Université Abdelmalek Essaadi

Faculté des Sciences et Techniques de Tanger

Année académique 2024-2025

$1^{ére}$année Master AISD

Rapport de Mini Projet

La détection de la maladie de Parkinson Via Les techniques d'apprentissage supervisé : Classification

Machine Learning

Réalisé par:

- DOHA KARIM
- SOUFIANE HBICH
- AYMANE RIHANE

Encadré par :

- Pr. M'hamed AIT KBIR

TABLE DES MATIERES

Liste des figures

Introduction

L'étude s'appuie sur un jeu de données spécifique lié à la détection de la maladie de Parkinson. Ce dataset contient des enregistrements détaillés, regroupant à la fois des caractéristiques biologiques et comportementales des patients. Ces caractéristiques incluent notamment des mesures issues de tests vocaux, des évaluations biométriques, ou d'autres paramètres cliniques permettant de différencier les individus atteints de la maladie de Parkinson de ceux en bonne santé.

Le dataset offre une richesse d'informations, mais également des défis en termes de dimensionnalité et de redondance des caractéristiques. Ces aspects nécessitent une attention particulière dans le cadre du prétraitement des données, de la sélection des caractéristiques et de la réduction de leur dimensionnalité, afin d'améliorer les performances des modèles de machine learning.

En exploitant ce dataset, plusieurs approches ont été mises en œuvre pour évaluer et comparer l'efficacité des techniques utilisées dans la détection de cette maladie. Ces analyses visent à tirer parti des données disponibles tout en optimisant la précision des modèles prédictifs.

CHAPITRE I : PRESENTATION DU DATASET

1. Origine et Contexte

Le jeu de données utilisé provient du *UCI Machine Learning Repository* et a été soumis le 11 avril 2018. Il contient des données vocales collectées auprès de 188 patients atteints de la maladie de Parkinson (107 hommes et 81 femmes, âgés de 33 à 87 ans, moyenne d'âge 65.1±10.9) ainsi que de 64 individus sains (23 hommes et 41 femmes, âgés de 41 à 82 ans, moyenne d'âge 61.1±8.9). Les données ont été recueillies au Département de Neurologie de la Faculté de Médecine Cerrahpaşa, Université d'Istanbul.

2. Caractéristiques des Données

- **Type** : Multivarié
- **Instances** : 756
- **Attributs** : 754, avec des types de données réels ou entiers.
- **Tâche associée** : Classification

3. Méthodes de Collecte

Les enregistrements vocaux ont été effectués avec un microphone réglé à 44,1 kHz. Les participants ont prononcé le son /a/ de manière soutenue trois fois, après un examen médical. Ces données acoustiques permettent d'extraire des informations cliniques pour évaluer la maladie.

4. Description des Caractéristiques Vocales

Les caractéristiques incluent :

- **Fréquence fondamentale** : Indicateur de la fréquence de vibration des cordes vocales.
- **Jitter et Shimmer** : Mesures des variations de fréquence et d'amplitude dans les signaux vocaux, respectivement.
- **MFCC (Mel Frequency Cepstral Coefficients)** : Paramètres qui capturent l'enveloppe spectrale du son.

5. Taille et Classes

Les données se répartissent en deux classes : *malade* et *sain*. Elles sont utilisées pour construire des modèles capables de différencier les individus en fonction de leurs caractéristiques acoustiques.

6. Analyse Exploratoire

Une première analyse des données révèle une répartition déséquilibrée des classes, caractérisée par un fort déséquilibre entre les échantillons de patients atteints de la maladie de Parkinson et les individus sains.

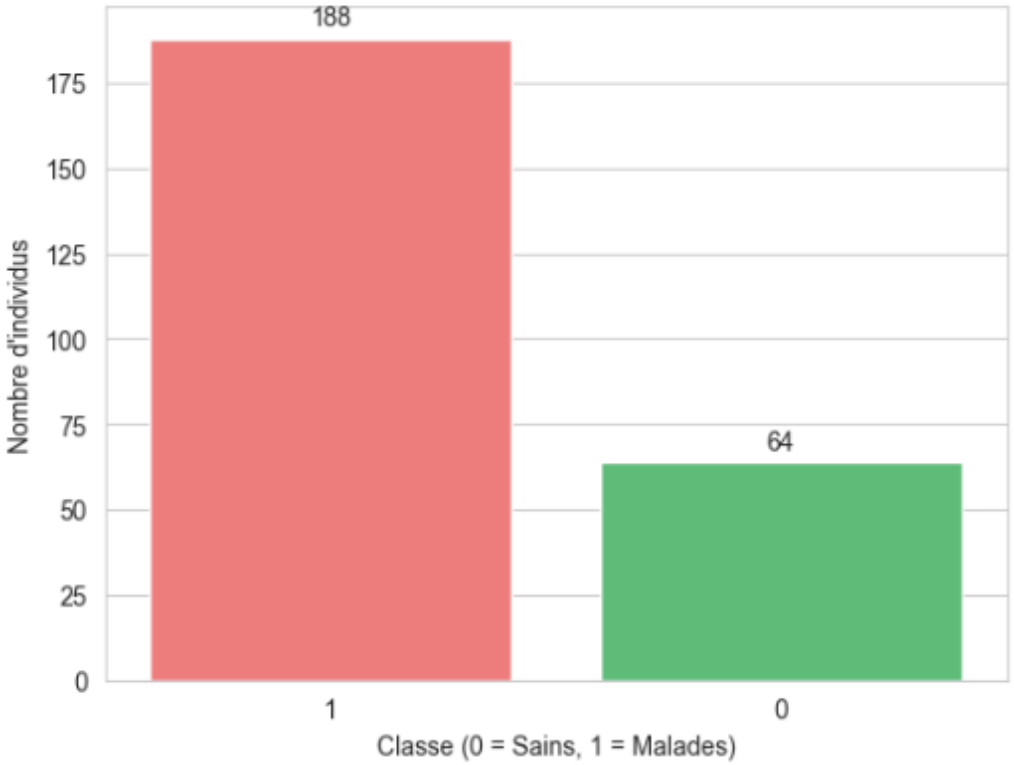


Figure 1 Distributions des classes (Malades vs Sains)

Une deuxième analyse des données inclut la matrice de corrélation qui montre les relations linéaires entre les caractéristiques vocales, avec des valeurs allant de -1 (corrélation négative forte) à +1 (corrélation positive forte). Les blocs de variables fortement corrélées (en rouge) mettent en évidence des groupes de caractéristiques interdépendantes, utiles pour la réduction de dimension ou la sélection des variables pertinentes pour les modèles.

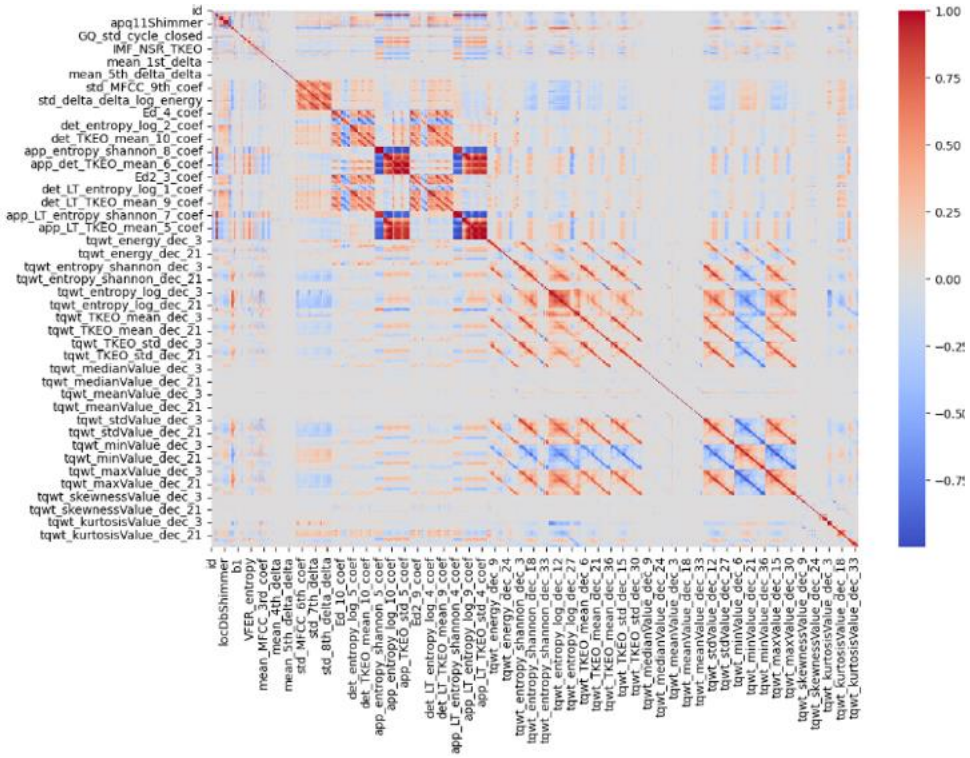


Figure 2 Matrice de corrélation

CHAPITRE II : SYNTHESE DES TRAVAUX DE RECHERCHE

1. Contexte et Motivation :

La maladie de Parkinson est une pathologie neurodégénérative qui affecte principalement les fonctions motrices et non motrices, rendant le diagnostic précoce difficile. Avec l'augmentation de l'espérance de vie, la prévalence de cette maladie progresse, justifiant le besoin de techniques innovantes pour une détection précoce. Les caractéristiques vocales, telles que le jitter, le shimmer et les coefficients MFCC, offrent une opportunité unique d'analyser la maladie grâce à l'apprentissage automatique.

2. Synthèse des Méthodes et Approches Examinées :

L'article met en avant diverses méthodes utilisées pour prédire la maladie de Parkinson à partir de données vocales. Voici un résumé des approches clés :

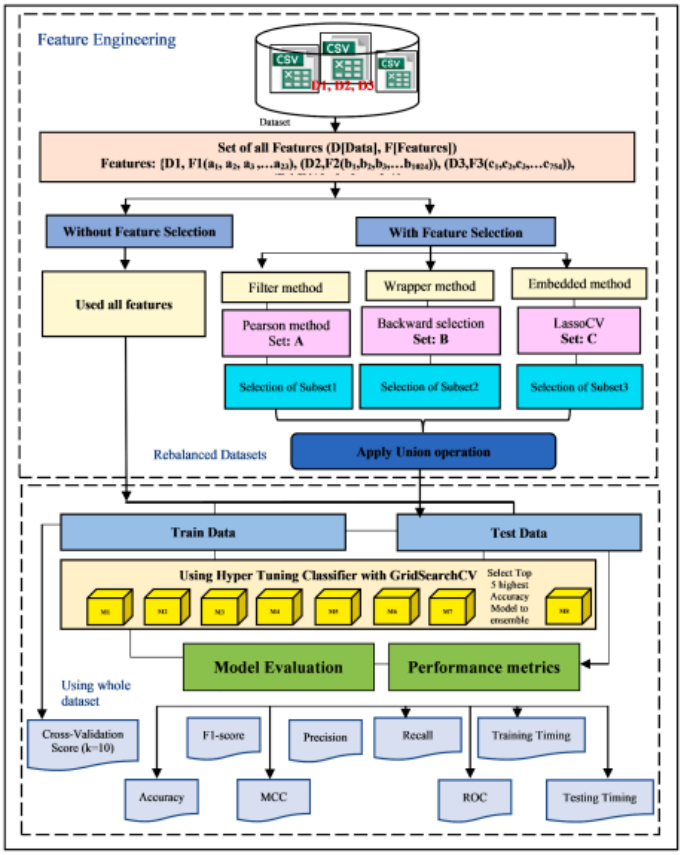


Figure 3 un résumé des approches

a. **Sélection des Caractéristiques :**
    - **EFSA (Ensemble Feature Selection Algorithm)** : Combine les méthodes basées sur les filtres (corrélation de Pearson), les wrappers (Backward Elimination), et les techniques intégrées (LassoCV). EFSA améliore la précision en sélectionnant les caractéristiques les plus pertinentes et réduit le temps de calcul.

b. **Techniques de Classification :**

Pour classifier les patients atteints de Parkinson et les individus sains, plusieurs modèles de machine learning ont été appliqués :

- **Régression Logistique** : Utilisée comme modèle de base pour comparer les performances.
- **Random Forest** : Exploite plusieurs arbres de décision pour améliorer la robustesse et réduire le surapprentissage.
- **LightGBM** : Modèle basé sur des gradient boosting machines, optimisé pour des performances rapides et efficaces avec de grandes données.

c. **Résultats et Performances :**

- Précision maximale de 94,44 % avec EFSA, contre 88,9 % sans sélection de caractéristiques et 86.73% avec PCA.
- F1-score de 90,66 % et rappel de 94,73 %, démontrant l'efficacité de la sélection de caractéristiques sur les grands ensembles de données.

d. **Méthodes de Prétraitement :**

- Techniques telles que SMOTE ont été utilisées pour équilibrer les classes minoritaires et majoritaires.

3. Résultats Notables

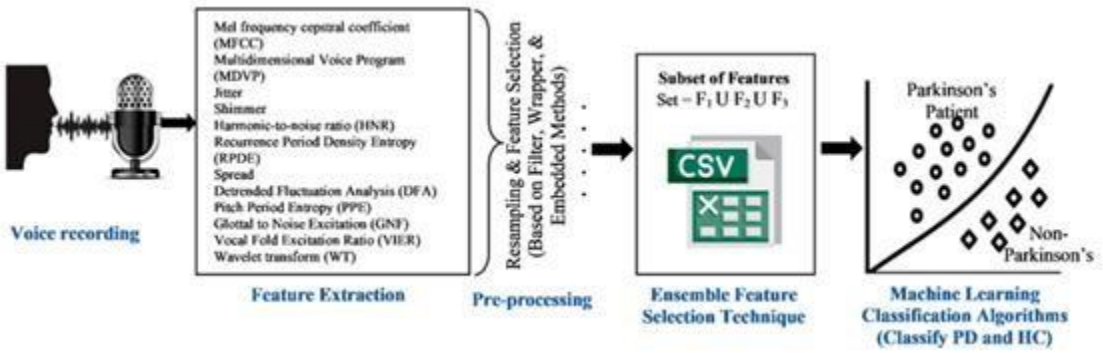
- Les approches basées sur EFSA surpassent les méthodes traditionnelles en termes de précision et de temps de calcul.
- La combinaison de plusieurs méthodes dans un modèle d'ensemble permet de mieux capturer la complexité des données vocales.

4. Perspectives

Les travaux mettent en lumière la nécessité d'une intégration des techniques d'apprentissage profond pour aller au-delà de la simple classification, en explorant également la progression de la maladie. L'utilisation d'appareils mobiles pour le suivi vocal est une voie prometteuse pour l'application pratique en télémédecine.

CHAPITRE III : THECHNIQUES DE PRETRAITEMENT DE DONNEES

Le prétraitement des données est une étape essentielle pour garantir que les modèles de machine learning fonctionnent correctement et produisent des résultats fiables.



1. Nettoyage de données

- **Suppression des colonnes inutiles** (comme 'id') à l'aide de la fonction *drop()* pour éviter les variables non pertinentes.
- **Gestion des valeurs manquantes** avec *dropna()* pour éliminer les lignes contenant des informations incomplètes, ce qui améliore la qualité des données utilisées.
- **Réinitialisation de l'index** avec *reset_index()* pour éviter toute incohérence après la suppression des lignes.
- **Normalisation des caractéristiques** en mettant toutes les caractéristiques à la même échelle pour éviter que certaines caractéristiques (avec des plages de valeurs plus grandes) ne dominent les autres.

2. Application de SMOTE (Synthetic Minority Oversampling Technique)

L'application de **SMOTE** (Synthetic Minority Oversampling Technique) est une méthode couramment utilisée pour résoudre le problème de déséquilibre des classes dans les jeux de données.

D'après l'analyse initiale des classes (Figure 1), il a été constaté que le dataset est fortement déséquilibré :

- Nombre d'échantillons de la classe **Malade (1)** : 188
- Nombre d'échantillons de la classe **Sain (0)** : 64

Ce déséquilibre peut biaiser les modèles d'apprentissage automatique, car ils risquent de favoriser la classe majoritaire au détriment de la minoritaire. Pour corriger cela, **SMOTE** a

été appliqué afin de générer des exemples synthétiques pour la classe minoritaire et équilibrer les classes.

Après l'application de SMOTE :

- Les données ont été rééchantillonnées pour obtenir un nombre égal d'exemples dans chaque classe.
- Un nouveau DataFrame équilibré (data_balanced) a été créé.

La figure suivante illustre la distribution des classes après l'équilibrage avec SMOTE :

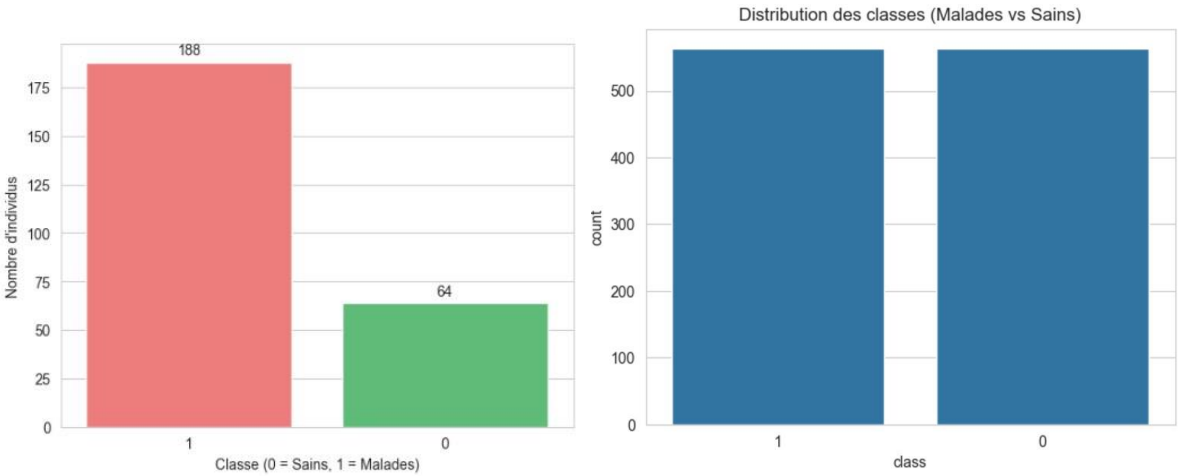


Figure 4 Distributions des classes après SMOTE

3. Sélection des Caractéristiques (1ère approche)

En appliquant la méthode d'EFSA à notre dataset, nous cherchons à identifier un sous-ensemble optimal de caractéristiques ayant pour objectif de maximiser la performance des modèles de classification tout en minimisant la complexité, les biais et le surapprentissage. Cela est réalisé à travers l'application de plusieurs techniques, notamment le filtrage, les méthodes intégrées comme Lasso, et la sélection basée sur les coefficients logistiques.

3.1. Structure et méthodologie

a. Objectifs principaux :

- Sélectionner les caractéristiques les plus pertinentes pour maximiser la performance prédictive.
- Réduire la complexité du modèle en éliminant les caractéristiques redondantes ou non-informatives.
- Fournir une méthode flexible combinant plusieurs approches de sélection.

b. Méthodes utilisées :

i. Méthode de filtrage basée sur la corrélation :

- Calcule les corrélations entre chaque caractéristique et la variable cible.
- Classe les caractéristiques en fonction de la valeur absolue des corrélations.
- Objectif : Retenir les caractéristiques ayant les corrélations les plus fortes.

ii. Régression logistique pénalisée Lasso (méthode intégrée) :

- Implémente une régularisation L1 pour contraindre les coefficients à zéro lorsque les caractéristiques ne sont pas pertinentes.
- Utilise une validation croisée pour sélectionner le meilleur hyperparamètre alpha.
- Classe les caractéristiques en fonction de l'importance de leurs coefficients.

iii. Méthode RFE (***Recursive Feature Elimination***) :

- Utilise la régression logistique comme estimateur de base pour évaluer l'importance des caractéristiques.
- Élimine récursivement les caractéristiques les moins importantes jusqu'à atteindre le nombre cible.

iv. Concaténation et tri des scores des caractéristiques :

- Fusionne les résultats des trois méthodes ci-dessus.
- Attribue un score global à chaque caractéristique en combinant les scores des différentes approches.
- Identifie les caractéristiques communes entre les trois méthodes pour priorisation.

3.2. Résultats et observations

a. Méthode de filtrage :

- Les corrélations calculées donnent une idée rapide des caractéristiques les plus fortement associées à la cible.
- Limitation : Ne tient pas compte des interactions entre les caractéristiques.

b. Régression logistique Lasso :

- La pénalisation L1 a permis d'éliminer les caractéristiques non pertinentes tout en conservant celles ayant un impact direct sur la cible.
- Le choix optimal du paramètre alpha via validation croisée a renforcé la robustesse de la méthode.

c. Méthode RFE :

- A montré de bonnes performances pour identifier les caractéristiques importantes dans un sous-ensemble limité.
- Son principal avantage réside dans l'élimination progressive, mais elle est plus coûteuse en termes de calculs.

3.3. Fusion et sélection finale

- Les caractéristiques communes aux trois méthodes ont été priorisées pour garantir leur pertinence globale.
- Les scores globaux ont permis de trier les caractéristiques restantes et de sélectionner celles qui maximisent l'information tout en limitant la redondance.

3.4. Résultats d'application d'EFSA sur le jeu de données

Après l'application de **SMOTE** pour équilibrer les classes, la méthode **EFSA (Embedded Feature Selection Algorithm)** a été utilisée afin d'identifier les caractéristiques les plus pertinentes du jeu de données.

Résultats obtenus :

- **Meilleur alpha sélectionné** : 0.0001
- **Nombre total de caractéristiques sélectionnées** : 100

Structure des données après application d'EFSA :

- **X_selected** : (1128, 100), ce qui signifie que 1128 échantillons (après équilibrage avec SMOTE) et 100 caractéristiques pertinentes ont été conservés pour l'entraînement du modèle.

Ces résultats permettent de réduire efficacement la dimensionnalité du jeu de données tout en conservant les informations essentielles, établissant ainsi une base solide pour améliorer les performances du modèle d'apprentissage automatique dans les étapes suivantes.

4. Réduction des données à l'aide de l'Analyse en Composantes Principales (ACP)

L'Analyse en Composantes Principales (ACP) est une méthode statistique utilisée pour réduire la dimension des données tout en conservant la majeure partie de l'information. Dans ce projet, l'ACP a été appliquée pour simplifier les données acoustiques extraites des signaux vocaux tout en préservant la variabilité importante des caractéristiques.

4.1. Structure et méthodologie

- **Standardisation des Données** :

  - Les données sont centrées et réduites afin de garantir une échelle uniforme entre les variables. Cela permet d'éliminer l'influence des unités et des amplitudes différentes.
  - Mathématiquement, on centre les données en soustrayant la moyenne de $x =$ chaque variable : $\mathbf{X_{centre}} = \mathbf{X} - \boldsymbol{\mu}$

    Où $\boldsymbol{\mu} = \sum_{i=1}^{n} \boldsymbol{x_i}$ est le vecteur des moyennes de chaque variable.

- **Calcul de la Matrice de Covariance** :

  - Une fois les données standardisées, la matrice de covariance est calculée pour analyser les relations entre les variables et identifier les directions de variabilité maximale.
  - L'utilisation de la matrice de covariance pour calculer les valeurs propres est au cœur de l'ACP. Cela permet de quantifier la variabilité dans différentes directions et de sélectionner les axes (composantes principales) qui retiennent le maximum d'information dans les données tout en réduisant la dimensionnalité.

    La matrice de covariance est définie comme :

$$\boldsymbol{Cov(X) = \frac{1}{n-1}\, X^{t}_{\text{centré}} \,.\, X_{\text{centré}}}$$

  - Dans le code, cela se fait via la fonction np.cov en transposant X pour traiter les échantillons comme colonnes.

- **Calcul des Valeurs et Vecteurs Propres** :

L'ACP utilise la matrice de covariance pour calculer les valeurs propres et les vecteurs propres.
Les valeurs propres : mesurent l'importance des composantes principales,
Les vecteurs propres : indiquent les directions de variabilité maximale.
Les valeurs propres (λ) et les vecteurs propres (v) de la matrice de covariance C sont obtenus en résolvant l'équation caractéristique : **det(cov(X)- λ I)=0**
La somme des valeurs propres représente la variance totale des données.

  - Dans le code :
    - Les valeurs propres et vecteurs propres sont obtenus avec np.linalg.eig(cov).
    - Les valeurs propres sont triées par ordre décroissant pour retenir les composantes principales les plus significatives.

- **Projection des Données**:
  - Les données sont projetées dans le nouvel espace formé par les composantes principales. Cela réduit la dimensionnalité tout en préservant le maximum de variance.
  - La projection est donnée par : $\mathbf{X_{proj}=X_{centré}\cdot V}$ où V contient les vecteurs propres sélectionnés correspondant aux plus grandes valeurs propres.

- **Choix du Nombre de Composantes Principales**

Plusieurs critères peuvent être utiliser pour déterminer le nombre optimal de composantes à conserver :

- **Critère de Kaiser**:
  - Retenir les composantes ayant des valeurs propres supérieures à 1.
- **Graphique d'Éboulis** :
  - Identifier le "point de coude", où l'ajout de nouvelles composantes n'apporte qu'une faible contribution supplémentaire.

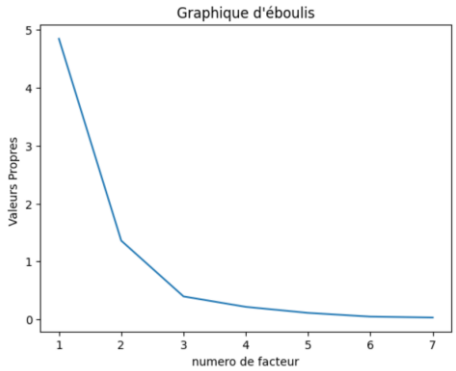


Figure 5 Graphique d'Éboulis

- **Test des Bâtons Brisés** :
  - Comparer les valeurs propres avec des seuils définis par un modèle aléatoire.
- **Variance Expliquée**:
  - Dans ce projet, nous avons utilisé le critère de **variance expliquée cumulative** pour déterminer le nombre optimal de composantes principales à conserver. L'objectif était de préserver un maximum de variance tout en réduisant la dimensionnalité des données.

    Pour cela, nous avons fixé un seuil de **99,98 %** de variance cumulative. En analysant la courbe cumulative de la variance expliquée (voir graphique ci-dessous), nous avons déterminé que le nombre optimal de composantes nécessaires pour atteindre ce seuil est de **23**. Cela signifie que ces 23 composantes principales contiennent pratiquement toute l'information significative présente dans les 576 caractéristiques originales.

    Dans ce projet, nous avons utilisé le critère de **variance expliquée cumulative** pour déterminer le nombre optimal de composantes principales à conserver. L'objectif était de préserver un maximum de variance tout en réduisant la dimensionnalité des données.

    Pour cela, nous avons fixé un seuil de **99,98 %** de variance cumulative. En analysant la courbe cumulative de la variance expliquée (voir graphique ci-dessous), nous avons déterminé que le nombre optimal de composantes nécessaires pour atteindre ce seuil est de **23**. Cela signifie que ces 23 composantes principales contiennent

pratiquement toute l'information significative présente dans les 576 caractéristiques originales.

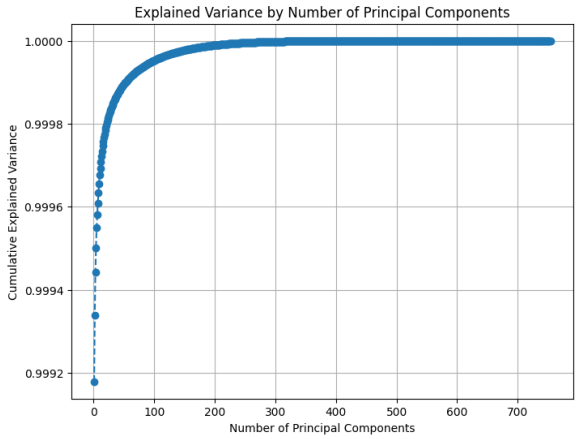
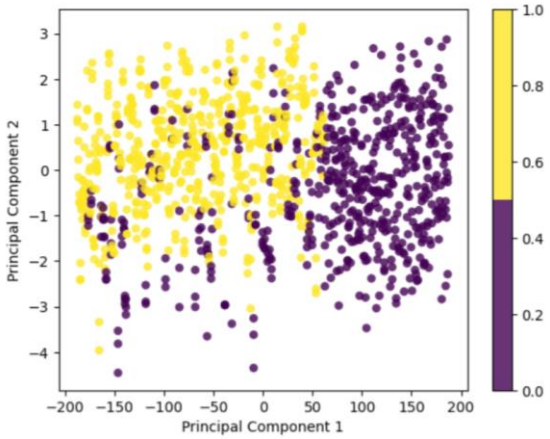


Figure 6 Visualisation de la Variance Expliquée Cumulée

4.2. Projection des Données sur les Composantes Principales

Après avoir identifié les composantes principales, nous avons projeté les données dans cet espace réduit pour visualiser leur distribution. Les résultats montrent une bonne séparation dans l'espace formé par les deux premières composantes principales. Cette distribution est illustrée par le graphique suivant :



Pour une analyse approfondie, les données ont également été projetées sur les **23 composantes principales** retenues. Cette représentation met en évidence l'apport de chaque composante dans la réduction de la dimensionnalité tout en conservant l'essentiel de l'information :

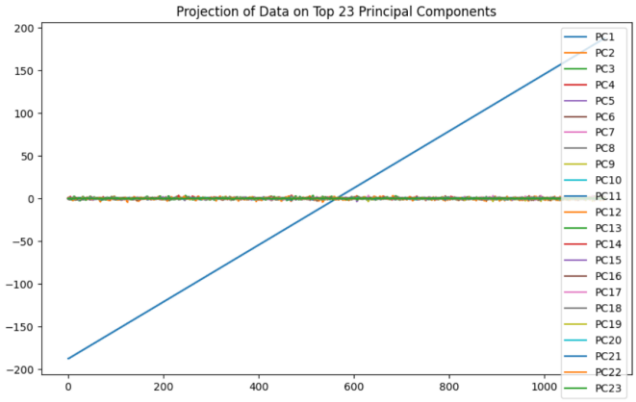


Figure 7 Représentation des Données sur les 23 Composantes Principales

4.3. Interprétation des Résultats

	Dataset Shape	Before Pre-processing	After Pre-processing	After Applying FS	After Applying PCA
Training Size	1128 x 753	756 x 755	(902, 753)	(902, 100)	(902, 23)
Testing Size			(226, 753)	(226, 100)	(226, 23)

Le jeu de données a été transformé via prétraitement, sélection de caractéristiques (FS) et réduction de dimensionnalité avec PCA. Initialement, le jeu d'entraînement comptait 1128 échantillons avec 753 caractéristiques, et le jeu de test 226. Après prétraitement, la taille du jeu d'entraînement a été réduite à 902 échantillons, sans changer le nombre de caractéristiques. La sélection de caractéristiques a ensuite réduit ce nombre à 100 pour les deux jeux, conservant les plus pertinentes. Enfin, PCA a réduit les caractéristiques à 23, capturant la variance maximale et minimisant les risques de surapprentissage. Ces étapes ont préparé le jeu de données pour des modèles d'apprentissage automatique plus efficaces et précis.

CHAPITRE IV : ENTRAINEMENT DU MODELE

1. Introduction :

Dans cette section, nous détaillons l'implémentation manuelle des différents modèles de machine learning utilisés pour classifier les individus atteints de la maladie de Parkinson. Nous avons choisi d'implémenter ces modèles à partir de zéro pour mieux comprendre leur fonctionnement interne et optimiser chaque étape de l'apprentissage.

2. Modèle de machine learning

2.1 Régression Logistique

La régression logistique est un modèle de classification binaire utilisé pour prédire des probabilités. Pour cette étude, une régression logistique a été mise en œuvre manuellement en utilisant un algorithme de descente de gradient pour ajuster les poids du modèle.

a. Structure et méthodologie

- **Initialisation des paramètres** :

Le taux d'apprentissage et le nombre d'itérations (époques) sont définis lors de l'initialisation du modèle. Par défaut, le taux d'apprentissage est de 0,01 et le nombre d'époques est de 1000.

- **Fonction Sigmoïde** :

La fonction sigmoïde est utilisée pour prédire la probabilité d'appartenance à la classe positive. La formule est donnée par :

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

où z est le produit scalaire entre les données d'entrée et les poids du modèle.

- **Fonction de perte (Binary Cross-Entropy)** :

La fonction de perte est calculée à l'aide de l'entropie croisée binaire, qui mesure l'écart entre la prédiction du modèle et la vérité terrain. La formule est :

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

où y est la valeur réelle et $\hat{y}$ est la probabilité prédite.

- **Descente de gradient** :

Pour minimiser la fonction de perte, la méthode de descente de gradient est utilisée pour ajuster les poids du modèle :

$$w = w - \alpha \nabla_w L$$

où α est le taux d'apprentissage et $\nabla_w L$ est le gradient de la fonction de perte par rapport aux poids.

2.2. Random Forest

Le modèle Random Forest (forêt aléatoire) est un ensemble d'arbres de décision qui fonctionne par agrégation des prédictions de plusieurs arbres. Chaque arbre est entraîné sur un sous-ensemble aléatoire des données (avec remise), et les prédictions finales sont basées sur un vote majoritaire des arbres. Ce modèle permet de réduire le risque de surapprentissage par rapport à un seul arbre de décision.

a. Structure et méthodologie

1. **Échantillonnage bootstrap** :

Pour chaque arbre de la forêt, un échantillon bootstrap est créé. Il s'agit d'un échantillon de données tiré aléatoirement avec remise parmi les données d'entraînement. Cela permet de former des arbres indépendants et d'améliorer la robustesse du modèle global. La fonction *bootstrap_sample(X, y)* génère ces sous-échantillons.

2. **Construction des arbres de décision** :

Chaque arbre de décision est construit à l'aide de la classe *DecisionTree*. Lors de la construction d'un arbre, la fonction *best_split()* est utilisée pour trouver la meilleure séparation (split) des données en fonction de l'information la plus pertinente (gain d'information). La construction de l'arbre est récursive et s'arrête lorsque la profondeur maximale est atteinte ou lorsque toutes les instances d'un sous-ensemble appartiennent à la même classe.

Fonctionnement des arbres de décision

La classe *DecisionTree* utilise un critère d'information basé sur l'entropie pour déterminer les meilleurs splits. L'entropie est calculée pour chaque feature, et la séparation qui maximise le gain d'information est choisie. Voici un aperçu du processus :

- **Calcul de l'entropie** : L'entropie est utilisée pour mesurer l'incertitude d'une partition des données. Elle est calculée à partir de la fréquence des classes dans un sous-ensemble donné.
- **Gain d'information** : Après avoir calculé l'entropie de l'ensemble de données, le gain d'information est calculé pour chaque séparation possible. Le gain d'information

mesure la réduction de l'entropie suite à la division des données selon un seuil particulier d'une caractéristique donnée.

- **Construction récursive** : À chaque nœud de l'arbre, la meilleure caractéristique et le meilleur seuil de séparation sont choisis en fonction du gain d'information. L'arbre continue à se diviser de manière récursive jusqu'à ce qu'une condition d'arrêt soit atteinte (comme la profondeur maximale ou une pureté des nœuds).

3. **Prédiction avec le Random Forest**

Après que tous les arbres aient été entraînés, la prédiction de la forêt est déterminée par un vote majoritaire parmi les arbres. Chaque arbre produit une prédiction, et la classe avec le plus grand nombre de votes devient la prédiction finale. Cela est réalisé par la fonction *predict*() de la classe *RandomForest*.

2.3. LightGBM

LightGBM est un algorithme de boosting basé sur des arbres décisionnels, conçu pour être rapide et efficace. Contrairement aux méthodes traditionnelles de boosting, il utilise une croissance **feuille par feuille** au lieu d'une croissance **niveau par niveau**, ce qui lui permet de réduire l'erreur avec un nombre d'arbres plus faible.

Dans cette implémentation simplifiée, nous avons utilisé des concepts essentiels de LightGBM, notamment :

- **Boosting des gradients** : Mise à jour des prédictions en minimisant une fonction de perte.
- **Croissance optimisée des arbres** : Construction d'arbres avec des divisions basées sur le gain.
- **Utilisation des gradients et Hessians** : Pour décider des splits et calculer les mises à jour.

a. Structure et méthodologie

i. Modèle global (LightFBM)

Le modèle LightFBM suit les étapes suivantes :

1. Initialisation avec une prédiction constante ($p_0 = 0.5$).
2. Construction d'arbres de décision de manière séquentielle.
3. À chaque étape :
   - Calcul des **gradients** ($g_i = y_i - p_i$) et des **Hessians** ($h_i = 1$).
   - Construction d'un arbre optimisé (avec *LeafWiseTree*) pour minimiser l'erreur.
   - Mise à jour des prédictions avec un taux d'apprentissage (η).

ii. Construction des arbres (LeafWiseTree)

L'algorithme construit les arbres en suivant une approche **feuille par feuille** :

- À chaque étape, la feuille avec le **meilleur gain** est divisée.
- Le gain pour chaque division est calculé à l'aide des gradients et Hessians :

$$\text{Gain} = \frac{G_L^2}{H_L + \epsilon} + \frac{G_R^2}{H_R + \epsilon} - \frac{(G_L + G_R)^2}{H_L + H_R + \epsilon}$$

où G_L, H_L sont respectivement les gradients et Hessians à gauche, et G_R, H_R ceux à droite.

iii. Utilisation de la parallélisation

La recherche des meilleurs splits est optimisée avec la bibliothèque *concurrent.futures* pour paralléliser les calculs par caractéristiques.

3. Conclusion

Dans ce chapitre, nous avons mis en œuvre trois techniques de machine learning pour la classification des individus atteints de la maladie de Parkinson : la régression logistique, le Random Forest et le LightGBM. Ces modèles ont été implémentés manuellement afin d'avoir une compréhension approfondie de leur fonctionnement interne et de leurs mécanismes d'apprentissage.

Chaque modèle a été entraîné et évalué dans trois configurations différentes :

1. **Sans sélection de caractéristiques** : Toutes les variables disponibles sont utilisées sans aucune réduction.
2. **Avec sélection de caractéristiques via EFSA**: Les caractéristiques les plus pertinentes sont sélectionnées pour réduire le bruit et améliorer les performances.
3. **Avec réduction dimensionnelle via ACP**: Les données sont transformées en un espace à dimensions réduites pour capturer les tendances principales tout en réduisant la complexité.

CHAPITRE V : ÉVALUATION DES MODELES ET RESULTATS

1. Évaluation sur l'ensemble d'entraînement et de test

Nous avons évalué nos modèles sur des ensembles d'entraînement et de test pour garantir une mesure fiable des performances. Nous avons testé trois modèles principaux :

- **Random Forest**
- **Logistic Regression**
- **LightGBM**

Ces modèles ont été évalués avec différentes approches de traitement des caractéristiques :

- Sans sélection (aucune réduction de dimensionnalité ou sélection de caractéristiques).
- Avec sélection des caractéristiques via **EFSA**.
- Avec réduction de dimensionnalité via **ACP (PCA)**.

Ces évaluations ont permis de comparer les gains de performances et d'efficacité computationnelle obtenus en appliquant ces techniques.

2. Métriques de performance

2.1. Objectifs

La performance d'un algorithme de Machine Learning est directement liée à sa capacité à prédire un résultat. Lorsque l'on cherche à comparer les résultats d'un algorithme à la réalité, on utilise une matrice de confusion.
On utilise la matrice de confusion dans les problèmes de classification.

Réel

Prédiction		**Positive**	**Négative**
	Positive	**True Positive (TP)**	**False Négative (FN)**
	Négative	**False Positive (FP)**	**True Négative (TN)**

On classe les résultats en 4 catégories (Prévoir si une personne malade=> Malade =positive) :

- True Positive (TP) : la prédiction et la valeur réelle sont positives.
  Exemple : Une personne malade est prédite malade.
- True Negative (TN) : la prédiction et la valeur réelle sont négatives.

Exemple : Une personne saine est prédite saine.

- False Positive (FP) : la prédiction est positive alors que la valeur réelle est négative.

Exemple : Une personne saine est prédite malade.

- False Negative (FN) : la prédiction est négative alors que la valeur réelle est positive.

Exemple : Une personne malade est prédite saine.

Dans cette section, nous évaluons les performances des modèles en utilisant plusieurs métriques standard pour les tâches de classification. Les résultats sont obtenus à partir des prédictions des modèles sur les ensembles de données d'entraînement et de test.

2.2. Les métriques utilisées

- **Accuracy (Précision globale)** :

Accuracy donne une vue d'ensemble de la capacité d'un modèle à prédire correctement les classes.
Proportion de prédictions correctes sur l'ensemble des prédictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **Recall (Rappel/Sensibilité)** :

Le recall mesure la proportion d'instances réellement positives qui ont été correctement identifiées par le modèle.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Precision (Précision)** :

La précision nous indique combien de cas correctement prédits sont réellement s'est avéré positif. Cela déterminerait si le modèle est fiable ou pas.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **F1 Score** :

Le F1-score combine à la fois la Precision et le Recall en une seule mesure. Il est particulièrement utile lorsque l'on souhaite équilibrer la nécessité de minimiser à la fois les faux positifs et les faux négatifs

$$\text{F1 score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **MCC (Matthews Correlation Coefficient)** :

Mesure robuste de la qualité de la classification, utile pour des jeux de données déséquilibrés.

$$\mathbf{MCC} = \frac{\mathbf{TP} * \mathbf{TN} - \mathbf{FP} * \mathbf{FN}}{\sqrt{(\boldsymbol{TP} + \boldsymbol{FP})(\boldsymbol{TP} + \boldsymbol{FN})(\boldsymbol{TN} + \boldsymbol{FP})(\boldsymbol{TN} + \boldsymbol{FN})}}$$

3. Résultats et interprétations

3.1. Comparaison des approches

Une fois les calculs effectués, nous avons généré des tableaux comparant les performances pour chaque modèle testé.

	Model	Feature Selection	Accuracy	Precision	Recall	F1 Score	MCC	Time Duration
0	Random Forest	None	80.09	77.69	86.32	81.78	60.37	0h 38min 21s
1	Random Forest	EFSA	82.30	76.55	94.87	84.73	66.36	0h 5min 40s
2	Random Forest	PCA	84.07	81.89	88.89	85.25	68.27	0h 1min 1s
3	Logistic Regression	None	86.73	93.07	80.34	86.24	74.29	0h 0min 3s
4	Logistic Regression	EFSA	90.71	94.44	87.18	90.66	81.70	0h 0min 0s
5	Logistic Regression	PCA	86.73	86.55	88.03	87.28	73.41	0h 0min 0s
6	LightGBM	None	78.32	79.82	77.78	78.79	56.64	0h 2min 55s
7	LightGBM	EFSA	79.20	85.71	71.79	78.13	59.44	0h 0min 25s
8	LightGBM	PCA	81.42	80.00	85.47	82.64	62.85	0h 0min 4s

Tableau 1 comparaison entre les trois approches

3.2. Evaluation des modèles entre les approches

- ✓ **Sans réduction ou sélection des caractéristiques** : Logistic Regression a obtenu les meilleures performances globales en termes de précision (86.73 %) et de F1-Score (86.24 %).

- ✓ **Avec EFSA** : Logistic Regression a surpassé les autres avec une précision de 90.71 % et un F1-Score de 90.66 %, démontrant l'efficacité de cette méthode pour la sélection des caractéristiques pertinentes.

- ✓ **Avec PCA** : Bien que toutes les performances aient été globalement améliorées, Random Forest avec PCA a montré une précision accrue de 84.07 % et une meilleure gestion des compromis précision/rappel.

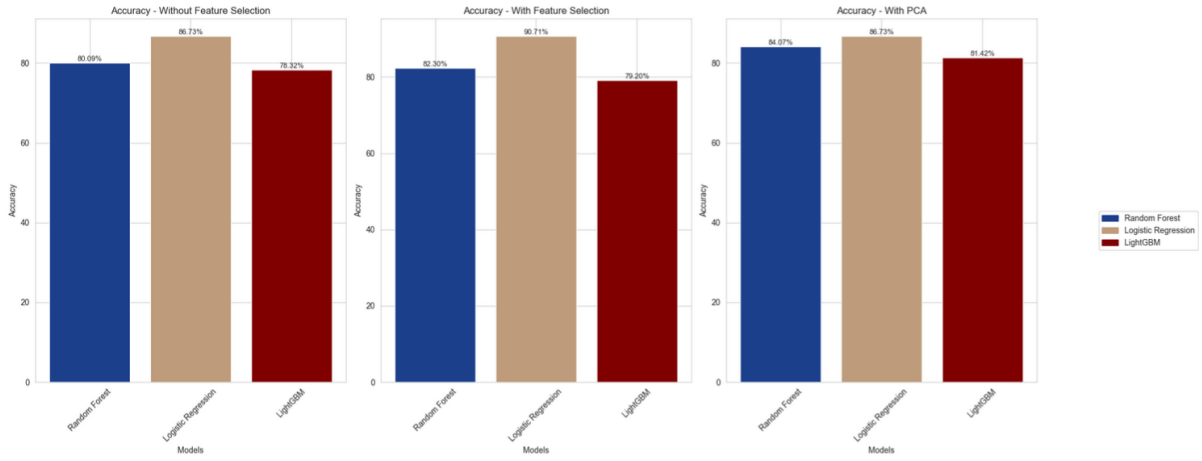


Figure 8 Evaluation de performances de chaque modèles entre les trois approches

Analyse de l'efficacité computationnelle

- Le **Random Forest** est le plus coûteux en temps d'exécution, en particulier sans réduction de dimensionnalité (38 min 21 s). L'application de PCA a considérablement réduit ce temps à 1 min 1 s.
- Logistic Regression s'est distinguée par une rapidité exceptionnelle avec une durée négligeable pour l'entraînement (moins de 3 secondes pour tous les cas).

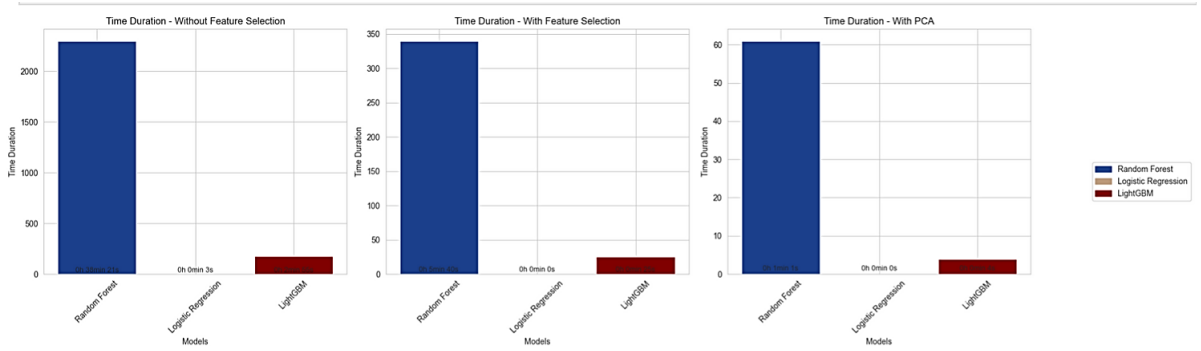


Figure 9 Temps d'exécution pour chaque modèle en fonction des techniques utilisées.

4. Conclusion

L'application des techniques de sélection des caractéristiques (EFSA) et de réduction de dimensionnalité (PCA) a non seulement amélioré les performances des modèles, mais a également réduit de manière significative les coûts computationnels. Les résultats mettent en évidence l'importance de choisir la bonne combinaison de techniques en fonction des priorités du projet (précision ou rapidité). **Logistic Regression avec EFSA** s'est avérée être la meilleure approche, obtenant les scores les plus élevés en précision et en F1-Score avec un temps d'exécution optimal.

CONCLUSION

Ce projet démontre l'importance d'une approche méthodique et comparative dans le développement de modèles de machine learning. En combinant des techniques de sélection de caractéristiques, de réduction dimensionnelle, et des algorithmes avancés, nous avons créé une base solide pour classifier efficacement la maladie de Parkinson. Les résultats de l'évaluation permettront de déterminer les meilleures pratiques pour des applications similaires dans d'autres contextes médicaux et scientifiques.

Ce travail met également en lumière l'importance de comprendre les algorithmes à un niveau fondamental, ce qui ouvre des perspectives pour des améliorations sur mesure et une adaptation à des problèmes spécifiques.