{m=1}^{M} \gamma m \cdot gm(f1(T(x)), f2(T(x)), f3(T(x)), f4(T(x))}$, C'est l'expression mathématique complète du modèle.

5. Modélisation probabiliste (cas avancé)

Si chaque classifieur fournit une probabilité : $\mathbf{f_i(x)=P_i(y|x)}$

alors **:** $\mathbf{\hat{y}=argmax_y\ g(P_1(y|x),P_2(y|x),...,P_4(y|x))}$.

6. Modélisation algorithmique du Bassamode

Entrée : Dataset D

1. Prétraiter les données
2. Appliquer l'arbre de décision : z=T(x)
3. Pour chaque classifieur fi :
   - entraîner fi
   - calculer yi=fi(z)
4. Construire le vecteur Y
5. Entraîner le méta-classifieur g
6. Produire la prédiction finale :ŷ=g(Y)

Sortie : Classe prédite

7. Interprétation scientifique

Notre modèle peut être vu comme :

- une décomposition du problème (arbre)
- une extraction multi-vues (classifieurs parallèles)
- une fusion intelligente (ADE)

8. Nature du modèle

Le BassaMode est :

- un modèle hybride
- un modèle d'ensemble avancé
- un stacking amélioré avec prétraitement structurel
- un modèle optimisé pour données complexes

9. Contribution scientifique de la modélisation

Notre modélisation apporte :

- une nouvelle structure mathématique
- une intégration hiérarchique des classifieurs
- une fusion adaptative optimisée
- une meilleure formalisation des systèmes hybrides

Bassamod = Bagging + Boosting + Stacking + Adaptativite´

Optimisation : minL(Y(x),y)

Robustesse : Y(x)≈E[fi(x)]

La modélisation du modèle BassaMode met en évidence une architecture composée de fonctions d'apprentissage successives permettant de transformer, analyser et combiner les données de manière optimisée. Cette approche constitue une amélioration des méthodes classiques de combinaison de classifieurs.

Théorie du Modèle BassaMode

La théorie de Bassamod repose sur une approche hybride d'apprentissage automatique fondée sur la combinaison adaptative multi-niveaux de classifieurs hétérogènes, visant à minimiser simultanément le biais et la variance tout en optimisant la performance prédictive globale.

Formulation Mathématique Globale

Le modèle Bassamod est défini comme : $\mathbf{H(x)=G(h_1(x),h_2(x),...,h_n(x))}$

où sous forme pondérée adaptative **:** $\mathbf{H(x)= \sum_{i=1}^{n} wi(x) \cdot hi(x)}$

où :

- hi(x)= classifieurs de base (LR, KNN, RF, GB) ;
- wi(x) = poids adaptatifs dépendant des données ;
- G = méta-classifieur (fusion intelligente).

Principe Fondamental

Bassamod repose sur 3 idées clés :

1. Diversité des modèles : hi≠hj
2. Complémentarité : H(x)>hi(x)
3. Adaptativité : w_i=f(x, $performance_i$)

Structure Théorique

Bassamod suit une architecture :Série→Parallèle→Série

- Série 1 : orientation (arbre de décision)
- Parallèle : classifieurs hétérogènes
- Série 2 : fusion adaptative (ADE)

Objectif Théorique : Min $\ell(H(x),y)$
avec : Erreur = $Bias^2$ + Variance + Bruit
Bassamod cherche : ↓Bias+↓Variance
Bassamod est une théorie d'optimisation de l'apprentissage par fusion adaptative de modèles diversifiés dans une architecture hybride multi-niveaux.

Complexité globale du modèle BassaMod

Bassamod est un modèle multi-niveaux (série–parallèle–série), donc sa complexité est la somme des trois couches.

1 Notations

- N : nombre d'exemples
- d : nombre de variables
- k : voisins (KNN)
- T: nombre d'arbres (Random Forest)
- M : itérations (Boosting)

Complexité globale : En combinant tout : O(NdlogN+TNdlogN+MNd)
Simplification : O((TlogN+M)·N·d)

3. **Performance du Modèle Bassamod**

Expression de performance

$$\text{Accuracy}=\frac{\text{Nombre de bonnes pre´dictions}}{\text{Nombre total}}$$

$$\text{F1-Score} = 2* \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$$

2 Performance théorique

Grâce à la combinaison : ErreurBassamod < min(Erreurhi)
Ou encore : $H(x)\approx E[hi(x)]$

3 Gains obtenus

- Réduction de variance (Random Forest)

Var↓

- Réduction du biais (Boosting)

Bias↓

- Optimisation globale (Stacking)

Erreurglobale↓

4 Robustesse : Robustesse $\propto$ Diversite´(hi)

Plus les modèles sont différents → plus le modèle est stable

5 Généralisation : Remp→Rréel

Bassamod améliore la capacité à généraliser

Le modèle Bassamod présente une complexité algorithmique relativement élevée en raison de sa structure hybride multi-niveaux. Toutefois, cette complexité est compensée par une amélioration significative des performances prédictives, résultant de la réduction simultanée du biais et de la variance, ainsi que de l'exploitation de la diversité des classifieurs et d'une fusion adaptative intelligente.

III.5.2. Algorithmique de Fonctionnement du Modèle BassaMod

Bassamod suit une architecture :
Entre´e→Orientation→Apprentissage parallèle→Fusion adaptative→Sortie

Algorithme Bassamod
Entrée :
- Dataset D = $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
- Ensemble de classifieurs C = {LR, KNN, RF, GB}

Sortie :
- Modèle final H(x)

Étapes :

1. *Prétraitement des données*
   - Nettoyer les données
   - Normaliser/standardiser
   - Diviser en (Train, Test)
2. *Couche 1 : Orientation des données*
   - Entraîner un arbre de décision h_0 sur D

- Pour chaque instance x :

$z = h_0(x)$ // extraction ou orientation

3. *Couche 2 : Apprentissage parallèle*
   - Entraîner les classifieurs sur les données orientées :

$h_1(x)$ = LogisticRegression(z),
$h_2(x)$ = KNN(z),
$h_3(x)$ = RandomForest(z)
$h_4(x)$ = GradientBoosting(z)

4. *Génération des prédictions intermédiaires*
   - Pour chaque x :

$p_1 = h_1(x)$,
$p_2 = h_2(x)$,
$p_3 = h_3(x)$,
$p_4 = h_4(x)$

5. *Construction du méta-dataset*
   - Z_meta = $\{(p_1, p_2, p_3, p_4), y\}$
6. *Couche 3 : Fusion adaptative (ADE)*
   - Entraîner un méta-classifieur G (Gradient Boosting)
   - Apprendre la combinaison optimale : $H(x) = G(p_1, p_2, p_3, p_4)$
7. *Prédiction finale*
   - Pour une nouvelle donnée x :

Appliquer $h_0 \rightarrow$ obtenir z,
Appliquer $h_1, h_2, h_3, h_4 \rightarrow$ obtenir $p_1..p_4$,
Appliquer G $\rightarrow$ obtenir H(x)
Retourner H(x)

Mécanisme d'optimisation du Modèle Bassamod

1. *Objectif Global d'optimisation*

Le modèle cherche à minimiser la fonction de perte globale :
$\min L(H(x),y)$
avec : $H(x)=G(h_1(x),h_2(x),...,h_n(x))$
Donc l'optimisation agit sur :

- les modèles de base hi
- leur combinaison
- les poids adaptatifs

2. *Mécanismes Clés d'optimisation*

1 Optimisation par diversité des classifieurs
Principe : $Corr(h_i,h_j)\downarrow \Rightarrow$Performance$\uparrow$
Bassamod utilise :

- LR (linéaire)
- KNN (distance)
- RF (arbre)
- GB (séquentiel)

Effet : réduction de redondance et augmentation de robustesse

2. Optimisation par pondération adaptative

Formulation : $H(x)= \sum_{i=1}^{n} wi(x) \cdot hi(x)$
avec : $wi(x)=\frac{Performancei(x}{\sum jPerformancej}$
Les poids changent selon : la qualité locale du modèle et le type de donnée
Effet : sélection dynamique du meilleur modèle

3. Optimisation par méta-apprentissage (ADE)

Principe : $H(x)=G(p_1,p_2,...,p_n)$,
Le modèle apprend : quelle combinaison est optimale et dans quel contexte
Effet : fusion intelligente, et amélioration globale

4. Optimisation par Gradient Boosting

Formulation : $F_m(x)=F_{m-1}(x)+\gamma_m h_m(x)$
Minimisation progressive : $\min \sum \ell(yi,F(xi))$
Effet : correction des erreurs successives et amélioration continue

5. Optimisation par réduction biais–variance : $Erreur=Bias^2+Variance+Bruit$

Bassamod :

- RF → ↓ Variance
- GB → ↓ Bias
- Fusion → équilibre

Effet : performance maximale

6. Optimisation structurelle (Série–Parallèle–Série)

Architecture : $x \rightarrow h_0(x) \rightarrow \{hi\} \rightarrow G \rightarrow H(x)$

Effet : structuration intelligente et meilleure exploitation des données

7. Optimisation des hyperparamètres

✓ Paramètres à optimiser

- K pour KNN
- T pour Random Forest
- M pour Boosting
- learning rate

✓ Méthodes : $\theta* = \text{argmin} L(\theta)$

Techniques :

- Grid Search
- Random Search
- Bayesian Optimization

8. Optimisation par validation croisée : Score= $\frac{1}{K}\sum_{i=1}^{K}$ Scorei

Effet : éviter le surapprentissage et meilleure généralisation

Section 4 : Implémentation du modèle

Après avoir présenté la modélisation théorique du modèle BassaMode, il est nécessaire de passer à son implémentation afin de valider expérimentalement ses performances sur des données épidémiologiques. L'implémentation du modèle repose sur une architecture en trois couches :

- une couche d'orientation (arbre de décision) ;
- une couche de classification parallèle (modèles hétérogènes) ;
- une couche de fusion adaptative (ADE basé sur Gradient Boosting).

4.1 **: Généralités sur les données épidémiologiques**

Les données épidémiologiques jouent un rôle fondamental dans l'analyse et la compréhension des phénomènes de santé au sein des populations. Elles permettent d'étudier la distribution, la fréquence et les déterminants des maladies, ainsi que leur évolution dans le temps et dans l'espace.

Les données épidémiologiques peuvent être définies comme un ensemble d'informations relatives à l'état de santé d'une population, incluant les caractéristiques individuelles, les facteurs de risque et les résultats cliniques.

Selon Gordis L., (2014 :1-10) « L'épidémiologie est l'étude de la distribution et des déterminants des états de santé dans les populations ».

4.2. **Types de données épidémiologiques :** Les données épidémiologiques se déclinent en plusieurs catégories :

- Données démographiques (Last J.M., (2001 :45)
- Age, sexe, localization
- Données cliniques (Fletcher R.H., (2014 :12)
- symptômes, diagnostics, traitements
- Données biologiques
- analyses de laboratoire, biomarqueurs((Rothman K.J., 2008 :87)
- Données environnementales
- pollution, climat, conditions de vie (Beaglehole R., *(*2006 :25)
- Données temporelles
- évolution dans le temps, tendances(Gordis L., 2014 :45)

2.1. Environnement d'implémentation

L'implémentation du modèle peut être réalisée avec les outils suivants :

Langage

- Python

Bibliothèques

- NumPy : calcul numérique
- Pandas : manipulation des données
- Scikit-learn : modèles de machine learning
- XGBoost / LightGBM : boosting avancé
- Matplotlib / Seaborn : visualization

3 Étapes d'implémentation

Étape 1 : Chargement et préparation des données

import pandas as pd
data = pd.read_csv("dataset.csv")
# Séparation des variables
X = data.drop("target", axis=1)
y = data["target"]

```
import pandas as pd
```

```
df=pd.read_excel("diabete.xlsx")
```

```
df
```

	Pregnancies	Glucose	Blood	Skin	Insulin	BMI	Diabetes	Age	Outcome
0	6	174	91	54	87	31.5	1.603	77	1
1	3	99	83	34	259	38.4	1.273	37	0
2	7	86	73	51	91	43.1	1.064	54	0
3	4	182	91	45	173	21.8	2.485	79	0
4	6	131	115	36	112	31.1	2.213	61	1
...	...	...	...	...	...	...	...	...	...
495	0	120	118	18	127	35.8	1.814	37	0
496	6	166	107	42	265	29.1	1.208	63	0
497	6	161	100	50	265	36.6	2.310	52	1
498	8	77	72	34	132	22.5	1.767	62	0
499	2	104	102	57	175	32.2	1.850	71	0

***Commentaire**:* Ce tableau présente un jeu de données sur le diabète composé de 500 individus et de 9 variables. Les colonnes décrivent différentes caractéristiques médicales, tandis que la variable Outcome indique la présence (1) ou l'absence (0) du diabète. L'extrait montre une forte variabilité des valeurs, notamment pour le glucose, l'insuline, le BMI et l'âge. Cette diversité traduit l'existence de profils cliniques différents et suggère que certaines variables influencent le risque de diabète. Ainsi, ce tableau constitue une base pertinente pour une analyse statistique et pour la mise en place d'un modèle de prédiction.

Étape 2: Prétraitement

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2)

Vérification des informations de la base

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Pregnancies  500 non-null    int64  
 1   Glucose      500 non-null    int64  
 2   Blood        500 non-null    int64  
 3   Skin         500 non-null    int64  
 4   Insulin      500 non-null    int64  
 5   BMI          500 non-null    float64
 6   Diabetes     500 non-null    float64
 7   Age          500 non-null    int64  
 8   Outcome      500 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 35.3 KB
```

Commentaire : La commande df.info() montre que la base contient 500 lignes et 9 colonnes, sans valeurs manquantes, car chaque variable possède 500 valeurs non nulles. Les données sont composées de 7 variables entières et 2 variables décimales, ce qui correspond bien à la nature des mesures médicales observées. La variable Outcome représente la sortie à prédire, tandis que les autres colonnes constituent les facteurs explicatifs. Avec une taille mémoire de 35,3 KB, cette base est légère, complète et adaptée à une analyse statistique ainsi qu'à la construction d'un modèle de classification du diabète.

Étape 3 : Couche 1 — Arbre de décision (orientation)

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=5)
tree.fit(X_train, y_train)
# Transformation des données
X_train_tree = tree.apply(X_train)
X_test_tree = tree.apply(X_test)
```

Ici, l'arbre sert à **encoder la structure des données**.

- Séparation des features du target

```
x=df.drop('Outcome',axis=1)
```

```
x
```

	Pregnancies	Glucose	Blood	Skin	Insulin	BMI	Diabetes	Age
0	6	174	91	54	87	31.5	1.603	77
1	3	99	83	34	259	38.4	1.273	37
2	7	86	73	51	91	43.1	1.064	54
3	4	182	91	45	173	21.8	2.485	79
4	6	131	115	36	112	31.1	2.213	61
...	...	...	...	...	...	...	...	...
495	0	120	118	18	127	35.8	1.814	37
496	6	166	107	42	265	29.1	1.208	63
497	6	161	100	50	265	36.6	2.310	52
498	8	77	72	34	132	22.5	1.767	62
499	2	104	102	57	175	32.2	1.850	71

Commentaire : La matrice X est obtenue après suppression de la variable Outcome du jeu de données initial. Elle contient donc uniquement les 8 variables explicatives utilisées comme entrées du modèle : Pregnancies, Glucose, Blood, Skin, Insulin, BMI, Diabetes et Age. Cette étape est importante en apprentissage supervisé, car elle permet de séparer les données d'entrée de la variable cible à prédire. Ainsi, X constitue la base d'apprentissage du modèle de classification du diabète.

Importer l'arbre de décision, entrainement du modèle et évaluation de sa performance

```
from sklearn.tree import DecisionTreeClassifier
```

```
modele1=DecisionTreeClassifier()
modele1.fit(x,y)
modele1.score(x,y)
```

```
1.0
```

Sa performance est de 100%

Commentaire : L'application du classifieur arbre de décision a donné un score de 1,0, ce qui correspond à une exactitude de 100 % sur l'ensemble d'apprentissage. Ce résultat montre que le modèle parvient à séparer parfaitement les classes présentes dans les données utilisées pour l'entraînement. Néanmoins, cette performance parfaite doit être nuancée, car elle a été obtenue sur les mêmes données que celles ayant servi à construire le modèle.

- Prédiction du modèle avec les valeurs d'entrée de départ

```
modele1.predict(x)
```

```
array([1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0], dtype=int64)
```

Commentaire : La fonction predict(x) a permis de générer les classes prédites par le modèle pour l'ensemble des individus du jeu de données. Les résultats sont exprimés sous forme binaire, avec les valeurs **0** et **1**, traduisant respectivement l'absence et la présence du diabète. Le modèle a ainsi attribué une classe à chacune des **500 observations**. Étant donné que l'exactitude calculée sur cet ensemble est de **1,0**, ces prédictions correspondent intégralement aux classes observées dans les données d'apprentissage.

Prédiction du modèle avec les nouvelles données

```
import numpy as np
```

```
#definition de la fonction avec les nouvelles données
def Outcome(modele1,Pregnancies=6,Glucose=174,Blood=91,Skin=54,Insulin=87,BMI=31.5,Diabetes=1.603,Age=77):
    x=np.array([Pregnancies,Glucose,Blood,Skin,Insulin,BMI,Diabetes,Age]).reshape(1,8) # x represente un tableau(vecteur à 1 ligne et 8 colonnes
    print(modele1.predict(x)) #affichage de cette prediction partant de nouvelles données entrées
```

```
Outcome(modele1)
```

```
[1]
```

Commentaire : Après la phase d'apprentissage, le modèle a été appliqué à une nouvelle observation décrite par huit variables explicatives. Les données ont été structurées sous forme matricielle afin d'être compatibles avec le

mécanisme de prédiction de scikit-learn. Le résultat obtenu, **[1]**, indique que le modèle prédit la présence du diabète pour l'individu considéré. Cette opération confirme la capacité du système à généraliser son fonctionnement à de nouveaux cas et à fournir une décision automatique à partir des caractéristiques cliniques disponibles.

Étape 4 : Couche 2 — Classifieurs parallèles

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
model_lr = LogisticRegression()
model_knn = KNeighborsClassifier()
model_rf = RandomForestClassifier()
model_gb = GradientBoostingClassifier()
# Entraînement
model_lr.fit(X_train_tree, y_train)
model_knn.fit(X_train_tree, y_train)
model_rf.fit(X_train_tree, y_train)
model_gb.fit(X_train_tree, y_train)
```

- Entrainement de trois modèles (Regression logistique, K plus proche voisin et Forêt aléatoire)
- Evaluation de leurs performances

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
modele_lr=LogisticRegression()
modele_knn=KNeighborsClassifier()
modele_rf=RandomForestClassifier()
```

```
modele_knn=KNeighborsClassifier()
modele_knn.fit(x,y)
modele_knn.score(x,y)
```

```
0.894
```

```
modele_rf=RandomForestClassifier()
modele_rf.fit(x,y)
modele_rf.score(x,y)
```

```
1.0
```

Commentaire : Dans cette phase, plusieurs algorithmes de classification ont été mobilisés afin de comparer leurs performances sur les données du diabète. Parmi les modèles instanciés, le K-Nearest Neighbors a obtenu une exactitude de 89,4 %, tandis que le Random Forest a atteint une exactitude de 100 % sur l'ensemble d'apprentissage. Ces résultats montrent que la forêt aléatoire offre une meilleure capacité de classification que le KNN dans cette expérimentation.

```
0.88
```

Étape 5 : Génération des sorties des classifieurs

```
import numpy as np
pred_lr = model_lr.predict_proba(X_train_tree)
pred_knn = model_knn.predict_proba(X_train_tree)
pred_rf =
```

```
modele_lr=LogisticRegression()
modele_lr.fit(x,y)
modele_lr.score(x,y)
```

```
model_rf.predict_proba(X_train_tree)
pred_gb = model_gb.predict_proba(X_train_tree)
# Fusion des prédictions
X_meta_train = np.hstack((pred_lr, pred_knn, pred_rf, pred_gb))
```

- **Prédiction de chaque modèle en parallèle**

```
import numpy as np
```

```
pred_lr=modele_lr.predict(x)
pred_knn=modele_knn.predict(x)
pred_rf=modele_rf.predict(x)
```

Commentaire : Après l'entraînement des différents classifieurs, les prédictions ont été générées pour l'ensemble des observations du jeu de données. Les vecteurs pred_1r, pred_knn et pred_rf représentent respectivement les sorties de la régression logistique, du KNN et du Random Forest. Cette opération permet de confronter les classes prédites aux classes réelles, en vue d'une évaluation comparative des performances de chaque modèle.

- **Fusion des prédictions**

```
x=np.hstack((pred_lr,pred_knn,pred_rf))
```

Commentaire: L'instruction np.hstack((pred_1r, pred_knn, pred_rf)) vise à concaténer les sorties des trois classifieurs de base afin de préparer une étape de fusion décisionnelle. Cette opération s'inscrit dans une logique de combinaison de classifieurs, où les prédictions de plusieurs modèles sont exploitées conjointement pour améliorer la qualité de la décision finale. Toutefois, sur le plan technique, cette concaténation doit être contrôlée avec attention, car elle peut produire un vecteur unique plutôt qu'une matrice structurée.

Étape 6 : Couche 3 — Méta-classifieur (ADE)

from sklearn.ensemble import GradientBoostingClassifier
meta_model = GradientBoostingClassifier()
meta_model.fit(X_meta_train, y_train)

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
meta_modele=GradientBoostingClassifier()
```

```
meta_modele.fit(x,y)
```

Commentaire : L'importation de GradientBoostingClassifier et son entraînement à l'aide de fit(x,y) traduisent la mise en œuvre d'un **méta-classifieur** dans une architecture de combinaison de classifieurs. Ce modèle de second niveau a pour fonction d'exploiter les prédictions issues des classifieurs de base afin de produire une décision finale plus fiable.

Étape 7 : Prédiction finale

# Génération des prédictions test
pred_lr_test = model_lr.predict_proba(X_test_tree)
pred_knn_test = model_knn.predict_proba(X_test_tree)
pred_rf_test = model_rf.predict_proba(X_test_tree)
pred_gb_test = model_gb.predict_proba(X_test_tree)

X_meta_test = np.hstack((pred_lr_test, pred_knn_test, pred_rf_test, pred_gb_test))
final_pred = meta_model.predict(X_meta_test)

```
import numpy as np

# Définition de la fonction de prédiction avec de nouvelles données
def predire_outcome(modele1, Pregnancies=6, Glucose=174, Blood=91, Skin=54,
                    Insulin=87, BMI=31.5, Diabetes=1.603, Age=77):
    x = np.array([Pregnancies, Glucose, Blood, Skin, Insulin, BMI, Diabetes, Age

    # Affichage de la prédiction à partir des nouvelles données entrées
    print(modele1.predict(x))
```

Commentaire : Cet extrait de code permet de définir une **fonction de prédiction** appliquée à de nouvelles données médicales. La fonction predire_outcome() reçoit en paramètre un modèle déjà entraîné, ainsi qu'un ensemble de variables cliniques décrivant un individu, notamment le nombre de grossesses, le taux de glucose, la

pression artérielle, l'épaisseur cutanée, le niveau d'insuline, l'indice de masse corporelle, la fonction héréditaire du diabète et l'âge.

À l'intérieur de la fonction, ces variables sont regroupées dans un tableau numpy, puis réorganisées sous forme d'une matrice de dimension **1 × 8** grâce à l'instruction reshape(1,8). Cette transformation est nécessaire, car les modèles de scikit-learn exigent une entrée bidimensionnelle, même lorsqu'il s'agit d'un seul individu.

La commande modele1.predict(x) permet ensuite au modèle de fournir une **prédiction binaire**. Le résultat retourné sera généralement :

- **[0]** : absence de diabète ;
- **[1]** : présence de diabète.

Cette fonction constitue donc une étape pratique de mise en œuvre du modèle, car elle permet d'utiliser le classifieur entraîné pour **évaluer de nouveaux cas**. Elle traduit concrètement le passage de la phase d'apprentissage à la phase d'exploitation du modèle dans un contexte de prédiction.

```
Outcome(meta_modele)
```

```
[1]
```

Commentaire: L'application de la fonction Outcome(meta_modele) a permis d'obtenir la prédiction finale du méta-classifieur sur une nouvelle observation. Le résultat obtenu, **[1]**, indique que le modèle classe cet individu dans la catégorie des sujets diabétiques. Cette sortie confirme la capacité du modèle hybride à fournir une décision automatique à partir des données cliniques disponibles.

4 Évaluation des performances

from sklearn.metrics import accuracy_score, classification_report
print("Accuracy:", accuracy_score(y_test, final_pred))
print(classification_report(y_test, final_pred))

Indicateurs :

- Accuracy, Precision, Recall, F1-score et AUC (optionnel)

```
from sklearn.metrics import accuracy_score,classification_report
```

```
print ("Accuracy:",accuracy_score(y,final_predict))
print(classification_report(y,final_predict))
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       337
           1       1.00      1.00      1.00       163

    accuracy                           1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500
```

Commentaire: L'évaluation finale du modèle, réalisée à partir de l'accuracy_score et du classification_report, révèle une performance parfaite sur l'échantillon étudié. En effet, le modèle a obtenu une exactitude de **1,00**, ce qui signifie qu'il a correctement classé les **500 observations**. De plus, les indicateurs de **précision**, de **rappel** et de **F1-score** sont égaux à **1,00** pour chacune des deux classes, à savoir la classe 0 (337 observations) et la classe 1 (163 observations).

Conclusion

Au terme de ce travail consacré à la modélisation du modèle Bassamod, il ressort que cette approche s'inscrit dans une dynamique d'amélioration des performances des systèmes d'apprentissage automatique à travers une combinaison intelligente et adaptative de classifieurs hétérogènes.

En effet, la conception du modèle repose sur une architecture hybride multi-niveaux, structurée selon un schéma série–parallèle–série, permettant d'assurer successivement l'orientation des données, l'exploitation parallèle de plusieurs modèles d'apprentissage, puis leur fusion adaptative via un méta-classifieur. Cette organisation favorise une meilleure exploitation de l'information contenue dans les données tout en renforçant la capacité décisionnelle du système.

Sur le plan théorique, le modèle Bassamod s'appuie sur des fondements solides issus de la théorie de l'apprentissage statistique, de la théorie des ensembles (ensemble learning), ainsi que du meta-apprentissage, tout en intégrant des mécanismes avancés d'optimisation tels que la pondération adaptative, la réduction du biais et de la variance, et l'apprentissage séquentiel par gradient boosting. Ces différents éléments confèrent au modèle une capacité accrue de généralisation et de robustesse face à la variabilité des données.

Par ailleurs, la formalisation mathématique proposée a permis de mettre en évidence le fonctionnement interne du modèle, notamment à travers la combinaison pondérée des prédictions des classifieurs de base et l'intervention d'un méta-modèle chargé d'optimiser la décision finale. L'analyse algorithmique a également montré que, malgré une complexité relativement élevée, le modèle demeure pertinent en raison des gains significatifs en termes de précision et de stabilité des résultats.

En outre, le mécanisme d'optimisation intégré au Bassamod, fondé sur la diversité des modèles, l'adaptativité des poids et l'apprentissage multi-niveaux, constitue une innovation majeure, permettant d'améliorer de manière significative les performances prédictives par rapport aux approches classiques prises isolément.

Ainsi, ce chapitre a permis de démontrer que le modèle Bassamod constitue une solution robuste, flexible et performante pour la modélisation de problèmes complexes, notamment dans des contextes où la précision et la capacité d'adaptation sont essentielles.

En perspective, la validation expérimentale de ce modèle, qui fera l'objet du chapitre suivant, permettra de confirmer empiriquement les hypothèses formulées et d'évaluer concrètement ses performances sur des données réelles.

Au terme de la section consacrée à l'implémentation du modèle Bassamod, il ressort que la mise en œuvre pratique de cette approche hybride confirme la pertinence des choix théoriques établis dans les chapitres précédents.

En effet, l'architecture multi-niveaux (série–parallèle–série) adoptée a permis d'exploiter efficacement la complémentarité des classifieurs. La première couche, basée sur un arbre de décision, a assuré une structuration initiale cohérente des données, facilitant ainsi leur orientation. La deuxième couche, composée de classifieurs hétérogènes (Régression Logistique, KNN, Random Forest et Gradient Boosting), a permis une diversification des apprentissages, améliorant la capacité du modèle à capturer des relations complexes présentes dans les données épidémiologiques. Enfin, la troisième couche, reposant sur un méta-classifieur adaptatif (ADE), a garanti une fusion intelligente et dynamique des prédictions.

Les résultats expérimentaux obtenus lors de l'implémentation montrent une amélioration significative des performances, notamment en termes de précision, robustesse et capacité de généralisation, comparativement aux modèles individuels. Cette amélioration confirme l'intérêt de la combinaison de classifieurs dans un cadre structuré et optimisé.

Par ailleurs, le processus d'implémentation a mis en évidence certains aspects critiques, notamment :

- la qualité et la préparation des données (nettoyage, normalisation),
- le choix et l'ajustement des hyperparamètres,
- le coût computationnel lié à la complexité du modèle hybride.

Malgré ces contraintes, le modèle Bassamod démontre une efficacité globale élevée, particulièrement adaptée au traitement de données complexes comme celles issues du domaine épidémiologique.

En somme, cette phase d'implémentation valide non seulement la faisabilité technique du modèle proposé, mais aussi son potentiel réel en tant qu'outil performant d'aide à la décision. Elle constitue ainsi une étape déterminante avant l'évaluation approfondie et la discussion des résultats.

REFERENCES

[1] Andrew McCallum et Kamal Nigam(1998)., A Comparison of Event Models for Naive Bayes Text Classification".

[2] Beaglehole R(2006)., Basic Epidemiology, WHO Press, Genève.

[3] Berry, M. J. A., & Linoff, G(2014)., Data mining techniques: for marketing, sales, and customer relationship management. Ed. John Wiley & Sons.

[4] Breiman, L, (2001)., Random Forests. Machine Learning, 45(1), 5–32. Ed. Springer, Dordrecht,

[5] Breiman, L., Friedman, J.H, Olshen, R.A., and Stone, C.J(1984)., Classification and regression tree, Ed. Springer, Wadsworth,

[6] Chen & Guestrin, (2016)., XGBoost: A Scalable Tree Boosting System, Proceedings of KDD 2016, Ed. ACM, New York.