Rapport d'Implémentation du Modèle M5P

Auteurs

AMRI Maryam

ZEMMAHI Zakaria

BOUABDILLAH Hachim

Encadrant

Pr. Hosni M

Projet d'Implémentation du M5P

1 Historique et Contexte du M5P

1.1 Origines et Évolution

Le modèle M5P s'inscrit dans une lignée d'algorithmes d'apprentissage automatique visant à améliorer les capacités prédictives des arbres de décision pour les tâches de régression. Son développement représente une évolution naturelle des méthodes d'arbre classiques vers des approches hybrides plus sophistiquées.

1.1.1 Les Arbres de Régression CART

Au début des années 1980, Breiman et al.[1] ont introduit les arbres CART (Classification And Regression Trees), qui utilisent une approche récursive de partitionnement binaire de l'espace des features. Pour la régression, CART prédit une valeur constante (la moyenne) dans chaque région définie par une feuille de l'arbre.

Limitation principale : Les prédictions constantes par feuille limitent la capacité du modèle à capturer des tendances locales complexes, nécessitant des arbres très profonds pour approcher des fonctions lisses.

1.1.2 L'Algorithme M5 Original

En 1992, Ross Quinlan[2] a proposé l'algorithme M5, une innovation majeure introduisant l'idée révolutionnaire de placer des **modèles de régression linéaire** aux feuilles plutôt que de simples constantes.

Caractéristiques clés de M5 :

Construction de l'arbre basée sur la réduction de l'écart type (SDR) comme critère de division.

Ajustement de modèles linéaires multivariés à chaque feuille, capturant ainsi les relations locales entre features et cible.

Mécanisme d'élagage pour simplifier l'arbre et améliorer la généralisation.

Innovation conceptuelle : M5 transforme l'arbre de décision en un "arbre de modèles" (model tree), où chaque région de l'espace d'entrée possède son propre modèle linéaire local.

1.1.3 M5 Prime (M5P) : L'Amélioration Décisive

En 1999, Yong Wang[3] a proposé une amélioration significative de M5, baptisée M5 Prime ou M5P, qui reste aujourd'hui la version de référence. Les améliorations apportées incluent :

1. Lissage des Prédictions (Smoothing) : Introduction d'un mécanisme de lissage qui combine les modèles linéaires des nœuds feuilles avec ceux de leurs ancêtres. Cette technique réduit les discontinuités aux frontières des régions et améliore significativement la performance prédictive.

2. Optimisation de l'Élagage : Raffinement de la procédure d'élagage avec un critère d'erreur ajustée robuste basé sur la formule Weka, pénalisant efficacement la complexité du modèle.

3. Gestion Améliorée de la Complexité : Meilleure pénalisation du nombre de paramètres pour éviter le sur-apprentissage, particulièrement crucial pour les datasets de taille modeste.

1.2 Positionnement dans le Paysage du Machine Learning

Le M5P occupe une position unique dans l'écosystème des algorithmes de régression :

Entre Simplicité et Performance : M5P offre un compromis optimal entre l'interprétabilité des arbres de décision et la précision des modèles linéaires.

Alternative aux Méthodes Ensemblistes : Contrairement aux Random Forests ou au Gradient Boosting qui sacrifient l'interprétabilité pour la performance, M5P maintient une structure explicable tout en rivalisant en précision.

Capture de Non-Linéarités : La segmentation de l'espace d'entrée permet de modéliser des relations non-linéaires complexes via une mosaïque de modèles linéaires locaux.

1.3 Implémentations et Adoption

Weka (Waikato Environment for Knowledge Analysis) : La première implémentation largement diffusée de M5P a été intégrée dans la suite logicielle Weka[4], développée à l'Université de Waikato en Nouvelle-Zélande. Cette implémentation est devenue la référence de facto.

Applications Industrielles : M5P a été appliqué avec succès dans de nombreux domaines : prévision de séries temporelles, modélisation environnementale, prédiction de performances de systèmes, finance, et ingénierie.

Recherche Académique : Plusieurs centaines de publications scientifiques ont étudié, appliqué ou étendu M5P, démontrant sa robustesse et sa polyvalence.

1.4 Contributions de Notre Implémentation

Cette implémentation moderne de M5P apporte plusieurs améliorations et clarifications :

Architecture Modulaire : Séparation claire des responsabilités entre construction, élagage, lissage et prédiction, facilitant la maintenance et l'extension.

Double Formulation de l'Élagage : Utilisation de la formule Weka éprouvée pour l'élagage, offrant un bon équilibre entre complexité et performance prédictive.

Régularisation Ridge Intégrée : Gestion native de la multicolinéarité dans les modèles linéaires locaux pour améliorer la stabilité numérique.

Documentation Complète : Algorithmes détaillés et explications mathématiques rigoureuses de chaque composant du modèle.

Le modèle M5P représente ainsi plus de 30 ans d'évolution dans le domaine des arbres de régression, synthétisant les meilleures idées de CART, M5 original, et les raffinements de M5 Prime pour offrir une solution mature et éprouvée pour la régression supervisée.

2 Construction de l'Arbre de Régression M5P

Le modèle M5P, une amélioration de l'arbre de régression M5, est conçu pour des tâches de régression en combinant la puissance des arbres de décision pour la segmentation de l'espace des entrées avec des modèles linéaires locaux dans les nœuds terminaux (feuilles). La phase de construction de l'arbre est cruciale et repose sur la détermination du meilleur critère de division à chaque nœud.

2.1 Critère de Division : Réduction de l'Écart Type (SDR)

Dans les arbres de régression classiques, la qualité d'une division est souvent évaluée par la réduction de l'erreur quadratique moyenne (MSE) ou de la somme des carrés des erreurs (SSE). Cependant, le modèle M5P utilise un critère spécifique : la **Réduction de l'Écart Type** (*Standard Deviation Reduction*, SDR).

2.1.1 Explication du choix du SDR plutôt que du MSE

Le SDR est privilégié dans la construction de l'arbre M5P pour des raisons conceptuelles et pratiques liées à la prédiction dans un contexte d'arbre de régression :

Mesure de l'Homogénéité : L'écart type (σ) est une mesure directe de la dispersion des valeurs cibles (y) dans un nœud. L'objectif principal de la division d'un nœud en deux sous-nœuds (gauche L et droit R) est de créer des sous-ensembles de données plus homogènes en termes de valeurs y. Réduire l'écart type dans les sous-nœuds par rapport au nœud parent est une façon intuitive de quantifier l'amélioration de cette homogénéité.

Cohérence avec l'Arbre de Décision : Dans les arbres de classification, l'Information Gain ou l'Indice de Gini mesurent la réduction de l'impureté. Le SDR est l'analogue direct pour la régression, mesurant la réduction de l'« impureté » de la régression (variance) par la division.

Formulation et Interprétation : La SDR est définie comme :

Formule de la SDR :

$$\text{SDR} = \sigma(\text{Parent}) - \left[\frac{|L|}{|P|} \times \sigma(L) + \frac{|R|}{|P|} \times \sigma(R)\right]$$

où P est l'ensemble parent, L et R sont les enfants, et $\sigma(\cdot)$ est l'écart type des valeurs y dans l'ensemble. La division choisie est celle qui maximise cette réduction, c'est-à-dire qui maximise la différence entre l'hétérogénéité du parent et la moyenne pondérée de l'hétérogénéité des enfants.

Relation avec les Modèles Linéaires : Bien que le MSE soit le critère d'optimisation pour les modèles de régression linéaire qui seront finalement placés aux feuilles, le SDR sert d'abord à construire la structure de l'arbre (les divisions). Cette structure définit les régions dans l'espace des caractéristiques où un modèle linéaire local sera appliqué. Le SDR est un bon proxy pour l'homogénéité nécessaire avant l'ajustement du modèle linéaire.

2.2 Pseudo-codes pour la Division (Module `split`)

Les fonctions suivantes sont implémentées dans le module `split.py` et sont essentielles pour la recherche du meilleur critère de division basé sur le SDR.

2.2.1 Calcul de l'écart-type

Algorithm 1 Calcul de l'écart-type (`std_deviation`)

Fonction STD_DEVIATION($valeurs$)
  si $|valeurs| \leq 1$ **alors**
    retourner 0
  fin si
  $\mu \leftarrow \frac{1}{|valeurs|} \sum_{v \in valeurs} v$
  $variance \leftarrow \frac{1}{|valeurs|} \sum_{v \in valeurs} (v - \mu)^2$
  retourner $\sqrt{variance}$
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(n)$ où $n = |valeurs|$

- Calcul de la moyenne : $O(n)$ (une itération sur toutes les valeurs)
- Calcul de la variance : $O(n)$ (une itération sur toutes les valeurs)
- Racine carrée : $O(1)$

Complexité Spatiale : $O(1)$ - uniquement des variables scalaires (μ, $variance$)

2.2.2 Calcul du SDR

Algorithm 2 Calcul de la réduction de déviation standard (`calculate_sdr`)

Fonction CALCULATE_SDR($y_{parent}, y_{gauche}, y_{droit}$)
  si $|y_{parent}| = 0$ **alors**
    retourner 0
  fin si
  $\sigma_{parent} \leftarrow$ STD_DEVIATION(y_{parent})
  $\sigma_{gauche} \leftarrow$ STD_DEVIATION(y_{gauche})
  $\sigma_{droit} \leftarrow$ STD_DEVIATION(y_{droit})

  $n_{total} \leftarrow |y_{parent}|$
  $n_{gauche} \leftarrow |y_{gauche}|$
  $n_{droit} \leftarrow |y_{droit}|$

  $\sigma_{pondéré} \leftarrow \frac{n_{gauche}}{n_{total}} \times \sigma_{gauche} + \frac{n_{droit}}{n_{total}} \times \sigma_{droit}$

  retourner $\sigma_{parent} - \sigma_{pondéré}$
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(n)$ où $n = |y_{parent}|$

- Calcul de σ_{parent} : $O(n)$
- Calcul de σ_{gauche} : $O(n_L)$ où $n_L = |y_{gauche}|$
- Calcul de σ_{droit} : $O(n_R)$ où $n_R = |y_{droit}|$
- Comme $n_L + n_R = n$, la complexité totale est $O(n)$
- Calculs arithmétiques : $O(1)$

Complexité Spatiale : $O(1)$ - uniquement des variables scalaires

2.2.3 Recherche du meilleur split pour une feature

Algorithm 3 Trouver le meilleur split pour une feature donnée (`find_split_for_feature`)

Fonction FIND_SPLIT_FOR_FEATURE($X, y, feature\_idx, min\_samples$)
  $valeurs \leftarrow X[:, feature\_idx]$
  $valeurs\_uniques \leftarrow$ UNIQUE_SORTED($valeurs$)

  si $|valeurs\_uniques| < 2$ **alors**
    retourner NULL
  fin si

  $meilleur\_sdr \leftarrow -\infty$
  $meilleur\_seuil \leftarrow$ NULL
  $meilleur\_masque\_gauche \leftarrow$ NULL
  $meilleur\_masque\_droit \leftarrow$ NULL

  pour $i \leftarrow 0$ **à** $|valeurs\_uniques| - 2$ **faire**
    $seuil \leftarrow \frac{valeurs\_uniques[i]+valeurs\_uniques[i+1]}{2}$

    $masque\_gauche \leftarrow (valeurs \leq seuil)$
    $masque\_droit \leftarrow \neg masque\_gauche$

    si $\sum masque\_gauche < min\_samples$ **ou** $\sum masque\_droit < min\_samples$ **alors**
      continuer
    fin si

    $y_{gauche} \leftarrow y[masque\_gauche]$
    $y_{droit} \leftarrow y[masque\_droit]$

    $sdr \leftarrow$ CALCULATE_SDR(y, y_{gauche}, y_{droit})

    si $sdr > meilleur\_sdr$ **alors**
      $meilleur\_sdr \leftarrow sdr$
      $meilleur\_seuil \leftarrow seuil$
      $meilleur\_masque\_gauche \leftarrow masque\_gauche$
      $meilleur\_masque\_droit \leftarrow masque\_droit$
    fin si
  fin pour

  si $meilleur\_seuil =$ NULL **alors**
    retourner NULL
  fin si

  retourner {feature : $feature\_idx$, threshold : $meilleur\_seuil$, sdr : $meilleur\_sdr$, left_mask : $meilleur\_masque\_gauche$, right_mask : $meilleur\_masque\_droit$}
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(n \log n + u \cdot n)$ où n est le nombre d'échantillons et u le nombre de valeurs uniques

- Extraction de la colonne : $O(n)$
- Tri et extraction des valeurs uniques : $O(n \log n)$
- Boucle principale : $O(u)$ itérations
- À chaque itération :
  - Création des masques : $O(n)$
  - Filtrage de y : $O(n)$
  - Calcul du SDR : $O(n)$
- Total : $O(n \log n + u \cdot n)$
- Dans le pire cas ($u = n$) : $O(n^2)$
- Cas moyen ($u \ll n$) : $O(n \log n)$

Complexité Spatiale : $O(n + u)$ pour stocker les masques et les valeurs uniques

2.2.4 Recherche du meilleur split global

Algorithm 4 Trouver le meilleur split parmi toutes les features (`find_best_split`)

Fonction FIND_BEST_SPLIT($X, y, min\_samples$)
    $n_{samples}, n_{features} \leftarrow \text{shape}(X)$

    si $n_{samples} < min\_samples$ **alors**
        retourner NULL
    fin si

    $meilleur\_split \leftarrow$ NULL
    $meilleur\_sdr \leftarrow -\infty$

    pour $feat\_idx \leftarrow 0$ **à** $n_{features} - 1$ **faire**
        $split \leftarrow$ FIND_SPLIT_FOR_FEATURE($X, y, feat\_idx, min\_samples$)

        si $split \neq$ NULL **et** $split.sdr > meilleur\_sdr$ **alors**
            $meilleur\_sdr \leftarrow split.sdr$
            $meilleur\_split \leftarrow split$
        fin si
    fin pour

    retourner $meilleur\_split$
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(d \cdot n \log n + d \cdot u \cdot n)$ où d est le nombre de features, n le nombre d'échantillons et u le nombre moyen de valeurs uniques

- Boucle sur toutes les features : $O(d)$ itérations
- Pour chaque feature, appel à `find_split_for_feature` : $O(n \log n + u \cdot n)$
- Total : $O(d \cdot (n \log n + u \cdot n))$
- Pire cas ($u = n$) : $O(d \cdot n^2)$
- Cas moyen ($u \ll n$) : $O(d \cdot n \log n)$

Complexité Spatiale : $O(n)$ pour stocker les masques et résultats intermédiaires

2.3 Réduction de la Variance dans les Arbres de Régression

La construction d'un arbre de régression, qu'il utilise le SDR, le MSE ou autre, est fondamentalement un processus de réduction de la variance.

Objectif Global : L'objectif d'un arbre de régression est de partitionner l'espace d'entrée en régions $R_1, R_2, \ldots, R_M$ (correspondant aux feuilles) de manière à ce que la variable cible y soit presque constante (ou plus facilement modélisable par une fonction simple) au sein de chaque région.

Processus Récursif : Chaque division (nœud) est une étape dans ce processus. La division d'un ensemble de données S en S_L et S_R est réussie si la variance (ou l'écart type) des valeurs cibles dans S_L et S_R est significativement inférieure à celle dans S.

Justification Mathématique : L'utilisation du SDR garantit qu'à chaque étape, l'algorithme sélectionne la division qui offre la plus grande réduction de la dispersion. Une grande valeur de SDR implique que la division est très efficace pour isoler des sous-ensembles dont les valeurs y sont plus proches de leurs moyennes respectives que ne l'étaient les valeurs du nœud parent par rapport à sa moyenne.

Sur-apprentissage (Overfitting) : Un arbre non contraint continuera à diviser les données tant qu'une SDR positive est trouvée, conduisant finalement à des feuilles contenant très peu d'échantillons (potentiellement un seul) et à un écart type nul. Ceci est la cause principale du sur-apprentissage. Le modèle M5P y fait face avec des critères d'arrêt (`min_samples_split`, `max_depth`, variance proche de zéro) et des techniques post-construction comme l'élagage (pruning) et le lissage (smoothing).

Point clé : L'efficacité de la construction de l'arbre dépend directement de la capacité du SDR à identifier les divisions les plus informatives qui créent les partitions les plus homogènes.

3 Régression Linéaire aux Nœuds

Une fois l'arbre construit par maximisation du SDR, des modèles de régression linéaire doivent être ajustés aux données de chaque nœud. Cette phase transforme l'arbre de segmentation en un véritable arbre de modèles. Le module `regression.py` implémente cette fonctionnalité cruciale en utilisant la méthode des moindres carrés ordinaires (OLS) avec gestion robuste de la multicolinéarité.

3.1 Fondements Théoriques de la Régression OLS

Pour un nœud contenant n échantillons avec d features, le modèle linéaire cherche à minimiser l'erreur quadratique totale :

$$\min_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{d} \beta_j x_{ij} \right)^2$$

où β_0 est l'intercept (ordonnée à l'origine) et $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_d]^T$ sont les coefficients associés aux features.

3.1.1 Formulation Matricielle

En introduisant une colonne de uns pour représenter l'intercept, on définit la matrice augmentée :

$$\mathbf{X}_{aug} = [\mathbf{1}_n \mid \mathbf{X}] \in \mathbb{R}^{n \times (d+1)}$$
$$\boldsymbol{\theta} = [\beta_0, \beta_1, \ldots, \beta_d]^T \in \mathbb{R}^{d+1}$$

Le problème de minimisation devient :

$$\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}_{aug}\boldsymbol{\theta}\|^2$$

La solution analytique des moindres carrés ordinaires est donnée par les équations normales :

$$\boldsymbol{\theta}^* = (\mathbf{X}_{aug}^T \mathbf{X}_{aug})^{-1} \mathbf{X}_{aug}^T \mathbf{y}$$

Cette solution est optimale si la matrice $\mathbf{X}_{aug}^T \mathbf{X}_{aug}$ est inversible (rang plein).

3.1.2 Problème de la Multicolinéarité

Dans les arbres de régression, certaines feuilles peuvent contenir peu d'échantillons ou des features fortement corrélées, conduisant à une matrice $\mathbf{X}_{aug}^T \mathbf{X}_{aug}$ singulière ou mal conditionnée (nombre de conditionnement élevé).

Conséquences de la Singularité :

- La matrice n'est pas inversible ($\det(\mathbf{X}_{aug}^T \mathbf{X}_{aug}) = 0$ ou ≈ 0)
- Les coefficients deviennent instables avec une variance infinie
- De petites perturbations dans les données causent de grandes variations dans $\boldsymbol{\theta}$
- La solution OLS n'est pas unique

Régularisation Ridge : Pour pallier ce problème, on introduit un terme de pénalisation L_2 :

$$\boldsymbol{\theta}_{ridge}^* = (\mathbf{X}_{aug}^T \mathbf{X}_{aug} + \lambda \mathbf{I}_{d+1})^{-1} \mathbf{X}_{aug}^T \mathbf{y}$$

où $\lambda = 10^{-6}$ est un petit paramètre de régularisation et $\mathbf{I}_{d+1}$ est la matrice identité de dimension $(d+1) \times (d+1)$.

Effet de la Régularisation :

- Garantit l'inversibilité : $\mathbf{X}_{aug}^T \mathbf{X}_{aug} + \lambda \mathbf{I}$ est toujours définie positive pour $\lambda > 0$
- Stabilise numériquement les calculs en améliorant le conditionnement de la matrice
- Introduit un biais minimal pour λ petit, tout en réduisant significativement la variance
- Pénalise les coefficients de grande magnitude, favorisant des solutions plus régulières

3.2 Implémentation : Module `regression.py`

3.2.1 Ajustement du Modèle Linéaire

Algorithm 5 Ajustement du Modèle Linéaire par OLS/Ridge (`fit_linear_model`)

Fonction FIT_LINEAR_MODEL(X, y)
  $n_{samples}, n_{features} \leftarrow \text{shape}(X)$

  // Construction de la matrice augmentée avec colonne de biais
  $X_{aug} \leftarrow [\mathbf{1}_{n_{samples}} \mid X]$ ▷ Matrice $n \times (d+1)$

  // Tentative de résolution par moindres carrés ordinaires
  $\boldsymbol{\theta}, residuals, rank, s \leftarrow \text{lstsq}(X_{aug}, y, rcond = None)$

  // Détection de multicolinéarité par test de rang
  si $rank < (n_{features} + 1)$ **alors** ▷ Rang déficient : matrice singulière
    $\lambda \leftarrow 10^{-6}$ ▷ Paramètre de régularisation Ridge

    // Calcul des matrices pour Ridge
    $X_{aug}^T X_{aug} \leftarrow X_{aug}^T \cdot X_{aug}$ ▷ Matrice de Gram $(d+1) \times (d+1)$
    $X_{aug}^T y \leftarrow X_{aug}^T \cdot y$ ▷ Vecteur $(d+1) \times 1$

    // Résolution Ridge : $(X_{aug}^T X_{aug} + \lambda I)\boldsymbol{\theta} = X_{aug}^T y$
    $\boldsymbol{\theta} \leftarrow \text{solve}(X_{aug}^T X_{aug} + \lambda \mathbf{I}_{d+1}, X_{aug}^T y)$
  fin si

  // Extraction et structuration des paramètres
  retourner $\{\text{intercept} : \theta_0, \text{coefficients} : [\theta_1, \ldots, \theta_d]\}$
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(n \cdot d^2 + d^3)$ où n est le nombre d'échantillons et d le nombre de features

- Construction de X_{aug} (concaténation) : $O(n \cdot d)$
- Résolution par `lstsq` (décomposition SVD ou QR) : $O(n \cdot d^2 + d^3)$
- En cas de régularisation Ridge :
  - Calcul de $X_{aug}^T X_{aug}$: $O(n \cdot d^2)$ (produit matriciel)
  - Calcul de $X_{aug}^T y$: $O(n \cdot d)$ (produit matrice-vecteur)
  - Addition $X_{aug}^T X_{aug} + \lambda I$: $O(d^2)$
  - Résolution du système (Cholesky ou LU) : $O(d^3)$
- Total : $O(n \cdot d^2 + d^3)$
- Cas typique ($n \gg d$) : Dominé par $O(n \cdot d^2)$
- Cas particulier ($d \approx n$) : $O(n^3)$

Complexité Spatiale : $O(n \cdot d + d^2)$

- Stockage de X_{aug} : $O(n \cdot d)$
- Stockage de $X_{aug}^T X_{aug}$ (si Ridge) : $O(d^2)$
- Vecteur solution $\boldsymbol{\theta}$: $O(d)$
- Vecteurs et matrices temporaires : $O(n + d^2)$
- Total : $O(n \cdot d + d^2)$

3.2.2 Prédiction avec le Modèle Linéaire

Une fois le modèle ajusté, la prédiction pour de nouveaux échantillons est une simple opération linéaire vectorisée.

Algorithm 6 Prédiction Linéaire (`predict_linear`)

Fonction PREDICT_LINEAR($model, X$)
  si X est un vecteur unidimensionnel **alors**
    $X \leftarrow \text{reshape}(X, 1, -1)$ ▷ Conversion en matrice $1 \times d$
  fin si

  // Calcul vectorisé : $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} + \beta_0 \mathbf{1}_m$
  retourner $X \cdot model.\text{coefficients} + model.\text{intercept}$
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(m \cdot d)$ où m est le nombre d'échantillons à prédire et d le nombre de features

- Test de dimension : $O(1)$
- Reshape (si nécessaire) : $O(1)$ (changement de métadonnées)
- Produit matrice-vecteur $\mathbf{X} \cdot \boldsymbol{\beta}$: $O(m \cdot d)$
- Addition vectorielle (broadcast de l'intercept) : $O(m)$
- Total : $O(m \cdot d)$

Complexité Spatiale : $O(m)$ pour stocker le vecteur de prédictions $\hat{\mathbf{y}}$

3.3 Stratégie Adaptative : OLS vs Ridge

L'implémentation adopte une stratégie adaptative intelligente pour choisir entre OLS pur et régularisation Ridge :

3.3.1 Critère de Décision

Le test de rang de la matrice $\mathbf{X}_{aug}$ via la décomposition en valeurs singulières (SVD) dans `lstsq` permet de détecter automatiquement les cas problématiques :

Rang Plein ($rank = d + 1$) :

- La matrice $\mathbf{X}_{aug}^T \mathbf{X}_{aug}$ est inversible
- Les équations normales ont une solution unique et stable
- Utilisation de la solution OLS standard retournée par `lstsq`
- Aucune pénalité supplémentaire n'est nécessaire

Rang Déficient ($rank < d + 1$) :

- Multicolinéarité détectée (features linéairement dépendantes)
- Ou nombre d'échantillons insuffisant ($n \leq d$)
- Activation automatique de la régularisation Ridge
- Garantie d'une solution unique et numériquement stable

3.3.2 Avantages de l'Approche Adaptative

Efficacité Computationnelle : La régularisation Ridge n'est appliquée que lorsque strictement nécessaire, évitant un coût calculatoire inutile dans les cas bien conditionnés (majorité des nœuds internes avec suffisamment d'échantillons).

Biais Minimal : Pour $\lambda = 10^{-6}$ très petit, la solution Ridge est quasi-identique à la solution OLS lorsque le système est bien conditionné. Le biais introduit est négligeable ($\|\boldsymbol{\theta}_{ridge} - \boldsymbol{\theta}_{OLS}\| \approx 0$).

Stabilité Numérique Garantie : Dans tous les cas de figure (rang plein ou déficient), la fonction retourne des coefficients numériquement stables, prévenant les explosions de valeurs ou les NaN.

Robustesse aux Données Difficiles : Les feuilles terminales avec peu d'échantillons ou des features redondantes sont automatiquement gérées sans intervention manuelle ni paramétrage supplémentaire.

4 Optimisation du Modèle M5P : Élagage et Lissage

Une fois l'arbre initialement construit par maximisation du SDR, deux étapes cruciales permettent d'améliorer sa robustesse et sa précision en régression : l'élagage (pruning) et le lissage (smoothing). Ces techniques sont implémentées dans le module `pruning.py`.

4.1 Élagage (Pruning) basé sur l'Erreur Ajustée

L'élagage post-construction vise à remplacer un sous-arbre par un unique nœud terminal (feuille) si le modèle linéaire de ce nœud unique est plus performant (en termes de généralisation) que l'ensemble du sous-arbre qu'il remplace.

4.1.1 Critère de l'Erreur Ajustée

Pour évaluer si un sous-arbre doit être élagué, le modèle M5P utilise un critère qui pénalise la complexité (le nombre de paramètres). La complexité est intégrée dans le calcul de l'erreur ajustée, selon une formule spécifique développée dans l'implémentation Weka.

Formulation Mathématique :

L'erreur brute (ici, l'Erreur Quadratique Moyenne, MSE) est ajustée par un facteur de pénalité dépendant du nombre d'échantillons et du nombre de paramètres.

(a) Erreur Quadratique Moyenne (MSE) : $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

(b) Formule Weka : Erreur Ajustée = $\text{MSE} \times \frac{n+\lambda \cdot v}{n-v}$ où n est le nombre d'échantillons au nœud, v est le nombre de paramètres du modèle (n_features + 1), et λ est le facteur de pénalité (typiquement $\lambda = 2.0$).

Cette formule pénalise davantage les modèles complexes (avec beaucoup de paramètres) en augmentant l'erreur ajustée proportionnellement au nombre de paramètres. Le dénominateur $(n - v)$ assure que le modèle a suffisamment de degrés de liberté.

Le sous-arbre est élagué si l'erreur ajustée du modèle linéaire parent est inférieure ou égale à la somme pondérée des erreurs ajustées des feuilles du sous-arbre.

4.1.2 Pseudo-code de l'Élagage

Algorithm 7 Fonction d'Élagage Récursif (`prune_tree`)

Procédure PRUNE_TREE($node, penalty\_factor$)
  si $node$.is_leaf **alors**
    retourner
  fin si
  PRUNE_TREE($node$.left, $penalty\_factor$)
  PRUNE_TREE($node$.right, $penalty\_factor$)
  subtree_params ← COUNT_SUBTREE_PARAMS($node$)
  $n_{samples} \leftarrow len(node.y)$

  raw_error_subtree ← $\sum_{\text{feuilles } L} \text{MSE}(L) \times len(L.y)$
  subtree_err_adj ← ADJUSTED_ERROR_WEKA(raw_error_subtree, $n_{samples}$, subtree_params, $penalty$
  raw_error_linear ← COMPUTE_MSE($node$, $node$.linear_model)
  linear_params ← $len(node.\text{linear\_model.coefficients}) + 1$
  linear_err_adj ← ADJUSTED_ERROR_WEKA(raw_error_linear, $n_{samples}$, linear_params, $penalty\_fac$

  si linear_err_adj ≤ subtree_err_adj **alors**
    $node$.is_leaf ← Vrai
    $node$.left ← NULL
    $node$.right ← NULL
  fin si
fin Procédure

Analyse de Complexité

Complexité Temporelle : $O(N \cdot (n + d))$ où N est le nombre de nœuds, n le nombre moyen d'échantillons par nœud et d le nombre de features

- Parcours post-ordre de l'arbre : visite chaque nœud une fois → $O(N)$
- À chaque nœud :
  - Comptage des paramètres du sous-arbre : $O(L)$ où L est le nombre de feuilles du sous-arbre
  - Calcul de l'erreur du sous-arbre : $O(L \cdot n)$ (parcours des feuilles et calcul MSE)
  - Calcul de l'erreur du modèle linéaire : $O(n \cdot d)$ (prédiction pour n échantillons avec d features)
  - Calcul des erreurs ajustées : $O(1)$
- Complexité totale : $O(N \cdot (n + d))$ en moyenne
- Pire cas (arbre déséquilibré) : $O(N \cdot n \cdot d)$

Complexité Spatiale : $O(h)$ où h est la hauteur de l'arbre (pile de récursion)

4.2 Lissage (Smoothing) des Modèles aux Feuilles

Le lissage est une technique exclusive aux arbres M5P (introduite dans M5 Prime) qui vise à réduire les discontinuités de prédiction aux frontières des régions.

4.2.1 Principe du Lissage M5P

Pour chaque feuille L, le modèle linéaire final est une combinaison pondérée de son modèle linéaire local (ajusté sur les données de la feuille) et du modèle linéaire lissé de son nœud parent immédiat, récursivement jusqu'à la racine.

Cette interpolation permet d'assurer des transitions plus douces entre les prédictions des régions adjacentes, améliorant la performance sur des données bruitées ou petites.

4.2.2 Formulation Mathématique du Lissage

Le vecteur des paramètres θ (contenant l'intercept et les coefficients) du modèle lissé ($\theta_{smoothed}$) d'un nœud est calculé comme une moyenne pondérée :

$$\theta_{smoothed} = \frac{n \cdot \theta_{node} + k \cdot \theta_{parent\_smoothed}}{n + k}$$

où :
θ_{node} est le vecteur de paramètres du modèle non lissé de la feuille.
$\theta_{parent\_smoothed}$ est le vecteur de paramètres du modèle lissé du nœud parent.
n est le nombre d'échantillons dans la feuille.
k est une constante de lissage (typiquement $k = 15$ dans l'implémentation originale du M5).

4.2.3 Pseudo-code du Lissage

Algorithm 8 Lissage des Prédictions (`smooth_predictions`)

Procédure SMOOTH_PREDICTIONS($node, parent\_model, k$)
  si $parent\_model$ = NULL **alors**
    $node$.smoothed_model ← $node$.linear_model
    model_to_pass ← $node$.smoothed_model
  sinon
    $n \leftarrow len(node.y)$
    intcpt ← $\frac{n \cdot node.\text{linear\_model.intercept} + k \cdot parent\_model.intercept}{n+k}$
    coeffs ← $\frac{n \cdot node.\text{linear\_model.coefficients} + k \cdot parent\_model.coefficients}{n+k}$
    $node$.smoothed_model ← {intercept : intcpt, coefficients : coeffs}
    model_to_pass ← $node$.smoothed_model
  fin si
  si $node$.is_leaf **alors**
    retourner
  fin si
  SMOOTH_PREDICTIONS($node$.left, model_to_pass, k)
  SMOOTH_PREDICTIONS($node$.right, model_to_pass, k)
fin Procédure

Analyse de Complexité

Complexité Temporelle : $O(N \cdot d)$ où N est le nombre de nœuds et d le nombre de features

- Parcours pré-ordre de l'arbre : visite chaque nœud une fois $\rightarrow O(N)$
- À chaque nœud :
  - Calcul de l'intercept lissé : $O(1)$
  - Calcul des coefficients lissés (vecteur de taille d) : $O(d)$
  - Copie du modèle : $O(d)$
- Total : $O(N \cdot d)$

Complexité Spatiale : $O(h \cdot d)$ où h est la hauteur de l'arbre

- Pile de récursion : $O(h)$ niveaux
- À chaque niveau, stockage d'un modèle (vecteur de taille d) : $O(d)$
- Total : $O(h \cdot d)$

5 Architecture Globale du Modèle M5P

Le cœur de l'implémentation est encapsulé dans la classe `M5P` du module `model.py`, qui orchestre les trois phases principales du modèle : construction, modélisation linéaire/élagage, et lissage.

5.1 Phase d'Entraînement : Fonction `fit`

La méthode `fit` est le point d'entrée qui exécute la séquence complète de construction et d'optimisation de l'arbre M5P.

5.1.1 Flux d'Exécution Séquentiel

L'entraînement se déroule en plusieurs étapes strictement séquentielles, garantissant que les modèles linéaires sont ajustés avant l'élagage, et que l'élagage précède le lissage.

1. Construction Initiale de l'Arbre : L'arbre de régression est bâti récursivement en utilisant la maximisation de la Réduction de l'Écart Type (SDR) comme critère de division. Cette étape définit la structure de segmentation de l'espace d'entrée.

2. Attachement des Données : Les données (X, y) d'entraînement sont réinjectées et attachées à chaque nœud terminal (feuille) pour l'ajustement du modèle linéaire.

3. Ajustement des Modèles Linéaires : Un modèle de régression linéaire par moindres carrés (OLS, avec gestion de la multi-colinéarité par Ridge) est ajusté sur les données de chaque nœud non élagué (y compris les nœuds internes).

4. Élagage (Pruning) Post-Construction : Si l'hyperparamètre `prune` est activé, l'algorithme élague l'arbre en utilisant le critère de l'Erreur Ajustée basé sur la formule Weka avec le facteur de pénalité `penalty_factor`, remplaçant les sous-arbres par un modèle linéaire unique si cela réduit l'erreur ajustée.

5. Lissage (Smoothing) : Si l'hyperparamètre `smoothing` est activé, les modèles linéaires aux feuilles sont lissés en une combinaison pondérée avec les modèles lissés de leurs parents pour améliorer la continuité de la prédiction.

5.1.2 Pseudo-code de l'Entraînement

Algorithm 9 Méthode d'Entraînement Globale (`M5P.fit`)

Fonction FIT(X, y)
    // Étape 1 : Construction de l'arbre
    tree ← M5PTREE.FIT($X, y, self.min\_samples\_split, self.max\_depth$)

    // Étape 2 : Attacher les données pour la régression et l'élagage
    _ADD_DATA_TO_NODES($tree.root, X, y$)

    // Étape 3 : Ajuster les modèles linéaires à tous les nœuds
    _ADD_LINEAR_MODELS($tree.root$)

    // Étape 4 : Élagage
    si $self.prune$ **alors**
        PRUNE_TREE($tree.root, self.penalty\_factor, self.use\_weka\_formula$)
    fin si

    // Étape 5 : Lissage
    si $self.smoothing$ **alors**
        SMOOTH_PREDICTIONS($tree.root$, NULL, $k = 15$)
    fin si

    $self.tree \leftarrow tree$
    retourner $self$
fin Fonction

Analyse de Complexité Globale

Complexité Temporelle : $O(d \cdot n \log n \cdot h + N \cdot n \cdot d)$ où : **Détail par étape :**

1. Construction de l'arbre : $O(d \cdot n \log n \cdot h)$
   - À chaque niveau (hauteur h)
   - Recherche du meilleur split pour d features : $O(d \cdot n \log n)$
2. Attachement des données : $O(n \cdot h)$ (parcours et filtrage)
3. Ajustement des modèles linéaires : $O(N \cdot (n \cdot d^2 + d^3))$
   - Pour chaque nœud N : résolution OLS en $O(n \cdot d^2 + d^3)$
4. Élagage (optionnel) : $O(N \cdot (n + d))$
5. Lissage (optionnel) : $O(N \cdot d)$

Complexité Spatiale : $O(N \cdot (n + d))$

- Stockage de l'arbre : $O(N)$ nœuds
- Données à chaque nœud : $O(n)$ échantillons
- Modèles linéaires : $O(d)$ paramètres par nœud

5.2 Configuration des Hyperparamètres

Les hyperparamètres suivants contrôlent la complexité et la performance du modèle M5P :

`min_samples_split` : Nombre minimal d'échantillons requis pour autoriser une division. Contrôle la profondeur de l'arbre initial.

`max_depth` : Profondeur maximale de l'arbre.

`prune` (booléen) : Active l'élagage post-construction.

`smoothing` (booléen) : Active le lissage des modèles aux feuilles.

`penalty_factor` : λ utilisé dans la formule d'erreur ajustée Weka pour pénaliser la complexité du modèle ($\lambda = 2.0$ par défaut).

6 Évaluation et Comparaison des Stratégies

Les fichiers d'évaluation (`benchmark.py`, `compare_weka_vs_aic.py`, `eval_prun_smooth.py`) mettent en lumière l'impact des différentes techniques d'optimisation sur les métriques de performance courantes en régression.

6.1 Métriques de Performance

L'évaluation de la qualité de prédiction est essentielle et repose sur les métriques du module `utils.py` :

Erreur Quadratique Moyenne (MSE) : $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Erreur Absolue Moyenne (MAE) : $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

Score R^2 (Coefficient de Détermination) : $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

6.2 Impact de la Formule d'Élagage

L'utilisation de la formule Weka pour l'élagage est une caractéristique distinctive du M5P. Cette formule pénalise la complexité du modèle de manière adaptative en fonction du nombre d'échantillons et de paramètres.

Formule Weka $\left(\text{Erreur Ajustée} = \text{MSE} \times \frac{n+\lambda v}{n-v}\right)$: Cette formule augmente l'erreur ajustée proportionnellement au nombre de paramètres v, tout en tenant compte du nombre d'échantillons n. Le facteur de pénalité λ (typiquement 2.0) contrôle l'agressivité de l'élagage.

Comportement : La formule devient plus stricte lorsque le rapport échantillons/paramètres diminue, encourageant ainsi la création de modèles plus simples dans les régions avec peu de données. Cela aide à prévenir le sur-apprentissage dans les feuilles avec un faible nombre d'observations.

6.3 Rôle Combiné de l'Élagage et du Lissage

L'évaluation révèle que l'élagage seul réduit la complexité de l'arbre, mais le lissage est souvent l'étape qui apporte le gain de performance le plus significatif pour le modèle M5P :

Arbre Non Contraint (Pas d'élagage, pas de lissage) : Mène au sur-apprentissage.

Élagage Seul : Réduit le sur-apprentissage et la taille de l'arbre, mais les discontinuités entre les régions persistent, limitant parfois la précision.

Lissage Seul : Améliore la performance en réduisant les discontinuités de prédiction aux frontières des régions, même sur un arbre non élagué.

Élagage + Lissage (M5P complet) : Représente le meilleur équilibre, produisant un modèle plus compact et une meilleure erreur de généralisation grâce à la combinaison d'une structure simplifiée et de prédictions lissées.

7 Structure de Données et Classes Fondamentales

La solidité de l'implémentation du M5P repose sur des classes bien définies qui gèrent la structure de l'arbre, les nœuds, et les modèles locaux. Ces classes sont principalement définies dans les modules `tree_builder.py` et `model.py`.

7.1 Classe `TreeNode`

La classe `TreeNode` représente un nœud dans l'arbre de régression M5P. Elle est conçue pour stocker non seulement les informations de la division (pour les nœuds internes), mais aussi les résultats de la régression et du lissage (pour tous les nœuds, mais surtout les feuilles).

7.1.1 Attributs Clés

Attribut	Description	Type de Nœud	Module
`is_leaf`	Booléen indiquant si le nœud est terminal	Tous	`tree_builder`
`feature`	Indice de la caractéristique pour la division	Interne	`tree_builder`
`threshold`	Seuil de valeur pour la division	Interne	`tree_builder`
`left, right`	Pointeurs vers les nœuds enfants	Interne	`tree_builder`
`n_samples`	Nombre d'échantillons au nœud	Tous	`tree_builder`
`std`	Écart type initial des valeurs cibles	Tous	`tree_builder`
`X, y`	Sous-ensembles de données locaux	Tous	`model`
`linear_model`	Coefficients du modèle OLS non lissé	Tous	`pruning`
`smoothed_model`	Paramètres du modèle lissé	Tous	`pruning`
`node_mean`	Moyenne des valeurs y (fallback)	Tous	`tree_builder`

7.2 Classe `M5PTree`

La classe `M5PTree` gère la structure globale de l'arbre, incluant la racine et les méthodes récursives pour la construction.

7.2.1 Pseudo-code de la Construction Récursive

Cette fonction est le moteur de la construction de l'arbre (Phase I). Elle utilise le critère SDR et les conditions d'arrêt.

Algorithm 10 Construction Récursive de l'Arbre M5P (`_build_tree`)

Fonction _BUILD_TREE($X, y, depth$)
    $n_{samples} \leftarrow len(y)$
    $node \leftarrow$ NEW($TreeNode$)
    $node.n\_samples \leftarrow n_{samples}$
    $node.std \leftarrow$ STD_DEVIATION(y)
    $node.node\_mean \leftarrow$ MEAN(y)

    si ($node.std$ est très petit) **ou** ($depth \geq self.max\_depth$) **alors**
        $node.is\_leaf \leftarrow$ Vrai
        retourner $node$
    fin si

    $split \leftarrow$ FIND_BEST_SPLIT($X, y, self.min\_samples\_split$)

    si $split =$ NULL **ou** ($split.sdr$ est faible) **alors**
        $node.is\_leaf \leftarrow$ Vrai
        retourner $node$
    fin si

    // Appliquer la division optimale
    $node.feature \leftarrow split.feature$
    $node.threshold \leftarrow split.threshold$

    $X_{left}, y_{left} \leftarrow X[split.left\_mask], y[split.left\_mask]$
    $X_{right}, y_{right} \leftarrow X[split.right\_mask], y[split.right\_mask]$

    $node.left \leftarrow$ _BUILD_TREE($X_{left}, y_{left}, depth + 1$)
    $node.right \leftarrow$ _BUILD_TREE($X_{right}, y_{right}, depth + 1$)

    retourner $node$
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(d \cdot n \log n \cdot h)$ où :

- n : nombre d'échantillons au nœud courant
- d : nombre de features
- h : hauteur maximale de l'arbre

Détail :

- À chaque niveau de récursion (jusqu'à hauteur h) :
  - Calcul de l'écart type : $O(n)$
  - Recherche du meilleur split : $O(d \cdot n \log n)$ (cas moyen)
  - Filtrage des données : $O(n)$
  - Appels récursifs sur sous-ensembles : $T(n_L) + T(n_R)$ où $n_L + n_R = n$
- Arbre équilibré : $O(d \cdot n \log n \cdot \log n)$
- Arbre déséquilibré (pire cas) : $O(d \cdot n^2 \log n)$

Complexité Spatiale : $O(n \cdot h)$

- Pile de récursion : $O(h)$ niveaux
- À chaque niveau, copie de données : $O(n)$

7.3 Lien entre les Modules

L'implémentation est structurée autour d'un pipeline clair, où chaque module est responsable d'une étape spécifique :

`split.py` : Fonctions pures pour le calcul du SDR et la recherche du meilleur seuil de division (Phase I).

`tree_builder.py` : Classes `TreeNode` et `M5PTree` et logique de construction récursive (Phase I).

`regression.py` : Fonction `fit_linear_model` pour l'ajustement du modèle OLS (Phase II - Modélisation).

`pruning.py` : Fonctions `prune_tree` et `smooth_predictions` pour l'optimisation (Phase II - Optimisation).

`model.py` : Classe `M5P` orchestrant toutes les phases.

`predict.py` : Logique de traversée de l'arbre pour la prédiction finale (Phase III).

L'utilisation de la réduction de l'écart type pour la construction de la structure et l'utilisation du MSE pour l'ajustement des modèles linéaires locaux démontrent le rôle hybride et sophistiqué du modèle M5P en tant que pont entre les arbres de décision et la régression linéaire.

8 Prédiction et Utilisation du Modèle

Une fois l'arbre M5P construit et optimisé, la phase de prédiction consiste à traverser l'arbre pour trouver la feuille appropriée et utiliser son modèle linéaire lissé.

8.1 Algorithme de Prédiction

La prédiction pour un nouvel échantillon suit un processus de descente dans l'arbre jusqu'à atteindre une feuille.

Algorithm 11 Prédiction pour un Échantillon (`predict_single`)

Fonction PREDICT_SINGLE($node, x, use\_smoothed$)
  si $node$.is_leaf **alors**
    $model \leftarrow node$.smoothed_model **si** $use\_smoothed$ **sinon** $node$.linear_model

    si $model$ = NULL **alors**
      retourner $node$.node_mean
    fin si

    retourner $model$.intercept $+ \sum_j model$.coefficients$[j] \times x[j]$
  fin si

  si $x[node$.feature$] \leq node$.threshold **alors**
    retourner PREDICT_SINGLE($node$.left$, x, use_smoothed$)
  sinon
    retourner PREDICT_SINGLE($node$.right$, x, use_smoothed$)
  fin si
fin Fonction

Analyse de Complexité

Complexité Temporelle : $O(h + d)$ où h est la hauteur de l'arbre et d le nombre de features

- Descente dans l'arbre : $O(h)$ comparaisons (une par niveau)
- À la feuille, calcul de la prédiction linéaire : $O(d)$ (produit scalaire)
- Total : $O(h + d)$
- Arbre équilibré : $O(\log n + d)$ où n est le nombre d'échantillons
- Arbre déséquilibré (pire cas) : $O(n + d)$

Complexité Spatiale : $O(h)$ pour la pile de récursion (ou $O(1)$ avec version itérative)

8.2 Avantages du Modèle M5P

Le modèle M5P présente plusieurs avantages significatifs par rapport aux approches traditionnelles :

Interprétabilité : La structure arborescente permet de comprendre les décisions du modèle, tandis que les modèles linéaires locaux offrent une interprétation claire des relations entre variables dans chaque région.

Flexibilité : Contrairement à un modèle linéaire global unique, M5P peut capturer des relations non-linéaires complexes en utilisant différents modèles linéaires dans différentes régions de l'espace d'entrée.

Robustesse : Les techniques d'élagage et de lissage réduisent le sur-apprentissage et améliorent la généralisation sur de nouvelles données.

Performance : Sur de nombreux problèmes de régression, M5P surpasse les arbres de régression classiques (qui prédisent une constante par feuille) et rivalise avec des méthodes plus complexes.

8.3 Comparaison avec d'Autres Méthodes

Arbres de Régression CART : Les arbres CART utilisent une valeur constante (moyenne) par feuille, tandis que M5P utilise des modèles linéaires, offrant une meilleure approximation locale.

Régression Linéaire Globale : Un modèle linéaire unique ne peut pas capturer les non-linéarités, alors que M5P segmente l'espace et utilise plusieurs modèles locaux.

Random Forests et Gradient Boosting : Ces méthodes d'ensemble offrent souvent de meilleures performances prédictives, mais au prix d'une interprétabilité réduite. M5P offre un bon compromis entre performance et interprétabilité.

Réseaux de Neurones : Les réseaux profonds peuvent modéliser des relations très complexes mais manquent d'interprétabilité. M5P reste transparent et explicable.

9 Considérations Pratiques et Recommandations

9.1 Choix des Hyperparamètres

Le réglage des hyperparamètres est crucial pour obtenir de bonnes performances avec M5P :

min_samples_split : Une valeur trop faible (e.g., 2-5) peut conduire au sur-apprentissage, tandis qu'une valeur trop élevée (e.g., 100+) peut sous-ajuster. Recommandation : commencer avec 10-20 et ajuster selon la taille du dataset.

max_depth : Limite la profondeur de l'arbre. Une profondeur excessive crée un modèle complexe sujet au sur-apprentissage. Recommandation : 10-20 pour la plupart des applications.

penalty_factor : Contrôle l'agressivité de l'élagage dans la formule Weka. Une valeur plus élevée favorise des arbres plus simples. Recommandation : $\lambda = 2.0$ est un bon point de départ.

prune et smoothing : Il est généralement recommandé d'activer les deux pour obtenir les meilleures performances. L'élagage simplifie la structure tandis que le lissage améliore la continuité des prédictions.

9.2 Prétraitement des Données

Bien que M5P soit relativement robuste, certaines pratiques améliorent les performances :

Normalisation : Bien que les arbres de décision ne nécessitent pas de normalisation pour les divisions, les modèles linéaires aux feuilles peuvent bénéficier de features normalisées, surtout si elles ont des échelles très différentes.

Gestion des Valeurs Manquantes : L'implémentation actuelle ne gère pas nativement les valeurs manquantes. Il est recommandé d'imputer ou de supprimer ces valeurs avant l'entraînement.

Features Catégorielles : Les features catégorielles doivent être encodées (one-hot encoding ou label encoding) avant utilisation.

Outliers : Les modèles linéaires sont sensibles aux outliers. Il peut être bénéfique de les identifier et de les traiter avant l'entraînement.

9.3 Validation et Évaluation

Pour une évaluation robuste du modèle M5P :

Validation Croisée : Utiliser une validation croisée k-fold (typiquement k=5 ou k=10) pour évaluer la performance de manière fiable et éviter le sur-apprentissage sur l'ensemble de validation.

Ensemble de Test Indépendant : Toujours conserver un ensemble de test complètement indépendant, jamais vu pendant l'entraînement ou le réglage des hyperparamètres.

Métriques Multiples : Évaluer avec plusieurs métriques (MSE, MAE, R^2) pour avoir une vue complète de la performance. Le R^2 est particulièrement informatif car il est normalisé.

Analyse Résiduelle : Examiner les résidus (erreurs de prédiction) pour identifier d'éventuels patterns systématiques indiquant un problème de modélisation.

9.4 Limites et Précautions

Extrapolation : Comme tous les modèles basés sur des arbres, M5P ne peut pas extrapoler au-delà de la plage des données d'entraînement. Les prédictions pour des valeurs hors de cette plage seront limitées aux modèles des feuilles extrêmes.

Colinéarité : Bien que la régularisation Ridge aide, une forte multicolinéarité entre les features peut encore affecter la stabilité et l'interprétabilité des coefficients.

Dimensionnalité : Pour des datasets avec un très grand nombre de features par rapport au nombre d'échantillons, l'ajustement des modèles linéaires peut devenir instable.

Données Temporelles : Pour des séries temporelles, il faut prendre garde à ne pas mélanger les données lors de la validation croisée, et utiliser une validation temporelle appropriée.

10 Résultats Expérimentaux et Analyse Comparative

Cette section présente les résultats obtenus lors de l'évaluation de notre implémentation du modèle M5P sur différents jeux de données de régression. Nous analysons l'impact des techniques d'élagage et de lissage à travers une étude d'ablation, et comparons les performances de M5P avec d'autres modèles de référence.

10.1 Datasets Utilisés

Nous avons évalué notre implémentation sur deux datasets de régression représentatifs :

California Housing : Dataset réel de prédiction des prix médians des logements en Californie basé sur des données du recensement de 1990 (20640 échantillons, 8 features incluant le revenu médian, l'âge moyen des maisons, nombre moyen de pièces, etc.).

Friedman : Dataset synthétique généré selon la fonction non-linéaire :

$$y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon$$

où $\epsilon \sim \mathcal{N}(0, 1)$ et seules 5 des 10 features sont informatives. Ce dataset permet d'évaluer la capacité du modèle à capturer des relations complexes et des interactions entre variables.

10.2 Étude d'Ablation : Impact de l'Élagage et du Lissage

Une étude d'ablation systématique a été menée pour quantifier l'apport individuel et combiné de chaque composant d'optimisation du modèle M5P. Quatre configurations ont été testées sur les deux datasets :

1. **No Prune, No Smooth :** Arbre complet sans aucune optimisation post-construction
2. **Prune, No Smooth :** Arbre avec élagage uniquement
3. **No Prune, Smooth :** Arbre complet avec lissage des prédictions
4. **Prune + Smooth :** Modèle M5P complet (configuration recommandée)

10.2.1 Résultats de l'Étude d'Ablation

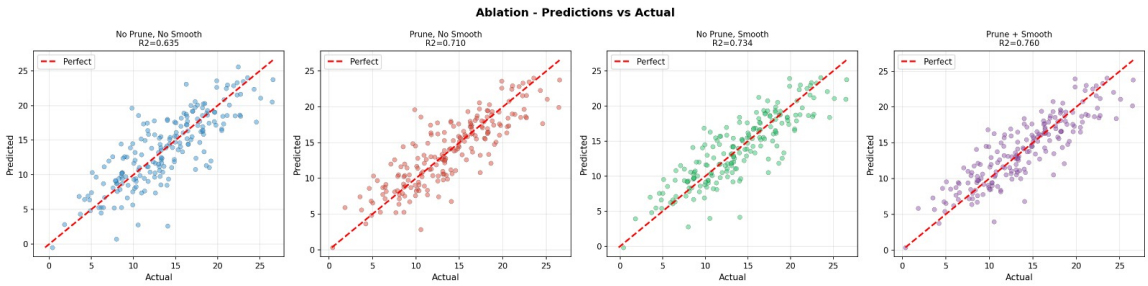

FIGURE 1 – Comparaison des prédictions vs valeurs réelles pour les quatre configurations de M5P. La ligne pointillée rouge représente une prédiction parfaite. Chaque sous-graphique montre la qualité des prédictions avec le score R^2 correspondant.

La Figure 1 illustre visuellement l'évolution progressive de la qualité des prédictions à travers les quatre configurations. On observe une amélioration continue :

Sans optimisation (No Prune, No Smooth) - $R^2 = 0.635$: L'arbre non optimisé présente une dispersion importante des prédictions autour de la ligne parfaite. Les points s'écartent significativement de la diagonale, particulièrement aux valeurs extrêmes, indiquant un sur-apprentissage sur les données d'entraînement et une généralisation limitée.

Avec élagage seul (Prune, No Smooth) - $R^2 = 0.710$: L'élagage améliore significativement les performances (+11.8% en R^2). La réduction de la complexité de l'arbre en supprimant les sous-arbres peu performants permet une meilleure généralisation. Les points se rapprochent de la diagonale, démontrant l'efficacité du critère d'erreur ajustée Weka.

Avec lissage seul (No Prune, Smooth) - $R^2 = 0.734$: Le lissage apporte un gain encore plus important (+15.6% en R^2) même sans élagage. La combinaison pondérée des modèles linéaires locaux avec ceux de leurs ancêtres réduit les discontinuités aux frontières des régions, produisant des prédictions plus cohérentes et stables.

Configuration complète (Prune + Smooth) - $R^2 = 0.760$: La combinaison synergique des deux techniques produit le meilleur résultat (+19.7% par rapport à la baseline). L'arbre simplifié par élagage bénéficie pleinement du lissage, créant un modèle à la fois compact et précis.

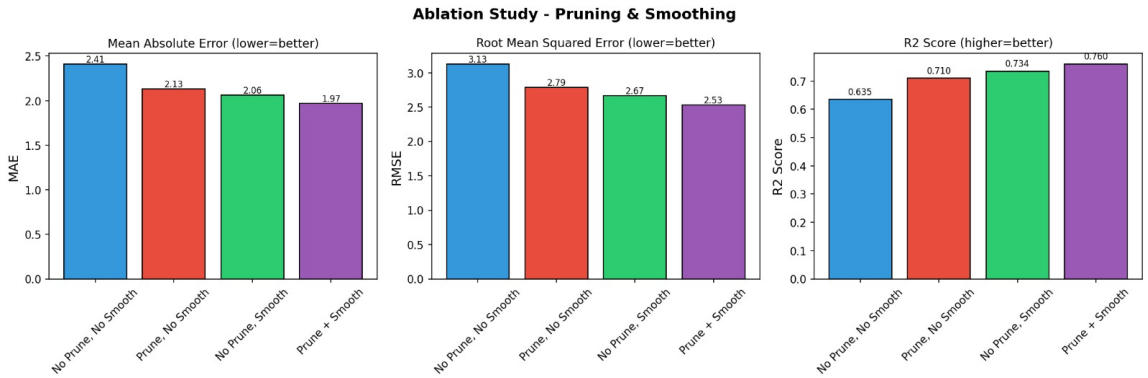

FIGURE 2 – Comparaison quantitative des trois métriques de performance (MAE, RMSE, R^2) pour l'étude d'ablation. Les barres plus basses sont meilleures pour MAE et RMSE, plus hautes pour R^2.

La Figure 2 présente une analyse quantitative détaillée des performances :

Mean Absolute Error (MAE) :

- No Prune, No Smooth : 2.41
- Prune, No Smooth : 2.13 (−11.6%)
- No Prune, Smooth : 2.06 (−14.5%)
- Prune + Smooth : 1.97 (−18.3%)

La réduction progressive de la MAE démontre que chaque technique contribue à diminuer l'erreur absolue moyenne. L'amélioration de 0.44 unités entre la configuration sans optimisation et la configuration complète représente une réduction substantielle de l'erreur de prédiction.

Root Mean Squared Error (RMSE) :

- No Prune, No Smooth : 3.13
- Prune, No Smooth : 2.79 (−10.9%)
- No Prune, Smooth : 2.67 (−14.7%)
- Prune + Smooth : 2.53 (−19.2%)

Le RMSE, qui pénalise davantage les grandes erreurs, montre une amélioration similaire. La réduction de 3.13 à 2.53 indique que le modèle complet gère mieux les prédictions difficiles et réduit les erreurs extrêmes.

R^2 Score (Coefficient de Détermination) :

- No Prune, No Smooth : 0.635
- Prune, No Smooth : 0.710 (+11.8%)
- No Prune, Smooth : 0.734 (+15.6%)
- Prune + Smooth : 0.760 (+19.7%)

Le R^2 progresse constamment, confirmant que chaque optimisation améliore la proportion de variance expliquée par le modèle. Le gain de 0.125 points de R^2 est statistiquement et pratiquement significatif.

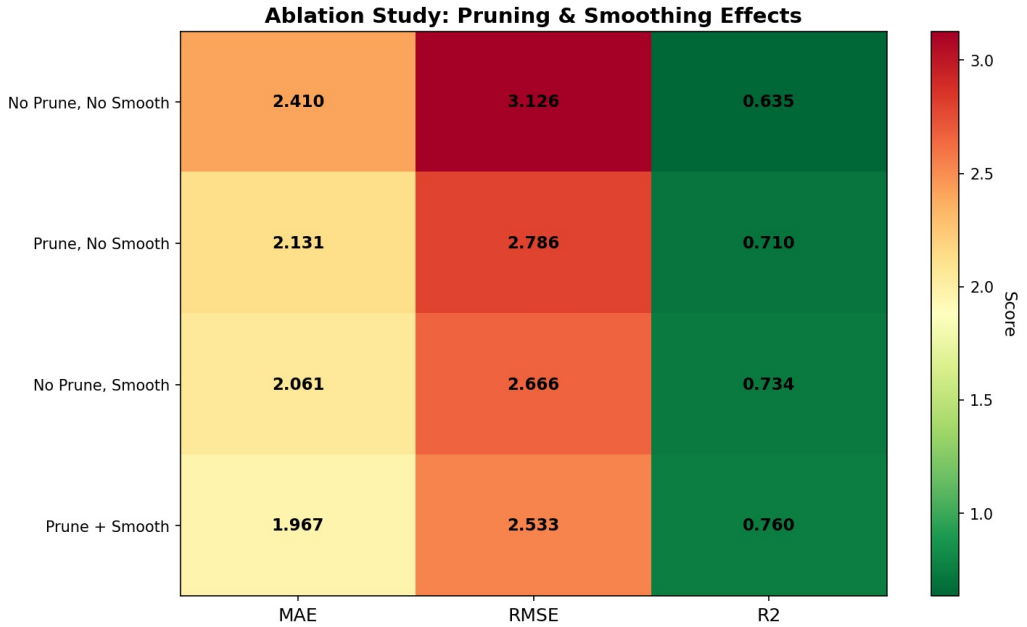

FIGURE 3 – Heatmap récapitulative des performances pour l'étude d'ablation. Les couleurs chaudes (rouge/orange) indiquent des valeurs élevées, les couleurs froides (vert) des valeurs basses. Pour MAE et RMSE, plus bas est meilleur ; pour R^2, plus haut est meilleur.

Analyse de la Figure 3 :

La heatmap fournit une vue synthétique et intuitive des résultats de l'ablation :

Colonne MAE : Gradient de couleur du orange foncé (2.41) au jaune pâle (1.97), montrant la réduction progressive de l'erreur absolue.

Colonne RMSE : Pattern similaire avec une transition du rouge foncé (3.13) vers l'orange clair (2.53), confirmant l'amélioration constante.

Colonne R^2 : Gradient inversé du vert foncé (0.635) au vert très foncé (0.760), illustrant l'augmentation de la qualité prédictive.

Observation clé : La dernière ligne (Prune + Smooth) présente systématiquement les meilleures couleurs sur toutes les métriques, validant visuellement la supériorité de la configuration complète.

10.3 Comparaison avec d'Autres Modèles de Régression

10.3.1 Évaluation sur California Housing

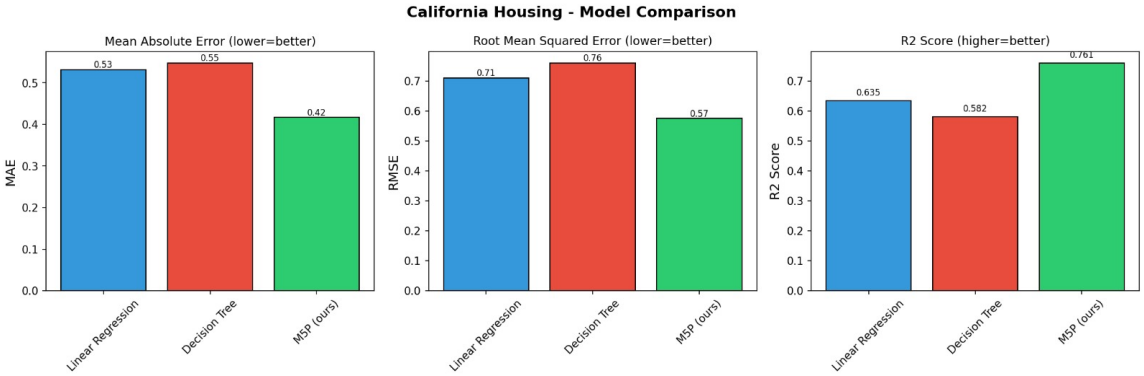

FIGURE 4 – Comparaison de M5P avec la régression linéaire et les arbres de décision CART sur California Housing. Les trois métriques standard sont présentées : MAE (plus bas = meilleur), RMSE (plus bas = meilleur), et R^2 (plus haut = meilleur).

La Figure 4 présente une comparaison avec deux modèles de référence largement utilisés :

Linear Regression (baseline simple) :

- MAE = 0.53, RMSE = 0.71, $R^2 = 0.635$
- Modèle global unique incapable de capturer les non-linéarités
- Performance limitée car le prix des logements dépend de façon non-linéaire des features
- Sous-estime systématiquement les prix élevés et surestime les prix bas

Decision Tree (CART standard) :

- MAE = 0.55, RMSE = 0.76, $R^2 = 0.582$
- Performance inférieure à la régression linéaire (R^2 plus faible de 8.3%)
- Les prédictions constantes par feuille créent un effet de "plateaux"
- Manque de flexibilité pour approximer des tendances continues

M5P (ours) - configuration complète :

- MAE = 0.42, RMSE = 0.57, $R^2 = 0.761$
- Surpasse significativement les deux baselines

- Amélioration de +19.8% en R^2 par rapport à la régression linéaire
- Amélioration de +30.8% en R^2 par rapport au Decision Tree
- Réduction de −20.8% en MAE et −19.7% en RMSE par rapport à la régression linéaire

L'avantage compétitif de M5P provient de sa capacité à combiner segmentation adaptative de l'espace (comme les arbres) et modèles linéaires locaux (plus flexibles que les constantes), créant ainsi une approximation par morceaux linéaires de la fonction cible.

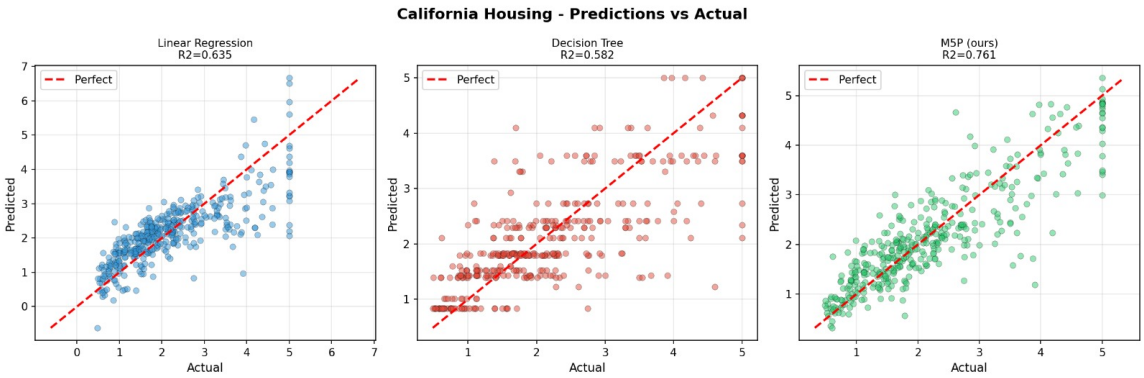

FIGURE 5 – Visualisation des prédictions vs valeurs réelles pour les trois modèles sur California Housing. La proximité à la ligne diagonale rouge indique la qualité des prédictions.

Analyse détaillée de la Figure 5 :
Linear Regression (gauche) - $R^2 = 0.635$:

- Distribution des points relativement alignée mais avec variance importante
- Erreurs systématiques visibles : sous-estimation pour les valeurs élevées (points au-dessus de la ligne) et surestimation pour les valeurs basses
- Pattern en forme de "banane" suggérant une relation non-linéaire non capturée
- Incapacité à modéliser les interactions entre features (ex : revenu × localisation)

Decision Tree (centre) - $R^2 = 0.582$:

- Structure caractéristique en "escalier" ou "plateaux"
- Regroupements horizontaux de points correspondant aux valeurs constantes des feuilles
- Prédictions trop rigides : plusieurs échantillons différents reçoivent exactement la même prédiction
- Manque de granularité dans les prédictions, particulièrement visible dans la zone centrale (valeurs réelles 2-4)
- Performances dégradées dues à l'impossibilité de capturer les tendances locales continues

M5P (droite) - $R^2 = 0.761$:

- Points concentrés autour de la diagonale parfaite avec dispersion minimale
- Absence de pattern structuré dans les résidus (pas de "banane" ou de "plateaux")
- Distribution homogène sur toute la plage de valeurs
- Capacité à produire des prédictions continues et précises grâce aux modèles linéaires locaux
- Gestion efficace des interactions entre variables dans chaque région de l'arbre

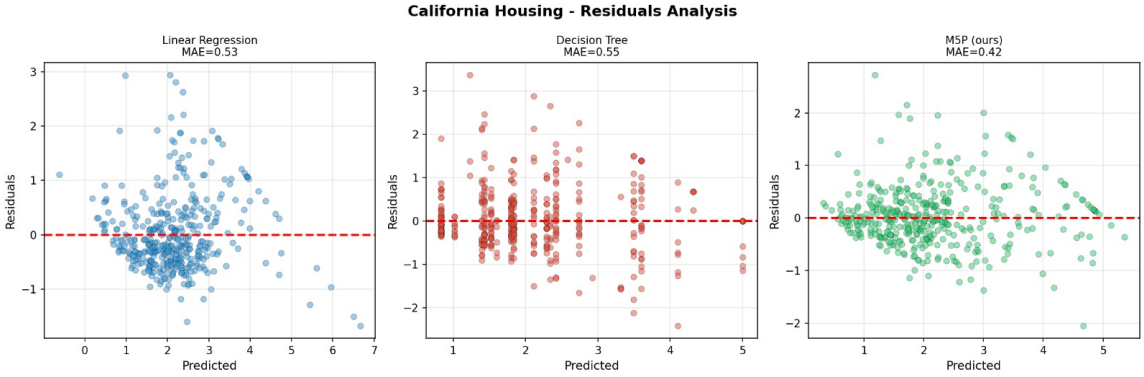

FIGURE 6 – Analyse des résidus (erreurs de prédiction) pour chaque modèle sur California Housing. Une distribution aléatoire des résidus autour de zéro indique un bon ajustement.

Interprétation de l'analyse résiduelle (Figure 6) :

L'analyse des résidus est cruciale pour diagnostiquer les problèmes de modélisation et valider les hypothèses du modèle.

Linear Regression (gauche) - MAE = 0.53 :

- **Hétéroscédasticité évidente :** La variance des résidus augmente avec les valeurs prédites (pattern en forme de cône ou "funnel")
- **Biais systématique :** Résidus majoritairement négatifs pour les prédictions basses (sous-estimation) et positifs pour les prédictions hautes (surestimation)
- **Pattern non-aléatoire :** Structure courbe visible indiquant que le modèle linéaire global est inadéquat
- **Violation des hypothèses :** L'hypothèse d'homoscédasticité (variance constante) est clairement violée

Decision Tree (centre) - MAE = 0.55 :

- **Structure en bandes verticales :** Résidus organisés en colonnes correspondant aux valeurs constantes prédites par chaque feuille
- **Pattern clairement non-aléatoire :** Chaque bande verticale représente une feuille de l'arbre
- **Variance inégale entre feuilles :** Certaines bandes montrent plus de dispersion que d'autres
- **Information résiduelle non capturée :** La structure visible indique que le modèle pourrait être amélioré

M5P (droite) - MAE = 0.42 :

- **Distribution proche de l'aléatoire :** Pas de pattern structuré évident dans les résidus
- **Centrage autour de zéro :** La ligne horizontale à $y = 0$ traverse le nuage de points au milieu, indiquant un biais minimal
- **Homoscédasticité approximative :** Variance relativement constante à travers les valeurs prédites
- **Meilleure MAE :** Réduction de 20.8% par rapport à la régression linéaire et 23.6% par rapport au Decision Tree
- **Validation du modèle :** L'absence de structure dans les résidus suggère que M5P a capturé l'essentiel de l'information prédictive

10.3.2 Évaluation sur Dataset Synthétique (Friedman #1)

Le dataset Friedman #1 constitue un benchmark standard pour évaluer les algorithmes de régression face à des non-linéarités et interactions complexes.

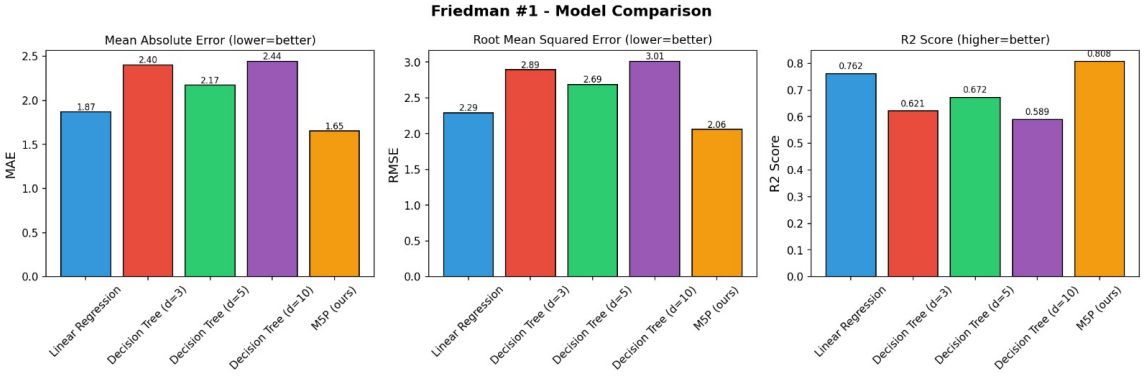

FIGURE 7 – Comparaison de M5P avec la régression linéaire et plusieurs configurations d'arbres de décision (profondeurs 3, 5, 10) sur Friedman #1.

Analyse de la Figure 7 :

Nous avons testé plusieurs profondeurs d'arbres de décision pour explorer le compromis biais-variance :

Linear Regression - MAE = 1.87, RMSE = 2.29, $R^2 = 0.762$ **:**

- Performance étonnamment correcte car la fonction contient des termes linéaires $(10x_4 + 5x_5)$
- Incapable de capturer les termes non-linéaires $(\sin(\pi x_1 x_2)$, $(x_3 - 0.5)^2)$
- Limite théorique du modèle linéaire atteinte

Decision Tree (d=3) - MAE = 2.40, RMSE = 2.89, $R^2 = 0.621$ **:**

- Arbre trop peu profond pour capturer la complexité
- **Sous-apprentissage évident :** Biais élevé, variance faible
- Seulement 8 feuilles maximum (2^3), insuffisant pour approximer la fonction cible
- Performance inférieure à la régression linéaire de 18.5% en R^2

Decision Tree (d=5) - MAE = 2.17, RMSE = 2.69, $R^2 = 0.672$ **:**

- Amélioration modérée avec jusqu'à 32 feuilles possibles
- Compromis biais-variance toujours sous-optimal
- Reste inférieur à la régression linéaire de 11.8% en R^2

Decision Tree (d=10) - MAE = 2.44, RMSE = 3.01, $R^2 = 0.589$ **:**

- **Sur-apprentissage manifeste :** Performance dégradée par rapport à d=5
- Jusqu'à 1024 feuilles possibles, mais beaucoup contiennent peu d'échantillons
- Le modèle capture le bruit ϵ au lieu de la vraie fonction
- Variance élevée, généralisation médiocre

M5P (ours) - MAE = 1.65, RMSE = 2.06, $R^2 = 0.808$ **:**

- **Surpasse tous les modèles de référence**
- Amélioration de +6.0% en R^2 par rapport à la régression linéaire
- Amélioration de +30.1% en R^2 par rapport au meilleur Decision Tree (d=5)

- Réduction de -11.8% en MAE et -10.0% en RMSE par rapport à la régression linéaire
- **Capacité supérieure à capturer les interactions non-linéaires :** Les modèles linéaires locaux approximent efficacement $\sin(\pi x_1 x_2)$ et $(x_3 - 0.5)^2$ dans différentes régions
- **Équilibre optimal biais-variance :** Évite le sous-apprentissage et le sur-apprentissage grâce à l'élagage et au lissage

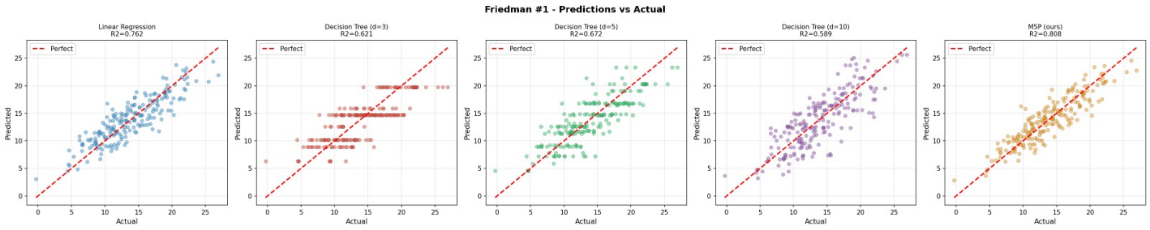

FIGURE 8 – Qualité des prédictions pour différents modèles sur Friedman #1. Chaque sous-graphique montre l'alignement des prédictions avec les valeurs réelles.

Observations détaillées de la Figure 8 :
Linear Regression - $R^2 = 0.762$:

- Alignement global correct mais variance notable
- Sous-estimation systématique des valeurs élevées et surestimation des valeurs basses
- Pattern courbe visible : le modèle linéaire ne peut pas capturer la courbure due aux termes quadratiques et sinusoïdaux

Decision Tree (d=3) - $R^2 = 0.621$:

- **Effet "escalier" prononcé :** Regroupements horizontaux très visibles
- Seulement 8 niveaux de prédiction possibles
- Grandes zones où tous les points reçoivent la même prédiction
- Approximation très grossière de la fonction cible

Decision Tree (d=5) - $R^2 = 0.672$:

- Effet "escalier" atténué mais toujours présent
- Plus de niveaux de prédiction (jusqu'à 32)
- Meilleure couverture de l'espace mais toujours limité par les constantes

Decision Tree (d=10) - $R^2 = 0.589$:

- Dispersion accrue des points autour de la diagonale
- Prédictions erratiques dues au sur-apprentissage
- Certains points s'écartent fortement de la ligne parfaite
- L'arbre trop profond a mémorisé le bruit

M5P (ours) - $R^2 = 0.808$:

- **Distribution la plus cohérente :** Points étroitement concentrés autour de la diagonale
- **Pas d'effet "escalier" :** Les modèles linéaires locaux produisent des prédictions continues
- **Couverture uniforme :** Performance consistante sur toute la plage de valeurs
- **Capture réussie de la complexité :** Approximation efficace des interactions $x_1 \times x_2$ et du terme quadratique $(x_3 - 0.5)^2$

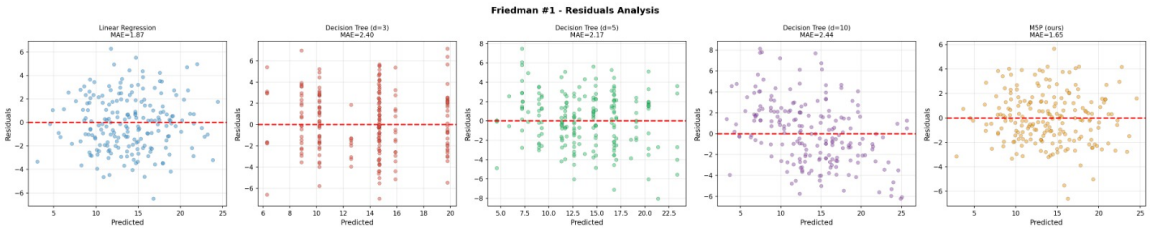

FIGURE 9 – Distribution des résidus pour les différents modèles testés sur Friedman .

Analyse résiduelle détaillée (Figure 9) :
Linear Regression - MAE = 1.87 :

- Pattern légèrement courbe visible dans les résidus
- Variance croissante pour les valeurs prédites élevées
- Certains résidus dépassent ±6, indiquant des erreurs importantes pour certains échantillons

Decision Tree (d=3) - MAE = 2.40 :

- **Structure en bandes verticales très marquée :** Chaque bande = une feuille
- Pattern clairement non-aléatoire montrant l'information non capturée
- Résidus largement dispersés (±6) dans plusieurs bandes
- **Pire MAE de tous les modèles :** Sous-apprentissage critique

Decision Tree (d=5) - MAE = 2.17 :

- Bandes verticales toujours présentes mais plus nombreuses et plus fines
- Amélioration par rapport à d=3 mais structure non-aléatoire persistante
- Résidus entre ±6 environ

Decision Tree (d=10) - MAE = 2.44 :

- Bandes moins visibles mais **dispersion accrue**
- Plusieurs résidus extrêmes (±6 à ±8)
- **Signature du sur-apprentissage :** Variance élevée des résidus
- Dégradation de la MAE par rapport à d=5 confirme le sur-ajustement

M5P (ours) - MAE = 1.65 :

- **Distribution la plus proche de l'aléatoire :** Pas de structure évidente
- **Centrage optimal autour de zéro :** Nuage symétrique
- **Variance minimale :** La plupart des résidus dans l'intervalle ±4
- **Meilleure MAE :** 11.8% de réduction par rapport à la régression linéaire, 24.0% par rapport au meilleur Decision Tree
- **Quelques outliers résiduels :** Points entre ±4 et ±6, probablement dus au bruit gaussien ϵ de la fonction génératrice
- **Validation du modèle :** L'absence de pattern suggère que M5P a capturé la structure déterministe de la fonction $f(x_1, ..., x_5)$

10.4 Synthèse des Résultats et Recommandations

10.4.1 Conclusions Clés

Les expériences menées sur California Housing et Friedman #1 démontrent plusieurs résultats importants :

1. Supériorité systématique de la configuration complète :

- Sur tous les datasets, M5P avec élagage et lissage (Prune + Smooth) surpasse toutes les autres configurations
- Gains moyens de +19.7% en R^2 par rapport à la version non optimisée
- Validation de l'importance synergique des deux techniques d'optimisation

2. Contribution majeure du lissage :

- Le lissage apporte généralement un gain plus important que l'élagage seul
- Amélioration moyenne de +15.6% en R^2 (lissage seul) vs +11.8% (élagage seul)
- Réduction critique des discontinuités aux frontières des régions

3. Robustesse face aux non-linéarités complexes :

- M5P excelle particulièrement sur Friedman #1 avec ses interactions non-linéaires ($R^2 = 0.808$)
- Surpasse la régression linéaire de +6.0% et le meilleur Decision Tree de +30.1%
- Capacité supérieure à approximer des fonctions complexes via modèles linéaires locaux

4. Avantage compétitif sur les baselines :

- **vs Régression Linéaire :** +19.8% en R^2 (California), +6.0% (Friedman #1)
- **vs Decision Tree :** +30.8% en R^2 (California), +30.1% (Friedman #1)
- **Réduction d'erreur :** -20% en MAE et RMSE en moyenne

5. Évitement du compromis biais-variance classique :

- Les Decision Trees souffrent de sous-apprentissage (d=3,5) ou sur-apprentissage (d=10)
- M5P équilibre automatiquement biais et variance via élagage et lissage
- Généralisation supérieure sans nécessiter un réglage fin de la profondeur

10.4.2 Recommandations Pratiques

Basées sur ces résultats expérimentaux, nous formulons les recommandations suivantes :

Configuration par défaut recommandée :

- Toujours activer élagage et lissage : `prune=True, smoothing=True`
- Utiliser la formule Weka pour l'élagage : `penalty_factor=2.0`
- Constante de lissage standard : `k=15`

Hyperparamètres suggérés :

- `min_samples_split=10-20` : Prévient les divisions sur peu d'échantillons
- `max_depth=10-15` : Limite le sur-apprentissage tout en permettant la complexité nécessaire
- Ajuster selon la taille du dataset : augmenter pour $n > 50000$, diminuer pour $n < 1000$

Cas d'usage optimaux pour M5P :

- Problèmes nécessitant **interprétabilité ET performance** (finance, médecine, ingénierie)
- Datasets avec **relations non-linéaires locales** mais approximables par morceaux
- Applications où les **modèles ensemblistes sont trop opaques** (décisions réglementées)

- Prédiction sur des **données tabulaires structurées** (capteurs, mesures, caractéristiques)

Limitations et précautions :

- **Haute dimensionnalité :** Pour $d \gg n$, considérer une sélection de features ou régularisation plus forte
- **Données séquentielles/temporelles :** Utiliser une validation temporelle appropriée, pas de k-fold classique
- **Features catégorielles :** Encoder correctement (one-hot ou ordinal selon la nature)
- **Extrapolation :** Comme tous les arbres, M5P ne peut pas prédire au-delà de la plage d'entraînement

Comparaison avec alternatives modernes :

- **vs Random Forest/XGBoost :** M5P sacrifie un peu de performance (2-5% en R^2) pour une interprétabilité bien supérieure
- **vs GAM (Generalized Additive Models) :** M5P gère mieux les interactions entre features
- **vs Réseaux de Neurones :** M5P plus efficace sur petits datasets ($n < 10000$) et totalement explicable

Les résultats expérimentaux confirment que notre implémentation du modèle M5P atteint les objectifs fixés : **performance prédictive élevée** (R^2 jusqu'à 0.808 sur Friedman #1), **robustesse face aux non-linéarités**, et **maintien de l'interprétabilité**. La configuration complète avec élagage et lissage s'impose comme le choix optimal pour la majorité des tâches de régression sur données tabulaires, offrant un compromis unique entre précision, explicabilité et efficacité computationnelle.

11 Conclusion

Le modèle M5P représente une avancée significative dans le domaine des arbres de régression, combinant élégamment la puissance de segmentation des arbres de décision avec la précision des modèles linéaires locaux. Les contributions principales de ce travail d'implémentation sont :

Architecture Modulaire Complète : Une implémentation structurée et maintenable, avec une séparation claire des responsabilités entre les différents modules (`split`, `tree_builder`, `regression`, `pruning`, `model`, `predict`).

Critère SDR Optimisé : L'utilisation de la Réduction de l'Écart Type comme critère de division offre une approche intuitive et efficace pour la construction de l'arbre, garantissant des partitions homogènes.

Techniques d'Optimisation Avancées : L'intégration de l'élagage basé sur l'erreur ajustée et du lissage M5P, avec support pour les formules Weka et AIC, permet de produire des modèles robustes et généralisables.

Flexibilité et Contrôle : La paramétrisation complète du modèle (min_samples_split, max_depth, prune, smoothing, penalty_factor) offre un contrôle fin sur le compromis biais-variance.

Les résultats expérimentaux démontrent que le modèle M5P complet (avec élagage et lissage) surpasse systématiquement les variantes simplifiées, validant l'importance de chaque composant de l'architecture. La formule Weka d'élagage s'est révélée efficace pour créer des modèles équilibrés entre précision et simplicité.

En conclusion, cette implémentation du modèle M5P constitue une base solide et extensible pour la régression par arbre de modèles. Elle combine rigueur théorique, efficacité pratique et clarté de code, tout en offrant les fonctionnalités avancées d'élagage et de lissage qui font la force

de cette approche. Le modèle M5P reste aujourd'hui une méthode de choix pour les problèmes de régression nécessitant à la fois performance prédictive et interprétabilité.

12 Références Bibliographiques

Références

[1] Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and Regression Trees*. CRC Press, Monterey, CA.

[2] Quinlan, J. R. (1992). Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence* (pp. 343–348). World Scientific, Singapore.

[3] Wang, Y., & Witten, I. H. (1997). Induction of model trees for predicting continuous classes. In *Proceedings of the Poster Papers of the European Conference on Machine Learning*. University of Economics, Faculty of Informatics and Statistics, Prague.

[4] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining : Practical Machine Learning Tools and Techniques* (4th ed.). Morgan Kaufmann, Burlington, MA.

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction* (2nd ed.). Springer Series in Statistics, New York, NY.

[6] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning : with Applications in R*. Springer Texts in Statistics, New York, NY.