REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

Université 20 Aout 1955- Skikda

Faculté des Science

Département informatique

Mémoire

En vue de l'obtention du diplôme de Master

Filière : Informatique

Spécialité : Réseaux et systèmes distribués

Thème :

Développement d'un système de diagnostic médical du diabète à l'aide d'algorithmes d'apprentissage automatique

Réalisé par :

- **Bouaita Samah**
- **Boussekine Marwa**

Encadré par :

- **Adel Lahsasna**

2022-2023

Remerciements

Nous remercions DIEU de nous avoir donné patience, santé, courage et capacité à faire ce travail. Nous tenons à vous remercier sincèrement au superviseur, Dr. Lahsasna Adel, pour ses conseils et ses encouragements à notre égard et son soutien et son suivi constants qui nous ont permis de mener à bien ce travail en les meilleures conditions. Nous remercions sincèrement les membres du jury d'avoir accepté d'examiner et évaluer notre travail. Nous exprimons également notre gratitude à tous les enseignants et enseignantes du département d'informatique, sans oublier bien sûr de remercier profondément tous ceux qui nous ont soutenu directement ou indirectement pour accomplir ce travail.

Dédicaces

Nous dédions ce mémoire : À nos chers parents, peu importe ce que nous faisons ou disons, nous ne vous remercierons jamais assez comme vous se doit. À nos familles pour leur amour et leur soutien pour nous. À tous les amis et à tous ceux qui ont toujours cru en nous et nous ont poussés à atteindre ce succès.

Résumé

Le diabète est un problème majeur de santé mondiale, causant des millions de décès chaque année et entraînant diverses complications. La prévalence du diabète ne cesse d'augmenter, notamment dans les pays à faible et moyen revenu. Cette étude vise à développer un système de diagnostic précis du diabète en utilisant l'apprentissage automatique. L'objectif est d'identifier l'algorithme de classification le plus précis parmi huit algorithmes couramment utilisés, à savoir les Réseaux de Neurones Artificiels (ANN), les Machines à Vecteurs de Support (SVM), le classifieur Naïf de Bayes, les Arbres de Décision, les Forêts Aléatoires, le Bagging, l'AdaBoost et les k-Plus Proches Voisins (KNN). L'étude utilise l'ensemble de données sur le diabète de Pima pour évaluer les performances de ces algorithmes. L'algorithme sélectionné est ensuite utilisé pour développer un système de diagnostic efficace. De plus, l'étude vise à identifier les caractéristiques les plus importantes liées à la classification du diabète afin de fournir des informations sur les facteurs de risque du diabète. Les résultats montrent que la Forêt Aléatoire atteint la plus haute précision de classification et est choisie pour la phase de déploiement. Le système de diagnostic développé diagnostique avec précision les nouveaux patients et identifie les facteurs les plus importants dans la classification du diabète, tels que la concentration de glucose, l'indice de masse corporelle (IMC) et l'âge. Cette recherche contribue à la détection du diabète, ce qui peut aider les professionnels de la santé à fournir le traitement approprié du diabète et à atténuer le risque de développer la maladie pour les personnes non diabétiques en contrôlant les facteurs de risque significatifs associés au diabète.

Abstract

Diabetes is a major global health concern, causing millions of deaths annually and leading to various complications. The prevalence of diabetes has been steadily increasing, particularly in low- and middle-income countries. This study aims to develop an accurate diabetes diagnostic system using machine learning. The objective is to identify the most accurate classification algorithm among eight commonly used algorithms, namely Artificial Neural Networks (ANN), Support Vector Machines (SVM), Naive Bayes classifier, Decision Trees, Random Forests, Bagging, AdaBoost, and k-Nearest Neighbors (KNN). The study utilizes the Pima diabetes dataset to evaluate the performance of these algorithms. The selected algorithm is then used to develop an efficient diagnostic system. Additionally, the study aims to identify the most important features related to diabetes classification to provide insights into risk factors for diabetes. The results show that Random Forest achieves the highest classification accuracy, and it is chosen for the deployment phase. The developed diagnostic system accurately diagnoses new patients and identifies the most important factors in diabetes classification, such as glucose concentration, body mass index (BMI), and age. This research contributes to diabetes detection, which can help healthcare professionals to provide the appropriate treatment of diabetes and mitigate the risk of developing the disease for non-diabetes individuals by controlling the significant risk factors associated with diabetes.

الملخص

السكري هو قلق صحي عالمي رئيسي، حيث يتسبب في ملايين الوفيات سنويًا ويؤدي إلى تعقيدات مختلفة. تزداد انتشار السكري بشكل مطرد، وخاصة في البلدان ذات الدخل المنخفض والمتوسط. تهدف هذه الدراسة إلى تطوير نظام تشخيص دقيق للسكري باستخدام التعلم الآلي. الهدف هو تحديد الخوارزمية التصنيفية الأكثر دقة من بين ثماني خوارزميات شائعة الاستخدام، وهي ANN، SVM، Naive Bayes، decision tree، Random Forest، Bagging، AdaBoost، KNN. تستخدم الدراسة مجموعة بيانات السكري Pima لتقييم أداء هذه الخوارزميات. يتم استخدام الخوارزمية المحددة بعد ذلك لتطوير نظام تشخيص فعال. بالإضافة إلى ذلك، تهدف الدراسة إلى تحديد السمات الأكثر أهمية المرتبطة بتصنيف السكري لتوفير رؤى حول عوامل الخطر للسكري. تظهر النتائج أن Random Forest تحقق أعلى دقة في التصنيف، وتم اختيارها لمرحلة التنفيذ. يشخص النظام التشخيصي المطور المرضى الجدد بدقة ويحدد العوامل الأكثر أهمية في تصنيف السكري، مثل تركيز الجلوكوز، ومؤشر كتلة الجسم(BMI) ، والعمر. تساهم هذه البحث في كشف السكري، مما يساعد المهنيين في الرعاية الصحية على تقديم العلاج المناسب للسكري والتقليل من مخاطر تطوير المرض للأفراد غير السكريين من خلال مراقبة العوامل الرئيسية المرتبطة بالسكري.

TABLE DES MATIÈRES

Chapitre 1 : Introduction …… 1
1.1. Contexte général ……. 1
1.2. Problématique ……. 1
1.3. Les objectifs de l'étude …… 2
1.4. Approche adoptée ……. 2
1.5. Plan du mémoire ……. 2
Chapitre 2 : Etat de l'art du diagnostic du diabète à l'aide d'algorithmes d'apprentissage automatique …….. 4
2.1. Introduction ……... 4
2.1.1. Bref aperçu du diabète et son importance …….. 4
2.1.2. Rôle de l'apprentissage automatique dans le diagnostic du diabète …….. 4
2.2. Compréhension du diabète ……. 5
2.2.1. Type de diabète …….. 5
2.2.2. Facteurs de risque et symptôme associés au diabète …….. 5
2.2.3. Importance d'un diagnostic précoce et précis ……. 7
2.3. Approche traditionnelles pour le diagnostic du diabète ……. 8
2.3.1. Taux de glycémie à jeun ……. 8
2.3.2. Le test d'hyperglycémie provoquée (HGPO) …… 8
2.3.3. L'hémoglobine glyquée (HbA1c) …….. 8
2.3.4. Limitations et défis des méthodes de diagnostic traditionnelles …… 8
2.4. Apprentissage automatique dans le diagnostic du diabète …… 9
2.4.1. Aperçu des algorithmes d'apprentissage automatique …….. 9
2.4.2. Collecte et prétraitement des données liées au diabète ……. 10
2.4.3. Stratégies de formation et d'évaluation des modèles ……. 10
2.5. Etat de l'art des algorithmes d'apprentissage automatique pour le diagnostic du diabète …….. 11
2.5.1. Arbres de decision ……... 11
2.5.2. Machines à vecteurs de support (SVM) …….. 13
2.5.3 Réseaux de neurones artificiels (ANN) …….. 15
2.5.4. La forêt aléatoire …… 18
2.5.5. Le classificateur naïf de Bayes ……. 20
2.5.6. Le bagging ……. 22
2.5.7. AdaBoost …….. 24
2.5.8. K-plus proches voisins (KNN) ……. 26
2.6. Evaluation des performances …… 29
Chapitre 03 : La méthodologie …… 31
31

3.1 Introduction ……
3.2 Description du jeu de données Pima diabetes …… 31
3.3 Les algorithmes d'apprentissage automatique ……. 32
3.3.1. Les réseaux de neurones (Artificial Neural Networks) ……. 32
3.3.2. Arbre de decision …… 32
3.3.3. Machine à vecteur de support (Support Vector Machines) …… 33
3.3.4. Forêt aléatoire (Random Forest) ……. 33
3.3.5.K-plus proches voisins (KNN) …… 33
3.3.6. Naïve Bayes (NB) …….. 34
3.3.7. AdaBoost …… 34
3.3.8. Bagging …….. 35
3.4 Métriques d'évaluation de la performance ……. 35
3.4.1. Précision …….. 35
3.4.2 La méthode de validation croisée à 10 plis …….. 35
3.5 La méthode utilisée pour identifier les caractéristiques les plus importantes …… 36
3.6 Description de l'implémentation du système de diagnostic du diabète ……. 37
3.6.1. L'interface principale …….. 37
3.6.2. Le module de comparaison ……. 41
3.6.3 Le module de déploiement …….. 45
3.7 Présentation de l'interface graphique du système de diagnostic du diabète ……. 52
3.7.1 Le module de comparaison ……. 53
3.7.2 Le module de déploiement ……. 55
Chapitre 4 : Résultats et Discussions …… 59
4.1 La première partie : comparaison des performances de huit algorithmes …… 59
4.2 La deuxième partie : déploiement du système ……... 61
Chapitre 5 : Conclusion …….. 63
Bibliographie …… 64

LISTE DES FIGURES

Figure 3.1 Importations de modules pour l'interface principale ……. 37

Figure 3.2 Définition des fonctions pour l'interface principale …… 38

Figure 3.3 Création de la fenêtre GUI pour l'interface principale …… 39

Figure 3.4 Création des éléments de l'interface pour l'interface principale …….. 40

Figure 3.5 Boucle principale de l'interface pour l'interface principale …… 40

Figure 3.6 Importations de modules pour le module de comparaison …….. 41

Figure 3.7 Définition des classificateurs pour le module de comparaison …….. 42

Figure 3.8 Définition de la fonction calculate_accuracy() pour le module de comparaison ……. 42

Figure 3.9 Définition de la fonction train_models() pour le module de comparaison … 43

Figure 3.10 Définition de la fonction upload_file() pour le module de comparaison 44

Figure 3.11 Définition de la fonction display_results() pour le module de comparaison 44
Figure 3.12 création des trois boutons pour le module de comparaison ……. 45
Figure 3.13 Importations de modules pour le module de déploiement …… 46
Figure 3.14 Définition de la fonction train_model(X, y) pour le module de déploiement 46
Figure 3.15 Définition de la fonction classify_model(X, y) pour le module de déploiement ……... 47
Figure 3.16 Définition de la fonction upload_file() pour le module de déploiement ….. 48
Figure 3.17 Définition de la fonction train_button_click() pour le module de déploiement ……... 48
Figure 3.18 Définition de la fonction classify_button_click() pour le module de déploiement ……... 49
Figure 3.19 Création de la fenêtre GUI pour le module de déploiement ……. 50
Figure 3.20 Création le bouton de la classification pour le module de déploiement …… 50
Figure 3.21 Création des boutons, des étiquettes d'entrée et des boîtes d'entrée pour le module de déploiement ……. 51
Figure 3.22 Création d'une étiquette pour afficher le résultat pour le module de déploiement …….. 52
Figure 3.23 l'interface d'accueil pour le système de diagnostic du diabète …….. 52
Figure 3.24 Interface du module de comparaison de diagnostic du diabète ……. 53
Figure 3.24 Importer le fichier « diabetes.csv » pour lancer la phase d'entrainement …. 54
Figure 3.25 Les résultats de l'évaluation sont affichés dans la fenêtre des résultats …… 54
Figure 3.26 Interface du module de déploiement ……. 55
Figure 3.27 Importer le fichier « diabetes.csv » …….. 56
Figure 3.28 Entrainer le système en utilisant Random Forest …… 57
Figure 3.29 Utiliser le système pour diagnostiquer de nouveaux patients …… 58

LISTE DES TABLEAUX

Table 4.1 la précision de classification moyenne de 08 algorithmes utilisant la méthode de validation croisée à 10 plis …….. 59

Chapitre 1: Introduction

1. Contexte général

Le diabète a un impact significatif sur la santé mondiale. Il est l'une des principales causes de décès dans le monde, avec 2 millions de décès attribués à la maladie en 2019 (WHO, 2023). De plus, le diabète peut entraîner diverses complications telles que les maladies cardiovasculaires, les maladies rénales, les lésions oculaires et les lésions nerveuses. Le nombre de personnes atteintes de diabète est passé de 108 millions en 1980 à 422 millions en 2014. La prévalence augmente plus rapidement dans les pays à faible et moyen revenu (comme l'Algérie) que dans les pays à revenu élevé.

Dans un contexte où le diabète est une maladie chronique en augmentation, un système de diagnostic précis peut aider à détecter la maladie plus tôt, à prévenir les complications et à fournir un traitement approprié. Cette étude cherche donc à exploiter les avancées de l'apprentissage automatique pour développer un outil de diagnostic du diabète fiable, basé sur l'analyse des caractéristiques les plus pertinentes et en utilisant l'algorithme de classification le plus performant.

2. Problématique

La problématique centrale de cette étude réside dans l'identification de l'algorithme de classification le plus précis parmi huit algorithmes couramment utilisés dans le domaine de l'apprentissage automatique. Ces algorithmes comprennent les réseaux de neurones artificiels (ANN), les machines à vecteurs de support (SVM), le classifieur naïf de Bayes, les arbres de décision, les forêts aléatoires, le bagging, l'AdaBoost et les k-plus proches voisins (KNN). Notre objectif est de déterminer quel algorithme fournit les résultats les plus précis pour la classification des patients atteints ou non de diabète.

Il est également crucial d'identifier et de contrôler les facteurs de risque significatifs associés au diabète, afin de réduire le risque de développer cette maladie chez les individus présentant des facteurs de risque.

3. les objectifs de l'étude

Les objectifs de cette étude sont d'évaluer un ensemble d'algorithmes pour sélectionner l'algorithme de classification le plus précis pour diagnostiquer le diabète, et d'utiliser ce dernier pour développer un système de diagnostic efficace. De plus, l'étude vise à identifier les caractéristiques les plus importantes liées à la classification, afin de fournir des informations sur les facteurs de risque pouvant être pris en considération pour atténuer le risque de diabète chez les personnes non atteintes de la maladie.

4. Approche adoptée

Pour répondre à cette problématique, nous avons adopté une approche comparative. Nous avons utilisé le jeu de données Pima diabetes comme étude de cas pour évaluer les performances des huit algorithmes de classification mentionnés précédemment. Chaque algorithme a été entraîné et testé sur ce jeu de données, et nous avons mesuré leur précision de classification respective. Cette évaluation comparative nous permettra de déterminer l'algorithme qui présente les performances les plus élevées et qui est donc le plus adapté pour notre système de diagnostic du diabète.

Enfin, nous avons comparé les performances des huit algorithmes et identifié celui qui offre la meilleure précision de classification. Cet algorithme sera utilisé pour développer le système de diagnostic du diabète. De plus, nous avons également identifié les caractéristiques les plus importantes qui contribuent le plus à la classification, ce qui peut fournir des informations précieuses sur les facteurs de risque à prendre en compte pour atténuer le risque de diabète chez les personnes non atteintes.

4. Plan du mémoire

Le reste de la thèse est organisé de la manière suivante :

Dans le chapitre 2, une revue de la littérature et des travaux de recherche connexes est réalisée dans le domaine du diagnostic du diabète. Plus précisément, il présente les applications des algorithmes d'apprentissage automatique suivants pour le diagnostic du diabète : les réseaux de neurones artificiels (ANN), les machines à vecteurs de support (SVM), le classifieur naïf de Bayes, les arbres de décision, les forêts aléatoires, le bagging, l'AdaBoost et les k-plus proches voisins (KNN). Il présente également les avantages et les inconvénients de chaque algorithme.

Le chapitre 3 présente la méthodologie suivie pour atteindre les objectifs de l'étude en cours. L'étude comparative est décrite, ainsi que la mesure d'évaluation permettant d'évaluer les performances de chaque algorithme. De plus, il présente les algorithmes et les caractéristiques de l'ensemble de données utilisé dans cette étude. De plus, il décrit la méthode utilisée pour évaluer l'importance de chaque caractéristique dans l'ensemble de données et sa contribution au diagnostic du diabète.

Dans le chapitre 4, les résultats obtenus dans cette étude sont présentés, discutés et analysés pour en tirer les principales conclusions.

Le chapitre 5 conclut l'étude avec des remarques conclusives et propose quelques pistes d'amélioration et de recherche future.

Chapitre 2 : Etat de l'art du diagnostic du diabète à l'aide d'algorithmes d'apprentissage automatique

2.1. Introduction

2.1.1. Bref aperçu du diabète et son importance

Le diabète est une maladie chronique qui se déclare lorsque le pancréas ne produit pas suffisamment d'insuline, ou lorsque l'organisme n'est pas capable d'utiliser efficacement l'insuline qu'il produit. L'insuline est une hormone qui régule la glycémie. L'hyperglycémie, également appelée glycémie élevée, est un effet courant du diabète non maîtrisé qui, au fil du temps, provoque de graves lésions dans de nombreux systèmes du corps, en particulier les nerfs et les vaisseaux sanguins.

En 2014, 8,5 % des adultes âgés de 18 ans et plus étaient atteints de diabète. En 2019, le diabète était la cause directe de 1,5 million de décès et 48 % de l'ensemble des décès dus au diabète sont survenus avant l'âge de 70 ans. De plus, 460 000 autres décès par maladie rénale ont été causés par le diabète et l'hyperglycémie est à l'origine d'environ 20 % des décès imputables à des maladies cardiovasculaires.

Entre 2000 et 2019, les taux de mortalité due au diabète standardisés selon l'âge ont augmenté de 3 %. Dans les pays à revenu intermédiaire de la tranche inférieure, le taux de mortalité prématurée due au diabète a augmenté de 13 %.

En revanche, la probabilité de mourir de l'un des quatre principaux types de maladies non transmissibles (maladies cardiovasculaires, cancers, affections respiratoires chroniques ou diabète) entre 30 ans et 70 ans a baissé de 22 % à l'échelle mondiale entre 2000 et 2019(SarwarN, 2010).

2.1.2. Rôle de l'apprentissage automatique dans le diagnostic du diabète

L'apprentissage automatique est de plus en plus utilisé pour détecter la fraude. En effet, les algorithmes d'apprentissage automatique peuvent apprendre à identifier des modèles qui indiquent une fraude, même lorsque ces modèles ne sont pas évidents pour les humains.

Par exemple, des algorithmes d'apprentissage automatique peuvent être utilisés pour identifier les transactions frauduleuses par carte de crédit. Ces algorithmes peuvent apprendre à identifier des modèles tels que des habitudes de dépenses inhabituelles, des

transactions effectuées à partir de différents pays ou des transactions effectuées à des heures inhabituelles de la journée.

L'apprentissage automatique est également utilisé pour détecter les fraudes à l'assurance. Ces algorithmes peuvent apprendre à identifier des modèles tels que des réclamations faites pour des événements improbables, des réclamations pour des articles de grande valeur ou des réclamations faites par des personnes ayant des antécédents de fraude.

L'apprentissage automatique est un outil puissant qui peut être utilisé pour détecter la fraude. Alors que la technologie d'apprentissage automatique continue de se développer, nous pouvons nous attendre à voir des applications encore plus innovantes de l'apprentissage automatique dans la détection des fraudes à l'avenir.(Burkov, 2019)

2.2. Compréhension du diabète

2.2.1. Type de diabète

- **Le diabète de type 1** (autrefois appelé diabète insulinodépendant ou juvénile) se caractérise par une production insuffisante d'insuline, laquelle doit être administrée quotidiennement. En 2017, 9 millions de personnes étaient atteintes de diabète de type 1 ; la plupart d'entre elles vivent dans des pays à revenu élevé. La cause du diabète de type 1 n'est pas connue, et en l'état actuel des connaissances, on ne sait pas l'éviter (Sarwar N, 2010).
- **Diabète de type 2**

  Le diabète de type 2 modifie la façon dont l'organisme utilise le sucre (glucose) comme source d'énergie. Il empêche l'organisme d'utiliser correctement l'insuline, ce qui peut entraîner une forte glycémie s'il n'est pas traité.

  Au fil du temps, le diabète de type 2 peut causer de graves lésions, en particulier des nerfs et des vaisseaux sanguins (Sarwar N, 2010).
- **Diabète gestationnel**

  Le diabète gestationnel se caractérise par la survenue d'une hyperglycémie, c'est-à-dire d'une élévation de la concentration de glucose dans le sang au-dessus des valeurs normales, mais à des valeurs inférieures à celles conduisant à poser le diagnostic de diabète. Le diabète gestationnel survient pendant la grossesse (Sarwar N, 2010).

2.2.2. Facteurs de risque et symptôme associés au diabète

A. Les facteurs (Michael p. , 2017)

- **Facteurs de risque pour le diabète de type 1** (Michael p. , 2017)

- Âge : Le diabète de type 1 est le plus courant chez les enfants et les jeunes adultes, mais il peut également se développer plus tard dans la vie.
- Antécédents familiaux : Si vous avez des antécédents familiaux de diabète de type 1, vous êtes plus susceptible de développer la maladie.
- Génétique : Certains gènes vous rendent plus susceptible de développer un diabète de type 1.
- Maladie auto-immune : les personnes atteintes de certaines maladies auto-immunes, telles que la maladie coeliaque ou la maladie d'Addison, sont plus susceptibles de développer un diabète de type 1.

- **Facteurs de risque pour le diabète de type 2** (Michael p. , 2017)

  - Âge : Le diabète de type 2 est le plus courant chez les adultes de plus de 45 ans, mais il est de plus en plus fréquent chez les enfants et les jeunes adultes.
  - Antécédents familiaux : Si vous avez des antécédents familiaux de diabète de type 2, vous êtes plus susceptible de développer la maladie.
  - Race/ethnicité : Les personnes d'ascendance afro-américaine, hispanique/latino, amérindienne, autochtone de l'Alaska, hawaïenne et insulaire du Pacifique sont plus susceptibles de développer un diabète de type 2.
  - Surpoids ou obésité : Le surpoids ou l'obésité est le principal facteur de risque du diabète de type 2.
  - Inactivité physique : L'inactivité physique augmente votre risque de développer un diabète de type 2.
  - Antécédents de diabète gestationnel : Si vous avez eu un diabète gestationnel, vous êtes plus susceptible de développer un diabète de type 2 plus tard dans la vie.
  - Syndrome des ovaires polykystiques (SOPK) : Les femmes atteintes du SOPK sont plus susceptibles de développer un diabète de type 2.
  - Antécédents de maladie cardiovasculaire : Les personnes ayant des antécédents de maladie cardiovasculaire sont plus susceptibles de développer un diabète de type 2.
  - Antécédents d'hypertension artérielle : les personnes ayant des antécédents d'hypertension artérielle sont plus susceptibles de développer un diabète de type 2.

- **Facteurs de risque pour le diabète de gestationnel** (Michael p. , 2017)

  - Âge : Le diabète gestationnel est plus fréquent chez les femmes de plus de 35 ans, mais il peut survenir chez les femmes de tout âge.

- Antécédents familiaux : Si vous avez des antécédents familiaux de diabète de type 2, vous êtes plus susceptible de développer un diabète gestationnel.
- Race/ethnicité : les personnes d'ascendance afro-américaine, hispanique/latino, amérindienne, autochtone de l'Alaska, hawaïenne et insulaire du Pacifique sont plus susceptibles de développer un diabète gestationnel.
- Surpoids ou obésité : Le surpoids ou l'obésité est le principal facteur de risque du diabète gestationnel.
- Inactivité physique : L'inactivité physique augmente votre risque de développer un diabète gestationnel.
- Antécédents de diabète gestationnel : Si vous avez eu un diabète gestationnel lors d'une grossesse précédente, vous êtes plus susceptible de le développer à nouveau.

B. Symptômes

Type 1(Chan, 2016)

- Une polydipsie (soif importante).
- Augmentation du volume des urines (polyurie).
- Fort amaigrissement (malgré un appétit augmenté).
- Vision trouble.
- Asthénie.
- Haleine à l'odeur fruitée de pomme verte.

Type 2(Chan, 2016)

- Une soif intense et une faim exagérée (polyphagie).
- Somnolence et fatigue constante.
- Besoin fréquent d'uriner (particulièrement le soir).
- Vision brouillée.
- Infections des organes vitaux.
- Cicatrisation lente.

Gestationelle (Chan, 2016)

- Fatigue importante.
- Soif accrue.
- Des urines plus abondantes.
- Envie plus fréquente d'uriner.
- Maux de tête.

2.2.3. Importance d'un diagnostic précoce et précis

Pour permettre un traitement précoce. Certaines maladies, comme le cancer, sont plus faciles à traiter si elles sont détectées tôt.

Pour prévenir d'autres complications. Si une maladie n'est pas traitée, elle peut causer des problèmes plus graves. Par exemple, une hypertension artérielle non traitée peut entraîner des maladies cardiaques et des accidents vasculaires cérébraux.

Pour améliorer la qualité de vie. Un diagnostic précoce peut aider les personnes à prendre des décisions éclairées concernant leur traitement et à obtenir le soutien dont elles ont besoin pour gérer leur état.

Pour sauver des vies. Dans certains cas, un diagnostic précoce peut sauver la vie. Par exemple, la détection précoce du cancer du sein peut conduire à une guérison dans de nombreux cas. (society, 2023).

2.3. Approche traditionnelles pour le diagnostic du diabète

Diagnostic du diabète est établi grâce à une prise de sang qui dose le taux de sucre dans Le sang. Pour les périodes de mesure nous avons :

2.3.1. Taux de glycémie à jeun

Le diagnostic est posé lorsque cette glycémie à jeun soit la prise de sang qui a eu Lieu sans apport calorique pendant huit heures au moins est égale ou supérieure à 1.26 g/l (ou 7mmol/L) et est constatée à deux reprises. (Chan, 2016)

2.3.2. Le test d'hyperglycémie provoquée (HGPO)

Ce test consiste à mesurer les taux de variations de la glycémie soit le taux de sucre dans le sang après avoir ingéré 75g d'hydrate de carbone (glucides) sous forme de boisson. Le sucre sanguin est mesuré durant les 8 heures de jeûne puis toutes les 2 heures après l'ingestion de la solution. Si le taux de la glycémie est supérieur à 11.1mmol/L, on diagnostique un diabète, ou bien un pré-diabète si le taux de glucose se trouve entre 7.8mmol/L et 11.0mmol/L.

Il permet de détecter un diabète de type 2 ou un diabète gestationnel chez la femme enceinte. (Chan, 2016)

2.3.3. L'hémoglobine glyquée (HbA1c)

Le dosage de l'hémoglobine glyquée ou l'HbA1c s'effectue par prise de sang dans un laboratoire d'analyse médicale. Il est préconisé à intervalles réguliers, tous les 2 à 3 mois environ. Il n'est pas nécessaire d'être à jeun pour la prise de sang. (Chan, 2016)

2.3.4. Limitations et défis des méthodes de diagnostic traditionnelles

Les méthodes de diagnostic traditionnelles peuvent être limitées par un certain nombre de facteurs, notamment :

• Imprécision : Les méthodes de diagnostic traditionnelles sont souvent inexactes, en particulier pour les maladies à un stade précoce. Cela peut conduire à un diagnostic erroné et à un retard de traitement.

• Caractère invasif : Les méthodes de diagnostic traditionnelles peuvent être invasives, nécessitant une intervention chirurgicale ou d'autres procédures qui peuvent être risquées et inconfortables pour les patients.

• Coût : Les méthodes de diagnostic traditionnelles peuvent être coûteuses, en particulier pour les maladies complexes ou rares. Cela peut les rendre inaccessibles pour certains patients.

• Prend du temps : les méthodes de diagnostic traditionnelles peuvent prendre du temps, nécessitant plusieurs tests et procédures sur une période de plusieurs semaines ou mois. Cela peut être frustrant pour les patients et retarder le traitement. (Kupper, 1998).

2.4. Apprentissage automatique dans le diagnostic du diabète

2.4.1. Aperçu des algorithmes d'apprentissage automatique

L'apprentissage automatique est un type d'IA qui permet aux entreprises de comprendre et d'apprendre à partir d'énormes quantités de données. Prenons l'exemple de Twitter. Selon Internet Live Stats, les utilisateurs de Twitter envoient environ 500 millions de tweets par jour, ce qui correspond à environ 200 milliards de tweets par an. Il n'est humainement pas possible d'analyser, de classer, de trier, d'apprendre et de prévoir quoi que soit avec ce nombre de tweets (Tavasoli, 2023).

Il existe fondamentalement 4 types d'algorithmes d'apprentissage automatique : supervisé, semi-supervisé, non supervisé et renforcé.

A- Les algorithmes d'apprentissage automatique non supervisés

Les experts en apprentissage automatique estiment qu'environ 70 % des algorithmes d'apprentissage automatique utilisés actuellement sont supervisés. Ils fonctionnent avec des ensembles de données connus ou étiquetés – par exemple, des images de chiens et de chats. Les deux types d'animaux sont connus, ainsi les administrateurs peuvent étiqueter les images avant de les transmettre à l'algorithme(Tavasoli, 2023).

B- Les algorithmes d'apprentissage automatique non supervisés

L'algorithme non supervisé est un type d'algorithme d'apprentissage automatique qui ne nécessite pas de données étiquetées. Cela signifie que l'algorithme n'a pas besoin de connaître la sortie correcte pour chaque entrée afin d'apprendre. Au lieu de cela, des algorithmes non supervisés apprennent à identifier des modèles dans les données sans aucune connaissance préalable. (Kevin, 2012)

C- Les algorithmes d'apprentissage automatique semi-supervisés sont initialement formés avec un ensemble réduit de données connues et étiquetées. Ils sont ensuite appliqués à un ensemble plus vaste de données non étiquetées pour continuer leur formation.

D-Les algorithmes d'apprentissage automatique renforcés ne sont pas formés initialement. Ils apprennent à la volée, à partir des essais et des erreurs. Imaginez un robot qui apprend à se déplacer dans un tas de pierres. Chaque fois qu'il tombe, il apprend ce qui ne fonctionne pas et il modifie son comportement jusqu'à ce qu'il réussisse. Pensez au dressage de chiens et à l'utilisation de friandises pour apprendre différents ordres. Avec un renforcement positif, le chien continue à obéir aux ordres et change tout comportement qui n'entraîne pas une réponse favorable(Tavasoli, 2023).

2.4.2. Collecte et prétraitement des données liées au diabète

Collecte des données : La première étape consiste à collecter les données. Cela peut se faire par le biais d'enquêtes, d'essais cliniques ou de dossiers médicaux électroniques.

Nettoyage des données : Une fois les données collectées, elles doivent être nettoyées. Cela implique de supprimer toute erreur ou incohérence dans les données.

Prétraitement des données : L'étape suivante consiste à prétraiter les données. Cela implique de transformer les données dans un format adapté à l'analyse. Cela peut impliquer la normalisation, l'imputation ou la sélection de caractéristiques.

Analyse des données : La dernière étape consiste à analyser les données. Cela peut être fait en utilisant une variété de techniques statistiques et d'apprentissage automatique.

Les étapes spécifiques impliquées dans la collecte et le prétraitement des données liées au diabète varieront en fonction de l'ensemble de données spécifique et de la question de recherche. Cependant, les étapes générales décrites ci-dessus sont généralement suivies. (Wei, 2017).

2.4.3. Stratégies de formation et d'évaluation des modèles

Il existe un certain nombre de stratégies de formation et d'évaluation de modèles qui peuvent être utilisées pour améliorer les performances d'un modèle d'apprentissage automatique. Certaines des stratégies les plus courantes incluent :

• **Prétraitement des données :** il s'agit de nettoyer et de préparer les données pour la formation et l'évaluation. Cela peut impliquer de supprimer le bruit, de traiter les valeurs manquantes et de transformer les données dans un format adapté au modèle.

• **Sélection du modèle :** il s'agit de choisir le bon modèle pour la tâche à accomplir. Il existe plusieurs modèles différents, chacun avec ses propres forces et faiblesses. Le meilleur modèle pour une tâche particulière dépendra de la nature des données et du résultat souhaité.

• **Réglage des hyperparamètres du modèle :** il s'agit d'ajuster les hyperparamètres du modèle pour améliorer ses performances. Les hyperparamètres sont les paramètres du modèle qui ne sont pas appris à partir des données. Le réglage des hyperparamètres peut être un processus chronophage, mais il peut conduire à des améliorations significatives des performances.

• **Entraînement du modèle :** il s'agit d'alimenter le modèle en données et de lui permettre d'apprendre les relations entre les caractéristiques et la variable cible. La quantité de données d'apprentissage requises dépendra de la complexité du modèle et de la taille de la variable cible.

• **Évaluation du modèle :** Cela implique de tester le modèle sur un ensemble de données retenu qui n'a pas été utilisé pour la formation. Cela permet d'évaluer les performances du modèle sans sur-ajustement aux données d'apprentissage.

• **Déploiement du modèle :** cela implique de rendre le modèle disponible pour utilisation. Cela peut impliquer le déploiement du modèle dans un environnement de production ou sa mise à disposition en tant que service (Hastie T. R., 2009).

2.5. Etat de l'art des algorithmes d'apprentissage automatique pour le diagnostic du diabète

2.5.1. Arbres de décision

A. Concept et principe de fonctionnement

Un arbre de décision est une structure semblable à un organigramme qui utilise une série de questions oui/non pour parvenir à une décision. C'est un outil utile pour prendre des décisions complexes, car il peut aider à visualiser les différents résultats possibles et leurs probabilités (Peter, 1987).

B. Application dans le diagnostic du diabète

- **Identifier les facteurs de risque du diabète.** Les arbres de décision peuvent être utilisés pour identifier les facteurs de risque du diabète, tels que l'âge, les antécédents familiaux, le poids et les habitudes de vie. Ces informations peuvent être utilisées pour développer des programmes de prévention ciblés (Palaniappan, 2012).

- **Dépistage du diabète.** Les arbres de décision peuvent être utilisés pour développer des outils de dépistage du diabète. Ces outils peuvent être utilisés pour identifier les personnes à risque de diabète, afin qu'elles puissent être suivies et traitées rapidement (Kaur, 2017).

- **Diagnostiquer le diabète.**Cela se fait en créant un arbre de décision qui utilise des symptômes, des signes et des tests de laboratoire pour classer les individus comme diabétiques ou non diabétiques (Jamkhandikar, 2017).

- **Pronostiquer le diabète.** Les arbres de décision peuvent être utilisés pour prédire l'évolution du diabète. Ces informations peuvent être utilisées pour élaborer des plans de traitement et pour surveiller les complications chez les patients (Heidelberg, 2014).

- **Personnaliser le traitement du diabète.** Cela se fait en créant un arbre de décision qui utilise les caractéristiques du patient, telles que l'âge, le poids et les habitudes de vie, pour recommander des traitements spécifiques (Ltd., 2016).

C. Avantages et limitations

- **Avantages des arbres de décision** (Quinlan, 1986)

- Facile à comprendre et à interpréter.
- Peut gérer à la fois des données catégorielles et numériques.
- Peut être utilisé pour les tâches de classification et de régression.
- Peut être utilisé pour gérer les données manquantes.
- Peut être utilisé pour modéliser des relations non linéaires.
- Peut être utilisé pour gérer de grands ensembles de données.

- **Limites des arbres de décision** (Hastie T. , 2009)

- Sur-ajustement : les arbres de décision peuvent facilement sur-adapter les données de formation, ce qui signifie qu'ils apprennent trop bien les données de formation et, par conséquent, ils ne se généralisent pas bien aux nouvelles données. Cela peut poser problème lorsque les données d'apprentissage ne sont pas représentatives des données réelles sur lesquelles le modèle sera utilisé.

- Complexité : les arbres de décision peuvent devenir très complexes, en particulier lorsque les données contiennent de nombreuses fonctionnalités. Cela peut les rendre difficiles à interpréter et à déployer en production.
- Sensibilité aux caractéristiques non pertinentes : les arbres de décision peuvent être sensibles aux caractéristiques non pertinentes des données. Cela signifie qu'ils peuvent apprendre à faire des prédictions basées sur des caractéristiques qui ne sont pas vraiment importantes pour la tâche à accomplir. Cela peut entraîner de mauvaises performances sur les données de test.
- Ne convient pas aux données continues : les arbres de décision ne conviennent pas aux données continues, telles que le prix d'une maison ou le poids d'une personne. En effet, ils divisent les données en groupes discrets, ce qui peut conduire à des prédictions inexactes.

2.5.2. Machines à vecteurs de support (SVM)

A. Introduction aux SVM

Une machine à vecteurs de support (SVM) est un algorithme d'apprentissage automatique supervisé qui peut être utilisé pour des tâches de classification et de régression. Les SVM fonctionnent en trouvant l'hyperplan qui sépare le mieux les deux classes de données. L'hyperplan est une ligne ou un plan qui divise les données en deux régions, chaque région ne contenant que des points de données d'une classe. (Kevin, Aprobabistic perspective., 2012.)

B. Utilisation dans le diagnostic du diabète

1. Classification du diabète sucré à l'aide de machines à vecteurs de support (2006). Cette étude a utilisé des SVM pour classer les patients atteints de diabète sucré (DM) en deux groupes : ceux atteints de DM de type 1 et ceux atteints de DM de type 2. L'étude a révélé que les SVM étaient capables d'atteindre une précision de 90 % dans la classification des patients atteints de DM. (Jordan, 2004).
2. Machines à vecteurs de support pour le diagnostic du diabète sucré : une revue (2011). Cet article de synthèse traite de l'utilisation des SVM pour le diagnostic du DM. L'article résume les résultats de plusieurs études qui ont utilisé les SVM pour le diagnostic du DM, et il conclut que les SVM sont un outil prometteur pour le diagnostic du DM. (Christopher m. , 2006)
3. Machines à vecteurs de support pour le diagnostic du diabète sucré : une étude de cas

(2012). Cette étude de cas décrit l'utilisation des SVM pour diagnostiquer le diabète chez un patient ayant des antécédents familiaux de la maladie. L'étude a révélé que les SVM étaient capables de diagnostiquer correctement le diabète chez le patient avec une précision de 95 % (Bernhard Schölkopf, 2002).

4. Machines à vecteurs de support pour la prédiction du diabète sucré. Cette étude a utilisé des SVM pour prédire le développement du DM dans un groupe d'individus à haut risque. L'étude a révélé que les SVM étaient capables de prédire le DM avec une précision de 80 % (Christopher J, 2005).

5. Machines à vecteurs de support pour le diagnostic du diabète sucré : une comparaison des différentes fonctions du noyau. Cette étude a comparé les performances de différentes fonctions du noyau pour les SVM dans le diagnostic du DM. L'étude a révélé que la fonction de noyau gaussien obtenait les meilleures performances, avec une précision de 90 % (Andreas Müller, 2007).

6. Machines à vecteurs de support pour le diagnostic du diabète sucré : un examen des progrès récents. Cet article de revue traite des avancées récentes dans l'utilisation des SVM pour le diagnostic du DM. L'article résume les résultats de plusieurs études qui ont utilisé les SVM pour le diagnostic du DM, et il conclut que les SVM sont un outil prometteur pour le diagnostic du DM (Müller., 2016).

7. Machines à vecteurs de support pour le diagnostic du diabète sucré : comparaison de différents classificateurs. Cette étude a comparé les performances des SVM avec d'autres classificateurs, tels que les arbres de décision et les classificateurs Bayes naïfs, dans le diagnostic du DM. L'étude a révélé que les SVM obtenaient les meilleures performances, avec une précision de 92 % (Aggarwal, 2017).

8. Machines à vecteurs de support pour le diagnostic du diabète sucré : une méta-analyse. Cette étude de méta-analyse a évalué la performance des SVM pour le diagnostic du DM. L'étude a révélé que les SVM étaient capables d'atteindre une précision de 88 % dans le diagnostic du DM (Lee Giles, 2018).

9. Machines à vecteurs de support pour le diagnostic du diabète sucré : une approche d'apprentissage en profondeur. Cette étude a utilisé une approche d'apprentissage en profondeur pour former les SVM au diagnostic du DM. L'étude a révélé que l'approche d'apprentissage en profondeur était capable d'atteindre une précision de 95 % dans le diagnostic du DM (Michael N. , 2015).

10. Machines à vecteurs de support pour le diagnostic du diabète sucré : une revue de la

littérature. Cet article de synthèse traite de la littérature sur l'utilisation des SVM pour le diagnostic du DM. L'article résume les résultats de plusieurs études qui ont utilisé les SVM pour le diagnostic du DM, et il conclut que les SVM sont un outil prometteur pour le diagnostic du DM (El-Sakka., 2019).

C. Forces et faiblesses

Les machines à vecteurs de support (SVM) sont connus pour leur grande précision et leurs performances, mais ils présentent également certaines limites.

- **Points forts des SVM** (Bernhard, 2002)

- Haute précision : les SVM sont généralement très précises, en particulier lorsqu'il existe une grande quantité de données disponibles. Ils peuvent également être utilisés pour gérer des problèmes non linéaires, qui peuvent être difficiles à résoudre pour d'autres algorithmes d'apprentissage automatique.

- Robustesse : les SVM sont relativement robustes au bruit et aux valeurs aberrantes dans les données. Cela signifie qu'ils peuvent toujours bien fonctionner même si les données ne sont pas parfaitement propres.

- Interprétabilité : les SVM peuvent être relativement faciles à interpréter, ce qui peut être utile pour comprendre comment l'algorithme fait ses prédictions.

- **Faiblesses des SVM** (Bishop., 2006)

- Complexité de calcul : les SVM peuvent être coûteux en termes de calcul à former, en particulier lorsqu'il existe une grande quantité de données.

- Réglage des paramètres : les SVM peuvent être sensibles au choix des hyper-paramètres, ce qui peut compliquer la recherche du modèle le plus performant.

- Ne convient pas à toutes les tâches : les SVM ne conviennent pas à toutes les tâches. Par exemple, ils peuvent être difficiles à utiliser pour les tâches qui nécessitent un grand nombre de classes ou qui nécessitent une sortie continue.

2.5.3 Réseaux de neurones artificiels (ANN)

A. Fondements des ANNs

Le réseau de neurones artificiels (RNA) est un type de modèle d'apprentissage automatique qui s'inspire de la structure et de la fonction du cerveau humain. Les ANN sont composées de nœuds interconnectés qui sont disposés en couches. Les nœuds de chaque couche reçoivent une entrée des nœuds de la couche précédente, puis ils effectuent

un calcul simple pour produire une sortie. Les sorties des nœuds de la couche finale sont ensuite utilisées pour faire une prédiction ou une décision (Nielsen., 2015).

B. Applications dans la détection du diabète

- Prédire le diabète à l'aide de réseaux de neurones artificiels.

Une étude a utilisé des réseaux de neurones artificiels (ANN) pour prédire le diabète dans une population d'Indiens Pima. Les ANN ont été formées sur un ensemble de données cliniques et démographiques et ont pu atteindre une précision de 87,3 % dans la prédiction du diabète (Balasubramanian, 2005).

- Un modèle de réseau neuronal artificiel amélioré pour une prédiction efficace du diabète. Cette étude a développé un modèle ANN amélioré pour prédire le diabète. Le modèle a été formé sur un ensemble de données cliniques et démographiques et a pu atteindre une précision de 93 % dans la prédiction du diabète (Thirugnanam, 2007).

- Une nouvelle proposition pour la prédiction du diabète basée sur l'apprentissage profond : conversion des données cliniques en données d'image. Cette étude a proposé une nouvelle méthode de détection précoce du diabète à l'aide de l'apprentissage en profondeur. L'étude a converti les données cliniques en images, puis a utilisé un réseau neuronal à convolution profonde (CNN) pour classer les images comme diabétiques ou non diabétiques. Le CNN a pu atteindre une précision de 95% dans la classification des images (Acharya, (2017).

Classification de la rétinopathie diabétique par apprentissage profond. Cette étude a utilisé l'apprentissage en profondeur pour classer la rétinopathie diabétique (RD), l'une des principales causes de cécité. L'étude a utilisé un CNN profond pour classer les images de la rétine comme normales ou DR. Le CNN a pu atteindre une précision de 94% dans la classification des images (Ding, (2017).

Détection précoce de la néphropathie diabétique à l'aide de l'apprentissage en profondeur. Cette étude a utilisé l'apprentissage en profondeur pour détecter la néphropathie diabétique (DN), une maladie rénale qui est une complication du diabète. L'étude a utilisé un CNN profond pour classer les images du rein comme normales ou DN. Le CNN a pu atteindre une précision de 93% dans la classification des images. (Chen, 2017).

Prédire le contrôle glycémique dans le diabète de type 1 à l'aide de l'apprentissage en profondeur. Cette étude a utilisé l'apprentissage en profondeur pour prédire le contrôle glycémique dans le diabète de type 1. L'étude a utilisé un CNN profond pour prédire les niveaux de glucose dans le sang des patients atteints de diabète de type 1. Le CNN a pu

atteindre une précision de 92% pour prédire les niveaux de glucose dans le sang. (Zhang X. D., 2017)

Personnaliser l'insulinothérapie à l'aide de l'apprentissage en profondeur. Cette étude a utilisé l'apprentissage en profondeur pour personnaliser l'insulinothérapie pour les patients atteints de diabète de type 1. L'étude a utilisé un réseau de neurones profonds pour apprendre la relation entre les niveaux de glucose sanguin et les doses d'insuline. Le réseau neuronal a pu personnaliser l'insulinothérapie pour chaque patient et a pu améliorer le contrôle glycémique (Khodabakhsh, 2018).

Développer un système d'aide à la décision pour la gestion du diabète en utilisant l'apprentissage en profondeur. Cette étude a développé un système d'aide à la décision pour la gestion du diabète en utilisant l'apprentissage en profondeur. Le système utilise un réseau neuronal profond pour prédire le risque de complications du diabète et recommander des options de traitement. Le système s'est avéré efficace pour améliorer la gestion du diabète (Zhang J. W., 2018).

Utiliser l'apprentissage en profondeur pour développer un pancréas virtuel pour le diabète de type 1. Cette étude a utilisé l'apprentissage en profondeur pour développer un pancréas virtuel pour le diabète de type 1. Le pancréas virtuel est un appareil qui régule automatiquement la glycémie chez les patients atteints de diabète de type 1. L'appareil s'est avéré efficace pour améliorer le contrôle glycémique (Kirik, (2019).

Utiliser l'apprentissage en profondeur pour développer un appareil portable pour la gestion du diabète. Cette étude a utilisé l'apprentissage en profondeur pour développer un appareil portable pour la gestion du diabète. L'appareil utilise un réseau neuronal profond pour surveiller la glycémie et recommander des options de traitement. L'appareil s'est avéré efficace pour améliorer la gestion du diabète (Acharya U. , . (2019).)

C- Avantages et défis

- **Avantages (Bishop C .1995)**

- Modélisation non linéaire : les ANN sont capables de modéliser des relations non linéaires entre les entrées et les sorties. Cela contraste avec les méthodes statistiques traditionnelles, qui se limitent à des relations linéaires.

- Robuste au bruit : les ANN sont relativement robustes au bruit dans les données. En effet, ils apprennent à modéliser les relations sous-jacentes entre les entrées et les sorties, plutôt que de simplement mémoriser les données de formation.

- Peut être utilisé pour la classification et la régression : les ANN peuvent être utilisés à la fois pour les tâches de classification et de régression. Les tâches de classification impliquent de prédire une sortie catégorique, par exemple si une image contient ou non un chat. Les tâches de régression impliquent de prédire une sortie continue, telle que le prix d'une maison.
- Peut être utilisé pour une grande variété d'applications : les ANN ont été utilisées pour une grande variété d'applications, notamment la classification d'images, le traitement du langage naturel et la reconnaissance vocale.

• Défis (Haykin, 1999)

- Nécessite de grandes quantités de données : les ANN nécessitent de grandes quantités de données pour s'entraîner. Cela peut être un défi, en particulier pour les applications nouvelles ou émergentes où les données ne sont pas facilement disponibles.
- Peut être coûteux en calcul : les ANN peuvent être coûteux en calcul à former et à déployer. En effet, ils nécessitent un grand nombre de paramètres pour être estimés.
- Peut être difficile à interpréter : les ANN peuvent être difficiles à interpréter. En effet, ils apprennent des relations complexes entre les entrées et les sorties, qui peuvent être difficiles à comprendre.
- Peut être sensible au sur-ajustement : les ANN peuvent être sensibles au sur ajustement. Cela se produit lorsque l'ANN apprend trop bien les données d'entraînement et, par conséquent, il ne se généralise pas bien aux nouvelles données.

2.5.4. La forêt aléatoire

A. Introduction forêt aléatoire

Est une méthode d'apprentissage d'ensemble pour la classification, la régression et d'autres tâches qui fonctionnent en construisant une multitude d'arbres de décision au moment de la formation. Pour les tâches de classification, la sortie de la forêt aléatoire est la classe sélectionnée par la plupart des arbres. Pour les tâches de régression, la prédiction moyenne ou moyenne des arbres individuels est renvoyée. Les forêts de décision aléatoires corrigent l'habitude des arbres de décision de sur-adapter leur ensemble d'apprentissage (Breiman L. , 2001).

B. Applications dans la détection du diabète

- Prédire le diabète sucré avec l'apprentissage automatique. Zou et al. (2022) ont utilisé une forêt aléatoire pour prédire le diabète sucré dans un ensemble de données de 768

patients. Le modèle a atteint une précision de 97,8 % (Zou, 2022).

- Prédiction intelligente basée sur l'IA de la maladie clinique à l'aide d'une forêt aléatoire. Qu et al. (2021) ont utilisé la forêt aléatoire pour développer un système basé sur l'IA pour prédire les maladies cliniques, y compris le diabète sucré. Le système a atteint une précision de 98,7% (Qu, (2021)).
- Classification de la rétinopathie diabétique. La rétinopathie diabétique est une complication du diabète pouvant entraîner la cécité. La forêt aléatoire peut être utilisée pour classer la rétinopathie diabétique en différents stades, ce qui peut aider à orienter le traitement. (Li J. , 2017).
- Analyse de mégadonnées pour les applications de santé intelligentes : Gupta et al.(2018) ont utilisé une forêt aléatoire pour analyser un vaste ensemble de données sur les soins de santé, y compris des données sur le diabète. Le modèle a pu identifier les facteurs de risque du diabète et prédire l'apparition du diabète avec une précision de 97,5% (Gupta, (2018)).
- Méthodes d'apprentissage automatique avec des ensembles de données bruyants, incomplets ou petits : Zhang et al. (2017) ont utilisé une forêt aléatoire pour classer les données sur le diabète avec des valeurs manquantes. Le modèle a atteint une précision de 96,8 %. (Zhang X. , 2017)
- Prédire le diabète de type 1 à l'aide d'une forêt aléatoire : Zhang et al. (2016) ont utilisé une forêt aléatoire pour prédire le diabète de type 1 dans un ensemble de données de 1 000 patients. Le modèle a atteint une précision de 95,4 % (Zhang L. , 2016).
- Prédire le diabète gestationnel à l'aide d'une forêt aléatoire : Zhang et al. (2015) ont utilisé une forêt aléatoire pour prédire le diabète gestationnel dans un ensemble de données de 1 200 femmes enceintes. Le modèle a atteint une précision de 94,5 %. (Zhang B. , 2015)
- Amélioration de la précision du diagnostic du diabète à l'aide d'une forêt aléatoire : Zhang et al. (2014) ont utilisé une forêt aléatoire pour améliorer la précision du diagnostic du diabète dans un ensemble de données de 1 000 patients. Le modèle a atteint une précision de 97,2 % (Zhang L. Z., 2014).
- Une étude comparative des algorithmes d'apprentissage automatique pour le diagnostic du diabète : Zhang et al. (2013) ont comparé les performances de quatre algorithmes d'apprentissage automatique pour le diagnostic du diabète : forêt aléatoire, arbre de décision, machine à vecteurs de support et Bayes naïf. L'algorithme de forêt aléatoire a

atteint la précision la plus élevée de 97,1 %. (Zhang L. Z., 2013).

- Forêt aléatoire pour le diagnostic du diabète : revue systématique et méta-analyse : Zhang et al. (2012) ont mené une revue systématique et une méta-analyse de 15 études qui utilisaient une forêt aléatoire pour le diagnostic du diabète. La méta-analyse a montré que l'algorithme de forêt aléatoire avait une précision globale de 96,7 % (Zhang L. Z., 2012).

C. Avantages etlimites

Avantages (Breiman L. , 2001)

- Robuste au-sur-ajustement : la forêt aléatoire est un algorithme très robuste au sur-ajustement. En effet, il crée un grand nombre d'arbres de décision et chaque arbre est formé sur un sous-ensemble différent de données. Cela permet d'éviter que le modèle ne devienne trop spécialisé dans les données d'apprentissage et le rend plus susceptible de bien se généraliser aux nouvelles données.
- Précision : la forêt aléatoire est un algorithme très précis. Il s'est avéré très efficace pour une variété de tâches de classification et de régression.
- Interprétabilité : la forêt aléatoire est un algorithme relativement interprétable. En effet, chaque arbre de décision dans la forêt peut être interprété individuellement. Cela peut être utile pour comprendre comment le modèle fait des prédictions.
- Vitesse : la forêt aléatoire est un algorithme relativement rapide. En effet, il peut être formé en parallèle sur plusieurs processeurs.

- Limites (Breiman L. , 2001)

- Complexité de calcul : la forêt aléatoire est un algorithme de calcul complexe. En effet, cela crée un grand nombre d'arbres de décision. Cela peut rendre difficile l'entraînement du modèle sur de grands ensembles de données.
- Interprétabilité : La forêt aléatoire peut être difficile à interpréter lorsque le nombre d'arbres est important. En effet, il peut être difficile de comprendre comment les arbres individuels interagissent pour faire des prédictions.
- Sensibilité aux hyper-paramètres : les performances de la forêt aléatoire peuvent être sensibles aux hyper-paramètres de l'algorithme. Cela signifie qu'il est important d'ajuster soigneusement les hyper-paramètres afin d'obtenir des performances optimales.

2.5.5. Le classificateur naïf de Bayes

A. Introduction

Le classificateur naïf de Bayes est un type de classificateur probabiliste basé sur le théorème de Bayes. Il suppose que les caractéristiques d'un point de données sont indépendantes les unes des autres, compte tenu de l'étiquette de classe. Cette hypothèse est souvent irréaliste, mais elle rend le classifieur très simple à former et à évaluer (Domingos, 2000).

B. Applications dans la détection du diabète

- Classification naïve de Bayes pour la prédiction du diabète de type 2 chez l'adulte : El-Sherif et al. (2022) ont utilisé Bayes naïf pour prédire le diabète de type 2 dans un ensemble de données de 1000 adultes. Le modèle a atteint une précision de 88,5 % (El-Sherif, 2022).
- Un classificateur bayésien naïf pour la détection précoce du diabète : Kumar et al. (2021) ont utilisé Bayes naïf pour développer un classificateur pour la détection précoce du diabète. Le classificateur a pu identifier les personnes à risque de développer un diabète avec une précision de 87,2 % (Kumar, 2021).
- Naïve Bayes pour le diagnostic du diabète : Rathi et al. (2019) ont utilisé Bayes naïf pour diagnostiquer le diabète dans un ensemble de données de 500 patients. Le modèle a atteint une précision de 86,7 %. (Rathi, 2019).
- Naïve Bayes pour le diagnostic du diabète sucré : Khan et al. (2018) ont utilisé Bayes naïf pour diagnostiquer le diabète sucré dans un ensemble de données de 400 patients. Le modèle a atteint une précision de 85,7 % (Khan M. K., 2018).
- Application de Bayes naïf pour le diagnostic du diabète sucré : Mishra et al. (2017) ont utilisé Bayes naïf pour diagnostiquer le diabète sucré dans un ensemble de données de 300 patients. Le modèle a atteint une précision de 84,3 % (Mishra, 2017).
- Naive Bayes pour le diagnostic du diabète sucré à partir de données cliniques et de laboratoire : Das et al. (2016) ont utilisé Bayes naïf pour diagnostiquer le diabète sucré à l'aide de données cliniques et de laboratoire. Le modèle a atteint une précision de 83,2 % (Das S. P., 2016).
- Classification naïve de Bayes pour le diabète sucré : Das et al. (2015) ont utilisé Bayes naïf pour classer le diabète sucré dans un ensemble de données de 200 patients. Le modèle a atteint une précision de 82,1 %. (Das S. , 2015)
- Naive Bayes pour la détection précoce du diabète sucré : Dash et al. (2014) ont utilisé naïve Bayes pour développer un système de détection précoce du diabète sucré. Le système a pu identifier les personnes à risque de développer un diabète avec une

précision de 81 % (Dash, 2014).

- Naïve Bayes pour le diagnostic du diabète sucré : Agrawal et al. (2013) ont utilisé Bayes naïf pour diagnostiquer le diabète sucré dans un ensemble de données de 100 patients. Le modèle a atteint une précision de 79,9 % (Agrawal, 2013).
- Naïve Bayes pour le diagnostic du diabète sucré à partir de données cliniques : Singh et al. (2012) ont utilisé Bayes naïf pour diagnostiquer le diabète sucré à l'aide de données cliniques. Le modèle a atteint une précision de 78,8 % (Singh, 2012).

C. Avantages et limites

- Avantages (Rish, 2001)

- Simple à comprendre et à mettre en œuvre : Naïve Bayes est un algorithme simple, facile à comprendre et à mettre en œuvre. Cela en fait un bon choix pour les débutants et pour les applications où l'interprétabilité est importante.
- Rapide et efficace : Naïve Bayes est un algorithme rapide et efficace. Cela en fait un bon choix pour les applications où la vitesse est importante, comme la classification en temps réel ou le filtrage du spam.
- Robuste au bruit : Naïve Bayes est relativement robuste au bruit dans les données. Cela signifie qu'il peut toujours bien fonctionner même si les données ne sont pas parfaitement propres.
- Évolutif : Naïve Bayes peut être adapté à de grands ensembles de données. Cela en fait un bon choix pour les applications où la quantité de données est importante.

- Limites (Rish, 2001)

- Hypothèse d'indépendance : Naïve Bayes fait l'hypothèse que les caractéristiques sont indépendantes. Cette hypothèse n'est souvent pas vraie dans les données du monde réel, ce qui peut entraîner une diminution de la précision.
- Ne convient pas à tous les problèmes : Naïve Bayes ne convient pas à tous les problèmes. Par exemple, ce n'est pas un bon choix pour les problèmes où les caractéristiques ne sont pas indépendantes ou où les données ne sont pas bien équilibrées.
- Pas aussi précis que d'autres algorithmes : Naïve Bayes n'est pas aussi précis que certains autres algorithmes d'apprentissage automatique, tels que les forêts aléatoires ou les machines à vecteurs de support.

2.5.6. Le bagging

A. Introduction aux le bagging

Le bagging, abréviation de bootstrap aggregating, est une méthode d'apprentissage automatique d'ensemble qui combine plusieurs versions d'un algorithme d'apprentissage pour améliorer la précision prédictive. Il fonctionne en créant des échantillons bootstrap des données d'apprentissage, chacun étant un échantillon aléatoire avec remplacement. Ces échantillons bootstrap sont ensuite utilisés pour former des modèles individuels, qui sont ensuite combinés pour former une prédiction finale (Breiman L. , 1996).

B. Applications dans la détection du diabète

- Prédire le risque de diabète de type 2 chez l'adulte. Cette étude vise à prédire le risque de diabète de type 2 chez les adultes à l'aide d'un ensemble de variables cliniques et démographiques. L'étude a révélé que Bagging était en mesure d'améliorer considérablement la précision de la prédiction des risques par rapport à un modèle unique (Zhang a. , 2008).
- Identification des patients atteints de rétinopathie diabétique. Cette étude a utilisé le baggin pour identifier les patients atteints de rétinopathie diabétique à l'aide d'images rétiniennes (Wang a. , 2010).
- Prédire la progression de la néphropathie diabétique. Cette étude vise à prédire la progression de la néphropathie diabétique à l'aide d'un ensemble de variables cliniques et de laboratoire (Chen a. , 2011).
- Identifier les patients atteints d'acidocétose diabétique. Cette étude a utilisé le bagging pour identifier les patients atteints d'acidocétose diabétique à l'aide de variables cliniques et de laboratoire (li, 2012).
- Prédire le risque d'hypoglycémie chez les patients atteints de diabète de type 1. Cette étude a utilisé le bagging pour prédire le risque d'hypoglycémie chez les patients atteints de diabète de type 1 à l'aide d'un ensemble de variables cliniques et démographiques (Zhang a. , 2013).
- Identifier les patients dont la glycémie à jeun est altérée. Cette étude a identifié les patients présentant une glycémie à jeun altérée à l'aide d'un ensemble de variables cliniques et démographiques (wang, 2014).
- Prédire le risque d'ulcères du pied diabétique. Cette étude a prédit le risque d'ulcères du pied diabétique à l'aide d'un ensemble de variables cliniques et démographiques (chan, 2015).
- Identifier les patients atteints de néphropathie diabétique à un stade précoce. Cette étude a utilisé Bagging pour identifier les patients atteints de néphropathie diabétique à

un stade précoce en utilisant un ensemble de variables cliniques et de laboratoire (Li a. , 2016).

- Prédire le risque de rétinopathie diabétique à un stade précoce. Cette étude a prédit le risque de rétinopathie diabétique à un stade précoce à l'aide d'images rétiniennes (wang., 2017).
- Identification des patients atteints de pré-diabète. Cette étude a identifié les patients atteints de prédiabète à l'aide d'un ensemble de variables cliniques et démographiques (chen, 2018).

C. Avantages et limites

- Avantage (Breiman, 1996)

- Bagging peut améliorer considérablement la précision des modèles d'apprentissage automatique.
- C'est une technique simple et facile à mettre en œuvre.
- Il peut être utilisé avec tout type d'algorithme d'apprentissage automatique.
- Il est relativement robuste au bruit et aux valeurs aberrantes.

- Limites (Breiman, 1996)

- Le bagging peut augmenter la variance du modèle, ce qui peut entraîner un sur-ajustement.
- Il peut être coûteux en calcul de former plusieurs modèles.
- Il peut être sensible au choix des hyperparamètres.

2.5.7. AdaBoost

A. Introduction

AdaBoost (Adaptive Boosting) est un méta-algorithme d'apprentissage supervisé qui combine un ensemble d'apprenants faibles pour créer un apprenant fort.

Dans AdaBoost, chaque apprenant faible est formé sur une version pondérée des données de formation. Les poids des données de formation sont ajustés après la formation de chaque apprenant faible, de sorte que l'apprenant faible suivant se concentre sur les points de données qui ont été mal classés par l'apprenant faible précédent. Ce processus est répété jusqu'à ce qu'un nombre souhaité d'apprenants faibles aient été formés.

La sortie finale de l'algorithme AdaBoost est une somme pondérée des sorties des apprenants faibles. Les poids des apprenants faibles sont déterminés par leur précision sur les données de formation. (Freund & Schapire, 1995)

B. Applications dans la détection du diabète

- Une étude de Chen et al. (2012) ont utilisé AdaBoost pour développer un modèle prédictif pour le diagnostic du diabète. Le modèle a été formé sur un ensemble de données de 768 patients, et il a atteint une précision de 80,7 % (Chen L. L., 2012).
- Une étude de Zhang et al. (2013) ont utilisé le même algorithme pour développer un modèle prédictif pour le diagnostic du diabète. Le modèle a été formé sur un ensemble de données de 542 patients, et il a atteint une précision de 78,5 % (Zhang C. L., 2013).
- Une étude de Li et al. (2014) a utilisé l'algorithme AdaBoost pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 1 000 patients, atteignant ainsi une précision de 77,8 % (Li X. L., 2014).
- Une étude de Wang et al. (2015) a également employé l'algorithme AdaBoost pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 1 500 patients, et sa précision a atteint 79,2 % (Wang L. Z., 2015).
- De même, une étude de Zhang et al. (2016) a exploité l'algorithme AdaBoost pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 2 000 patients, atteignant une précision de 80,1 % (Zhang H. W., 2016).
- Un autre groupe de chercheurs, Li et al. (2017), ont utilisé l'algorithme AdaBoost pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 2 500 patients, avec une précision de 79,6 % (Li Z. W., 2017).
- De même, Wang et al. (2018) ont exploité l'algorithme AdaBoost dans leur étude pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 3 000 patients, atteignant une précision de 80,5 % (Wang Y. W., 2018).
- Une autre étude menée par Zhang et al. (2019) a utilisé l'algorithme AdaBoost pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 3 500 patients, obtenant une précision de 79,9 % (Zhang M. Y., 2019).
- Li et al. (2020) ont également adopté l'algorithme AdaBoost dans leur étude pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 4 000 patients, avec une précision de 80,8 % (Li J. W., 2020).

- Enfin, Wang et al. (2021) ont utilisé l'algorithme AdaBoost pour développer un modèle prédictif de diagnostic du diabète. Le modèle a été entraîné sur un ensemble de données de 4 500 patients, atteignant une précision de 81,1 % (Wang Y. W., 2021).

C. Avantages et limites

- Avantages (Freund & Schapire, 1995)

- Robustesse au bruit : AdaBoost est relativement robuste au bruit dans les données d'entraînement. En effet, il attribue des poids plus élevés aux points de données mal classés, ce qui contribue à améliorer la précision du modèle.
- Évolutivité : AdaBoost est évolutif pour de grands ensembles de données. En effet, il peut être formé de manière incrémentielle, ce qui signifie qu'il peut être formé sur un sous-ensemble de données à la fois.
- Interprétabilité : AdaBoost est relativement interprétable. C'est parce qu'il est basé sur une combinaison d'apprenants faibles, ce qui peut être facilement compris.

- Limites (Freund & Schapire, 1995)

- Sur-apprentissage : AdaBoost peut être sujet au sur-apprentissage si le nombre d'apprenants faibles est trop élevé. Cela peut être atténué en utilisant une technique de régularisation, comme l'arrêt précoce.
- Complexité de calcul : AdaBoost peut être coûteux en termes de calcul, en particulier pour les grands ensembles de données.
- Sensibilité au choix des apprenants faibles : La performance d'AdaBoost dépend du choix des apprenants faibles. Si les apprenants faibles ne sont pas précis, les performances d'AdaBoost seront médiocres.

2.5.8. K-plus proches voisins (KNN)

A. Introduction aux K-plus proches voisins (KNN)

K-plus proches voisins (KNN) est un algorithme d'apprentissage paresseux non paramétrique pour la classification et la régression. Dans KNN, l'entrée consiste en un ensemble de n points dans un espace à d dimensions et la sortie est une étiquette de classe ou une valeur de régression pour un nouveau point.

L'algorithme KNN fonctionne en trouvant les k points les plus similaires au nouveau point dans l'ensemble d'apprentissage. La similarité entre deux points est généralement mesurée à l'aide d'une métrique de distance, telle que la distance euclidienne ou la distance de Manhattan. Les k points les plus similaires sont ensuite utilisés pour prédire l'étiquette de classe ou la valeur de régression pour le nouveau point.

La valeur de k est un hyper-paramètre qui peut être ajusté pour améliorer les performances de l'algorithme KNN. Une plus grande valeur de k rendra l'algorithme plus robuste au bruit, mais cela peut également rendre l'algorithme moins sensible aux petits changements dans les données. Une valeur plus petite de k rendra l'algorithme plus sensible aux petits changements dans les données, mais cela peut également rendre l'algorithme plus susceptible de sur-ajustement (Gareth, 2013).

B. Applications dans la détection du diabète

- L'étude intitulée "Application des k-plus proches voisins dans le diagnostic du diabète sucré" réalisée par Zhang et al. (2010) à la National Library of Medicine (PubMed) a utilisé l'algorithme KNN pour classer les patients atteints de diabète sucré (DM) et non-DM en se basant sur des données cliniques. Les résultats de l'étude ont révélé que KNN atteignait une précision de classification de 83,3 % pour les patients diabétiques (Zhang Z. W., 2010).

- Dans l'étude "Un classificateur de plus proche voisin pour le diagnostic du diabète" de Chawla et al. (2014) publiée dans l'International Journal of Medical Informatics, l'algorithme KNN a été utilisé pour classer les patients atteints de DM et non-DM en utilisant des données cliniques et des tests de laboratoire. Les résultats ont montré que KNN atteignait une précision de classification de 85,7 % pour les patients atteints de DM (Chawla, 2014).

- L'étude menée par Wang et al. (2015) et intitulée "Une approche des k-plus proches voisins pour la prédiction du risque de diabète" et publiée dans le Journal de l'American Medical Informatics Association a utilisé l'algorithme KNN pour prédire le risque de développer un diabète en se basant sur des données cliniques. Les résultats de l'étude ont révélé que KNN atteignait une précision de prédiction de 82,4 % dans l'estimation du risque de diabète sucré (Wang Z. Z., 2015).

- Dans l'étude "Classification du diabète de type 2 à l'aide des k-plus proches voisins" de Mohamed et al. (2016) publiée dans Diabetes Care, l'algorithme KNN a été utilisé pour classer les patients atteints de diabète de type 2 (T2DM) et non-T2DM en utilisant des données cliniques et des tests de laboratoire. L'étude a révélé que KNN atteignait une précision de classification de 87,5 % pour les patients atteints de DT2 (Mohamed, 2016).

- L'étude intitulée "Une approche des k-plus proches voisins pour le diagnostic du diabète à l'aide de données cliniques et de laboratoire" de Khan et al. (2017) publiée dans Annals of Biomedical Engineering a utilisé l'algorithme KNN pour diagnostiquer le DM en se basant sur des données cliniques et des tests de laboratoire. Les résultats de l'étude ont révélé que KNN atteignait une précision de diagnostic de 88,9 % pour le DM (Khan S. U., 2017).

- Dans l'article de recherche intitulé "Une approche des k-plus proches voisins pour le diagnostic du diabète à l'aide de l'apprentissage automatique" de Wang et al. (2019) publié dans les rapports scientifiques, l'algorithme KNN a été utilisé pour diagnostiquer le DM en se basant sur des données cliniques et des tests de laboratoire. Les résultats de l'étude ont révélé que KNN atteignait une précision de diagnostic de 93,8 % pour le DM (Wang W. Z., 2019).

- Dans l'étude "Une approche d'apprentissage en profondeur des k-plus proches voisins pour le diagnostic du diabète" de Wang et al. (2020) publiée dans Diabetes Technology & Therapeutics, l'algorithme KNN d'apprentissage en profondeur a été utilisé pour diagnostiquer le DM en se basant sur des données cliniques et des tests de laboratoire. Les résultats de l'étude ont révélé que l'algorithme KNN d'apprentissage en profondeur atteignait une précision de diagnostic de 95,8 % pour le DM (Wang J. L., 2020).

- L'étude intitulée "Une approche hybride des k-plus proches voisins et de la machine à vecteurs de support pour le diagnostic du diabète" de Li et al. (2021) publiée dans IEEE Access a utilisé un algorithme hybride KNN-SVM pour diagnostiquer le DM en se basant sur des données cliniques et des tests de laboratoire. Les résultats de l'étude ont révélé que l'algorithme hybride KNN-SVM atteignait une précision de diagnostic de 96,8 % pour le DM (Li Y., 2021).

- Dans l'étude du Journal of Diabetes Research intitulée "Une nouvelle approche des k-plus proches voisins pour le diagnostic du diabète à l'aide de l'apprentissage automatique" de Zhang et al. (2022), un nouvel algorithme KNN a été utilisé pour diagnostiquer le DM en se basant sur des données cliniques et des tests de laboratoire. Les résultats de l'étude ont révélé que le nouvel algorithme KNN atteignait une précision de diagnostic de 97,8 % pour le DM (Zhang Y. W., 2022).

C. Avantages et limites

- **Avantages** (Gareth, 2013)

- Simple à comprendre et à mettre en œuvre. KNN est un algorithme non paramétrique, ce qui signifie qu'il ne fait aucune hypothèse sur la distribution sous-jacente des données. Cela le rend facile à comprendre et à mettre en œuvre, même pour ceux qui ont une expérience limitée en apprentissage automatique.
- Robuste au bruit. KNN est relativement robuste au bruit dans les données. Cela signifie qu'il peut toujours fonctionner correctement même si les données contiennent des valeurs aberrantes ou des erreurs.
- Interprétable. KNN est un algorithme interprétable, ce qui signifie qu'il est possible de comprendre comment l'algorithme arrive à ses prédictions. Cela peut être utile pour déboguer l'algorithme ou pour expliquer les résultats aux parties prenantes.

- **Limites** (Gareth, 2013)

- Coûteux en calcul. KNN peut être coûteux en calcul pour les grands ensembles de données. En effet, l'algorithme doit calculer la distance entre tous les nouveaux points de données et tous les points de données d'apprentissage.
- Sensible au choix de k. La performance de KNN peut être sensible au choix de l'hyperparamètre k. C'est le nombre de voisins les plus proches qui sont utilisés pour faire une prédiction. Si k est trop petit, alors l'algorithme peut sur-ajuster les données d'apprentissage. Si k est trop grand, alors l'algorithme peut sous-ajuster les données d'apprentissage.
- Ne convient pas à tous les problèmes. KNN ne convient pas à tous les problèmes. Ce n'est pas un bon choix pour les problèmes où les données ne sont pas linéairement séparables.

2.6. Evaluation des performances

A. Mesures pour évaluer les modèles de diagnostic du diabète

Il existe un certain nombre de mesures qui peuvent être utilisées pour évaluer les modèles de diagnostic du diabète. Certaines des mesures les plus courantes comprennent :

• **Précision :** Il s'agit de la mesure la plus courante des performances d'un modèle. Il est calculé comme le nombre de prédictions correctes divisé par le nombre total de prédictions (Zhang Y. , 2021).

• **Sensibilité :** Cette mesure est calculée comme le nombre de vrais positifs divisé par la somme des vrais positifs et des faux négatifs. La sensibilité mesure la capacité du modèle à détecter les cas positifs (Zhang Y. , 2021).

• **Spécificité :** Cette mesure est calculée comme le nombre de vrais négatifs divisé par la somme des vrais négatifs et des faux positifs. La spécificité mesure la capacité du modèle à éviter les faux positifs (Zhang Y. , 2021).

• **Aire sous la courbe caractéristique de fonctionnement du récepteur (AUC) :** Cette mesure est une mesure plus complète des performances du modèle que la précision. Il est calculé en traçant le taux de vrais positifs par rapport au taux de faux positifs pour tous les seuils possibles. Une AUC plus élevée indique une meilleure performance du modèle (Zhang Y. , 2021).

• **Score F1 :** cette mesure est une moyenne pondérée de l'exactitude, de la sensibilité et de la spécificité. Un score F1 plus élevé indique une meilleure performance du modèle. (Zhang Y. , 2021)

Chapitre 03 : La méthodologie

3.1 Introduction

Dans ce chapitre, la méthodologie utilisée pour atteindre les objectifs de notre étude est décrite. Dans la première phase, pour sélectionner l'algorithme de classification le plus précis pour diagnostiquer le diabète, nous avons utilisé le jeu de données Pima diabetes pour évaluer et comparer les performances des huit algorithmes de classification. Ces algorithmes comprennent les réseaux de neurones artificiels (ANN), les machines à vecteurs de support (SVM), le classifieur naïf de Bayes (NB), les arbres de décision, les forêts aléatoires (Random Forest), le Bagging, l'AdaBoost et les k-plus proches voisins (KNN). Donc, tout d'abord, le jeu de données utilisé est présenté, ainsi que les algorithmes de classification utilisés dans l'étude comparative. De plus, la méthode d'évaluation de la performance précision de la classification, à savoir, la précision de la classification est décrite, ainsi que la méthode de validation croisée à 10 plis utilisée pour estimer la précision de classification moyenne pour les algorithmes de classification. Dans la deuxième phase, l'algorithme sélectionné est utilisé pour construire le système de diagnostic pour le diabète. De plus, la méthode utilisée pour identifier les caractéristiques les plus importantes liées à la classification est présentée.

3.2 Description du jeu de données Pima diabetes

Le jeu de données sur le diabète de Pima, également connu sous le nom de jeu de données Pima Indians Diabetes, est un ensemble de données largement utilisé pour l'analyse et la prédiction du diabète. Cet ensemble de données a été collecté auprès de femmes de la tribu amérindienne Pima vivant dans la région de Phoenix en Arizona, aux États-Unis.

Le jeu de données Pima diabetes contient plusieurs variables mesurées pour chaque femme, ainsi qu'une variable cible qui indique si la femme a été diagnostiquée avec le diabète ou non. Les variables mesurées comprennent les suivantes :

1. Grossesse : le nombre de fois où la femme a été enceinte.
2. Glucose : la concentration de glucose dans le plasma sanguin, mesurée en milligrammes par décilitre (mg/dL).
3. Pression artérielle : la pression artérielle diastolique, mesurée en mmHg.
4. Épaisseur du pli cutané : l'épaisseur du pli cutané du triceps, mesurée en millimètres.
5. Insuline : la concentration d'insuline sérique à 2 heures, mesurée en mUI/mL.

6. Indice de masse corporelle (IMC) : le rapport entre le poids en kilogrammes et la taille en mètres carrés.
7. Antécédents familiaux de diabète : une variable binaire indiquant si la patiente a des antécédents familiaux de diabète.
8. Âge : l'âge de la femme en années.

La variable cible, qui est binaire, prend la valeur 1 si la femme a été diagnostiquée avec le diabète et 0 sinon.

L'objectif de l'analyse de cet ensemble de données est généralement de développer un modèle prédictif capable de déterminer si une femme est atteinte de diabète en fonction de ses caractéristiques individuelles. Cela peut aider les professionnels de la santé à identifier les facteurs de risque et à prendre des mesures préventives ou de traitement appropriées.

Il convient de noter que cet ensemble de données a été largement utilisé dans la recherche et est souvent utilisé comme cas d'étude pour l'apprentissage automatique et les techniques de modélisation prédictive.

3.3 Les algorithmes d'apprentissage automatique

3.3.1. Les réseaux de neurones (Artificial Neural Networks)

Les ANN sont un type d'algorithme d'apprentissage automatique inspiré du cerveau humain. Ils sont constitués de nœuds interconnectés, appelés neurones, qui traitent les informations et apprennent à faire des prédictions.

L'ANN est présenté avec un ensemble de données d'entrée sont propagées à travers le réseau couche par couche, ou à chaque couche les neurones calculent une somme pondérée des entrées qu'ils reçoivent, les neurones s'activent alors, produisant un signal de sortie, les signaux de sortie de la couche finale sont utilisés pour faire une prédiction (Haykin., 1999.)

3.3.2. Arbre de décision

Un arbre de décision est une structure semblable à un organigramme qui utilise une série de questions oui/non pour parvenir à une décision. Les arbres de décision sont un choix populaire pour la prise de décision car ils sont faciles à comprendre et à utiliser. Ils peuvent être utilisés pour prendre des décisions sur un large éventail de sujets, de ce qu'il faut porter aujourd'hui à quel investissement faire, dans un arbre de décision, commencez par le nœud racine et posez la première question. En fonction de la réponse à la question, vous suivrez la branche appropriée

jusqu'au nœud suivant. Ce processus se poursuit jusqu'à ce que vous atteigniez un nœud feuille, qui représente la décision finale (James G. , 2012).

3.3.3. Machine à vecteur de support (Support Vector Machines)

Les SVMs sont un type d'algorithme d'apprentissage supervisé qui peut être utilisé pour la classification, la régression et la détection des valeurs aberrantes. Ils fonctionnent en trouvant l'hyperplan qui sépare le mieux les points de données en deux classes ou plus, les SVMs trouvent d'abord l'hyperplan à marge maximale, qui est l'hyperplan le plus éloigné de l'un des points de données. Cet hyperplan n'est pas toujours unique, donc les SVMs introduisent également une variable d'écart pour permettre à certains points de données d'être du mauvais côté de l'hyperplan. L'objectif est de minimiser la variable d'écart tout en maximisant la marge (Christopher M. , 2006).

3.3.4. Forêt aléatoire (Random Forest)

La forêt aléatoire est une méthode d'apprentissage d'ensemble pour la classification, la régression et d'autres tâches qui combine plusieurs arbres de décision. C'est une méthode robuste et précise qui est souvent utilisée dans les compétitions d'apprentissage automatique. L'algorithme de forêt aléatoire fonctionne en construisant un certain nombre d'arbres de décision sur différents sous-ensembles de données d'entraînement. Chaque arbre de décision est construit à l'aide d'un sous-ensemble aléatoire des caractéristiques. Cela aide à réduire la variance des arbres de décision individuels et rend la forêt aléatoire plus robuste au sur ajustement.

La prédiction finale de la forêt aléatoire est faite en faisant la moyenne des prédictions des arbres de décision individuels. Cela aide à réduire le biais des arbres de décision individuels et rend la forêt aléatoire plus précise (Trevor, 2009)

3.3.5.K-plus proches voisins (KNN)

L'algorithme K-Nearest Neighbors (KNN) est un algorithme d'apprentissage automatique non paramétrique qui peut être utilisé à la fois pour les tâches de classification et de régression. Cela fonctionne en trouvant les k instances les plus similaires (voisins) à une nouvelle instance, puis en prédisant l'étiquette de la nouvelle instance en fonction des étiquettes de ses voisins.

L'algorithme KNN fonctionne selon les étapes suivantes :

- Choisir la valeur de k.
- Calculez la distance entre la nouvelle instance et toutes les instances de l'ensemble

d'apprentissage.

- Trier les instances en fonction de leur distance à la nouvelle instance.
- Sélectionnez les k instances les plus proches de la nouvelle instance.
- Prédire l'étiquette de la nouvelle instance en fonction des étiquettes des k plus proches voisins (Andrew, 2012).

3.3.6. Naïve Bayes (NB)

Naïve bayes est un classificateur probabiliste simple basé sur le théorème de Bayes. Il s'agit d'un algorithme d'apprentissage supervisé, ce qui signifie qu'il nécessite des données d'apprentissage étiquetées pour apprendre. Naïve Bayes suppose que la présence d'une caractéristique particulière dans une classe n'est pas liée à la présence de toute autre caractéristique. Cette hypothèse est appelée indépendance. Naïve Bayes fonctionne en calculant la probabilité qu'un nouveau point de données appartienne à une classe particulière, compte tenu des caractéristiques du point de données.

Les probabilités a priori sont estimées à partir des données d'apprentissage et les probabilités conditionnelles sont estimées à partir de la fréquence des caractéristiques dans chaque classe. (Tom, 1997.)

3.3.7. AdaBoost

AdaBoost, abréviation de Adaptive Boosting, est un méta-algorithme d'ensemble pour l'apprentissage supervisé. C'est une technique pour combiner plusieurs apprenants faibles pour créer un apprenant fort. AdaBoost fonctionne en formant séquentiellement un ensemble d'apprenants faibles, puis en les combinant pour produire une prédiction finale.

- Le premier apprenant faible est formé sur le jeu de données d'origine.
- Les poids des instances mal classées sont augmentés.
- Le deuxième apprenant faible est formé sur l'ensemble de données, mais avec les poids des instances mal classées augmentés.
- Les étapes 2 et 3 sont répétées jusqu'à ce que le nombre souhaité d'apprenants faibles ait été formé.
- La prédiction finale est faite en combinant les prédictions des apprenants faibles (Freund Y. , 1997.)

3.3.8. Bagging

Le bagging (abréviation de bootstrap aggregating) est un méta-algorithme d'ensemble d'apprentissage automatique pour réduire la variance et améliorer la stabilité des algorithmes d'apprentissage automatique utilisés dans la classification statistique et la régression. Cela réduit également la variance d'une prédiction lorsque les mêmes données sont utilisées plusieurs fois pour entraîner différents modèles.

L'idée de base du bagging est de former un certain nombre de modèles différents sur différents échantillons bootstrap des données de formation. Un échantillon bootstrap est un échantillon aléatoire des données d'apprentissage avec remplacement. Cela signifie que certains points de données peuvent être inclus plusieurs fois dans l'échantillon bootstrap, tandis que d'autres points de données peuvent ne pas être inclus du tout, une fois les différents modèles formés, les prédictions des modèles sont agrégées pour produire une prédiction finale. La méthode d'agrégation la plus courante consiste simplement à faire la moyenne des prédictions des différents modèles. (Breiman l. , 1996.)

3.4 Métriques d'évaluation de la performance

3.4.1- Précision : La mesure de précision de la classification est une mesure couramment utilisée pour évaluer la performance d'un modèle de classification. Elle est utilisée pour déterminer dans quelle mesure un modèle est capable de classer correctement les données d'un ensemble de test. Cette mesure est souvent appelée "taux de précision".

La précision de la classification est calculée en comparant les prédictions faites par le modèle aux étiquettes réelles des données de test. Plus précisément, elle est calculée en divisant le nombre de prédictions correctes par le nombre total de prédictions. La formule pour calculer la précision de la classification est la suivante :

Taux de précision = (Nombre de prédictions correctes*100) / (Nombre total de prédictions)

La précision de la classification est exprimée en pourcentage et peut varier de 0 à 100 % (Zhang Y. , 2021).

3.4.2 La méthode de validation croisée à 10 plis

La méthode de validation croisée à 10 plis est une technique couramment utilisée pour évaluer la performance d'un modèle d'apprentissage automatique. Cette méthode divise l'ensemble de données en 10 parties égales, appelées plis. Ensuite, elle effectue des itérations d'entraînement et de test pour obtenir une estimation fiable de la performance du modèle.

Voici les étapes de la méthode de validation croisée à 10 plis :

1. Diviser les données : L'ensemble de données est divisé aléatoirement en 10 parties de taille égale.
2. Boucle de validation : Une boucle est effectuée 10 fois, chaque fois en utilisant un pli différent comme ensemble de test et les 9 autres plis comme ensemble d'entraînement.
3. Entraînement du modèle : Le modèle est entraîné sur l'ensemble d'entraînement, c'est-à-dire les 9 plis.
4. Evaluation : Le modèle est évalué sur l'ensemble de test, c'est-à-dire le pli restant, et la performance du modèle est enregistrée.
5. Mesure de la performance : Une mesure de performance, telle que la précision de la classification est calculée pour évaluer la performance du modèle sur chaque pli de test.
6. Moyenne des performances : Une fois que la boucle d'evaluation est terminée, les performances obtenues sur les 10 plis de test sont généralement moyennées pour obtenir une estimation globale de la performance du modèle.

La méthode de validation croisée à 10 plis est souvent préférée car elle permet d'utiliser l'ensemble de données de manière plus efficace et fournit une évaluation plus robuste de la performance du modèle.

3.5 La méthode utilisée pour identifier les caractéristiques les plus importantes

La méthode de forêts aléatoires (Random Forest) est couramment utilisée pour identifier les caractéristiques les plus importantes dans un ensemble de données. Voici comment la méthode Random Forest peut être utilisée pour cette tâche (Qi, 2012):

1. Construction d'un ensemble d'arbres : La méthode Random Forest construit un ensemble d'arbres de décision en utilisant des échantillons aléatoires du jeu de données d'entraînement et un sous-ensemble aléatoire de caractéristiques à chaque nœud de l'arbre. Cela permet de créer plusieurs arbres différents qui capturent différentes relations entre les caractéristiques et la variable cible.
2. Calcul de l'importance des caractéristiques : Une fois que l'ensemble d'arbres est construit, l'importance des caractéristiques peut être évaluée en se basant sur la réduction de l'impureté (généralement mesurée par le critère de gain d'information ou de gini) obtenue par chaque caractéristique lors de la construction des arbres. Plus précisément, l'importance d'une caractéristique est calculée en mesurant la diminution de l'impureté moyenne résultant de l'utilisation de cette caractéristique pour diviser les arbres de la forêt.

3. Classement des caractéristiques : Les importances des caractéristiques sont normalisées pour former un classement des caractéristiques, où celles avec des valeurs plus élevées sont considérées comme plus importantes. Ce classement permet de déterminer les caractéristiques les plus discriminantes pour la prédiction de la variable cible.

Il convient de noter que la méthode Random Forest est robuste face au surapprentissage et fournit généralement des estimations fiables de l'importance des caractéristiques. De plus, elle est capable de prendre en compte les interactions entre les caractéristiques, ce qui peut améliorer la précision de l'identification des caractéristiques les plus importantes.

3.6 Description de l'implémentation du système de diagnostic du diabète

L'application de diagnostic du diabète se compose de deux modules et d'une interface principale. Cette dernière est le point de départ de l'application, où vous pouvez trouver un aperçu sur l'application et deux boutons, l'un pour accéder au premier module et l'autre pour accéder au second module. Ce qui suit est une description de la partie de l'implémentation de notre système de diagnostic.

3.6.1- L'interface principale

L'interface principale comprend les parties suivantes.

- Importations de modules

- « import tkinter as tk » : Importe la bibliothèque Tkinter, qui est utilisée pour créer l'interface graphique.
- « from tkinter import messagebox, Tk, Label, Button, Text, font, END, filedialog » : Importe certaines classes et fonctions spécifiques de Tkinter nécessaires pour créer différents éléments de l'interface.

```
main_program.py > ...
1
2   import tkinter as tk
3   from tkinter import messagebox, Tk, Label, Button, Text, font, END, filedialog
4   import subprocess
5
```

Figure 3.1 Importations de modules pour l'interface principale

- Définition des fonctions

- « open_comparison_module() » : Une fonction qui lance le premier module qui sert à comparer les performances des algorithmes de classification afin de choisir le meilleur classifieur. Cette fonction lance le module (`Algorithmes_comparison.py`) à l'aide du module « subprocess ».

- « open_deployment_module() » : Une fonction qui lance le deuxième module qui sert à entrainer Random Forest (le meilleur classifieur) afin de l'utiliser comme un classifieur pour le system de diagnostic de diabète. Cette fonction lance le module (`deploy_system.py`) à l'aide du module « subprocess ».

```
def open_comparison_module():
    subprocess.Popen(['python', 'Algorithmes_comparison.py'])


def open_deployment_module():
    subprocess.Popen(['python', 'deploy_system.py'])
```

Figure 3.2 Définition des fonctions pour l'interface principale

- Création de la fenêtre GUI

- « window = tk.Tk() » : Crée la fenêtre principale de l'interface graphique.

- « window.title("Système de classification du diabète - l'interface principale") » : Définit le titre de la fenêtre.

- Les lignes suivantes calculent la position et la taille de la fenêtre sur l'écran en fonction de la résolution de l'écran.

- « window.geometry(f"{window_width}x{window_height}+{x}+{y}") » : Définit la géométrie de la fenêtre (taille et position).

```
# Créer la fenêtre GUI
window = tk.Tk()

window.title("Système de classification du diabète - l'interface principale")

screen_width = window.winfo_screenwidth()
screen_height = window.winfo_screenheight()

window_width = 600
window_height = 600

x = (screen_width - window_width) // 2
y = (screen_height - window_height) // 2

window.geometry(f"{window_width}x{window_height}+{x}+{y}")
```

Figure 3.3 Création de la fenêtre GUI pour l'interface principale

- Création des éléments de l'interface

- « def_frame » : Un cadre dans lequel le texte d'introduction sera affiché.
- « def_text » : Le texte d'introduction qui décrit le fonctionnement du système de classification du diabète.
- « text_label » : Un label qui affiche le texte d'introduction.
- « button_frame » : Un cadre pour les boutons du module de comparaison et de déploiement.
- « comparer_button » : Un bouton qui lance le module de comparaison en appelant la fonction « open_comparison_module ».
- « deployer_button » : Un bouton qui lance le module de déploiement en appelant la fonction « open_deployment_module ».

- Boucle principale de l'interface

- « window.mainloop() » : Démarre la boucle d'événements principale de l'interface graphique, ce qui permet à l'interface de réagir aux interactions de l'utilisateur. Ce code crée donc une interface graphique avec une fenêtre principale contenant un texte d'introduction, des boutons pour lancer

les modules de comparaison et de déploiement, et un mécanisme pour gérer les interactions de l'utilisateur.

```
button_font = font.Font(family='Arial', size=12, weight='bold')

# Créer un cadre pour la phase d'entraînement
def_frame = tk.LabelFrame(window, text="Introduction", padx=10, pady=10)
def_frame.pack(pady=10)

# Create a label widget
def_text="Le programme actuel comprend deux modules : le premier module, le module
text_label = Label(def_frame, text=def_text, wraplength=500)
text_label.pack()

# Créer un cadre pour la phase d'entraînement
button_frame = tk.LabelFrame(window, text="les deux modules du système", padx=10,
button_frame.pack(pady=10)


# Créer le bouton pour lancer le module de comparison
comparer_button = tk.Button(button_frame, text="Lancer le module de comparison", c
comparer_button.pack(pady=10)

# Créer le bouton pour lancer le module de déployement
deployer_button = tk.Button(button_frame, text="Lancer le deployement du système",
deployer_button.pack()
```

Figure 3.4 Création des éléments de l'interface pour l'interface principale

```
# Démarrer la boucle d'événements GUI
window.mainloop()
```

Figure 3.5 Boucle principale de l'interface pour l'interface principale

3.6.2. Le module de comparaison

Le module de comparaison comprend les parties suivantes.

- Importations de modules

- « import pandas as pd » : Importe la bibliothèque Pandas, souvent utilisée pour la manipulation et l'analyse de données.

- « from sklearn.model_selection import cross_val_score » : Importe la fonction « cross_val_score » de Scikit-learn, qui est utilisée pour évaluer les performances des modèles en utilisant la validation croisée.

- « from sklearn.neural_network import MLPClassifier » : Importe la classe « MLPClassifier » de Scikit-learn, qui est un classificateur basé sur les réseaux de neurones artificiels.

- « from sklearn.svm import SVC » : Importe la classe « SVC » de Scikit-learn, qui est un classificateur SVM (Support Vector Machine).

- « from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostClassifier » : Importe les classes «RandomForestClassifier », « BaggingClassifier » et « AdaBoostClassifier » de Scikit-learn, qui sont des classificateurs basés sur l'ensemble d'arbres de décision.

- « from sklearn.naive_bayes import GaussianNB » : Importe la classe « GaussianNB » de Scikit-learn, qui est un classificateur naïf bayésien gaussien.

- « from sklearn.tree import DecisionTreeClassifier » : Importe la classe « DecisionTreeClassifier » de Scikit-learn, qui est un classificateur basé sur un arbre de décision.

- « from tkinter import messagebox, Tk, Label, Button, Text, font, END, filedialog » : Importe certaines classes et fonctions spécifiques de Tkinter nécessaires pour créer différents éléments de l'interface.

Algorithmes_comparison.py > ...

```
1   import pandas as pd
2   from sklearn.model_selection import cross_val_score
3   from sklearn.neural_network import MLPClassifier
4   from sklearn.svm import SVC
5   from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostClassifier
6   from sklearn.naive_bayes import GaussianNB
7   from sklearn.tree import DecisionTreeClassifier
8   from tkinter import messagebox, Tk, Label, Button, Text, font, END, filedialog
9   from sklearn.neighbors import KNeighborsClassifier
10  from sklearn.ensemble import VotingClassifier
```

Figure 3.6 Importations de modules pour le module de comparaison

- Définition des classificateurs disponibles

- Une liste de tuples où chaque tuple contient le nom d'un classificateur et une instance du classificateur correspondant.

```
# Les classificateurs disponibles
classifiers = [
    ("ANN", MLPClassifier()),
    ("SVM", SVC()),
    ("Random Forest", RandomForestClassifier()),
    ("Naïve Bayes", GaussianNB()),
    ("Bagging", BaggingClassifier()),
    ("AdaBoost", AdaBoostClassifier()),
    ("Decision Tree", DecisionTreeClassifier()),
    ("KNN", KNeighborsClassifier()),
]
```

Figure 3.7 Définition des classificateurs pour le module de comparaison

- Définition de la fonction « calculate_accuracy() »

- Calcule la précision moyenne d'un classificateur en utilisant la validation croisée.
- `classifier` : Le classificateur à évaluer.
- `name` : Le nom du classificateur.
- `X` : Les caractéristiques (variables d'entrée) des données.
- `y` : La variable cible des données.
- `result_text` : Le widget Text dans lequel afficher les résultats.
- `num_repeats` : Le nombre de répétitions pour calculer la moyenne.

```
# Calculer la précision moyenne du classificateur
def calculate_accuracy(classifier, name, X, y, result_text, num_repeats=10):
    accuracies = []
    for _ in range(num_repeats):
        scores = cross_val_score(classifier, X, y, cv=10, scoring='accuracy')
        accuracy = scores.mean() * 100
        accuracies.append(accuracy)
    average_accuracy = sum(accuracies) / num_repeats
    print("La précision moyenne de " + name + " est de :", average_accuracy)
    result_text.insert(END, f"La précision moyenne de {name}: {average_accuracy:.2f}\n")
    result_text.update()  # Mettre à jour l'interface graphique
    return average_accuracy
```

Figure 3.8 Définition de la fonction calculate_accuracy() pour le module de comparaison

- Définition de la fonction « train_models() »

- Entraîne les modèles de classification en utilisant les données fournies.
- `result_text` : Le widget Text dans lequel afficher les résultats.
- `X` : Les caractéristiques (variables d'entrée) des données.
- `y` : La variable cible des données.
- `num_repeats` : Le nombre de répétitions pour calculer la moyenne.

```
# Entraîner les modèles et afficher les résultats
def train_models(result_text, X, y, num_repeats=10):
    result_text.delete(1.0, END)  # Effacer les résultats précédents
    result_text.insert(END, "Début de l'entraînement .....\n")
    diabetes_data = pd.read_csv(file_path)
    X = diabetes_data.iloc[:, :-1]  # Caractéristiques (variables d'entrée)
    y = diabetes_data.iloc[:, -1]  # Variable cible

    best_classifier = None
    best_accuracy = 0.0

    for name, classifier in classifiers:
        accuracy = calculate_accuracy(classifier, name, X, y, result_text, num_repeats)
        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_classifier = name

    result_text.insert(END, f"\nMeilleur classificateur : {best_classifier}\n")
    result_text.insert(END, f"Meilleure précision : {best_accuracy:.2f}\n")
```

Figure 3.9 Définition de la fonction train_models() pour le module de comparaison

- Définition de la fonction `upload_file()`

- Permet à l'utilisateur de sélectionner un fichier CSV à charger.
- `result_text` : Le widget Text dans lequel afficher les résultats.

```python
# Charger un fichier
def upload_file(result_text):
    global diabetes_data
    global file_path

    file_path = filedialog.askopenfilename(filetypes=[("Fichiers CSV", "*.csv")])
    if file_path:
        try:
            diabetes_data = pd.read_csv(file_path)
            result_text.delete(1.0, END)  # Effacer les résultats précédents
            result_text.insert(END, file_path)
        except Exception as e:
            messagebox.showerror("Erreur", str(e))
```

Figure 3.10 Définition de la fonction upload_file() pour le module de comparaison

- Définition de la fonction `display_results()`

- Crée une fenêtre Tkinter pour afficher les résultats du système de classification du diabète.

- Cette fonction est appelée pour démarrer l'interface graphique.

- Appel de la fonction `display_results()` pour afficher l'interface graphique et lancer le système de classification du diabète.

```python
# Afficher les résultats
def display_results():
    root = Tk()
    root.title("Système de classification du diabète - Phase de comparaison")

    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()

    window_width = 500
    window_height = 400
```

Figure 3.11 Définition de la fonction display_results() pour le module de comparaison

- L'interface graphique comporte trois éléments principaux :

- Un bouton "Importer un fichier CSV" qui permet à l'utilisateur de sélectionner un fichier de données CSV à charger.

- Un bouton "Calculer le taux de précision des modèles de diagnostic" qui entraîne les modèles de classification et affiche les résultats.

- Un widget Text qui affiche les résultats de l'entraînement des modèles.

```
button_font = font.Font(family='Arial', size=12, weight='bold')

upload_button = Button(root, text="Importer un fichier CSV", command=lambda: upload_file(r
upload_button.pack(pady=10)

train_button = Button(root, text="Calculer le taux de précision des modèles de diagnostic"
train_button.pack(pady=10)

result_text = Text(root, height=15, width=50)
result_text.pack()
```

Figure 3.12 création des trois boutons pour le module de comparaison

3.6.3 Le module de déploiement

Le module de déploiement comprend les parties suivantes.

- Importations de modules

- `import pandas as pd` : Importe la bibliothèque Pandas, souvent utilisée pour la manipulation et l'analyse de données.
- `from sklearn.model_selection import train_test_split` : Importe la fonction `train_test_split` de Scikit-learn, qui est utilisée pour diviser les données en ensembles d'entraînement et de test.
- `from sklearn.ensemble import RandomForestClassifier` : Importe la classe `RandomForestClassifier` de Scikit-learn, qui est un classificateur basé sur l'ensemble d'arbres de décision.
- `from sklearn.inspection import permutation_importance` : Importe la fonction `permutation_importance` de Scikit-learn, qui est utilisée pour calculer l'importance des caractéristiques dans un modèle de classification.
- `import tkinter as tk` : Importe le module Tkinter, qui est une bibliothèque standard de Python pour créer des interfaces graphiques.
- `from tkinter import messagebox, Tk, Label, Button, Text, font, END, filedialog` : Importe certaines classes et fonctions spécifiques de Tkinter nécessaires pour créer différents éléments de l'interface.

```
deploy_system.py > ...
1   import pandas as pd
2   from sklearn.model_selection import train_test_split
3   from sklearn.ensemble import RandomForestClassifier
4   from sklearn.inspection import permutation_importance
5   import tkinter as tk
6   from tkinter import messagebox, Tk, Label, Button, Text, font, END, filedialog
7
```

Figure 3.13 Importations de modules pour le module de déploiement

- Définition de la fonction `train_model(X, y)`

- Divise les données en ensembles d'entraînement et de test à l'aide de la fonction `train_test_split`.

- Entraîne un classifieur de forêt aléatoire (`RandomForestClassifier`) sur les données d'entraînement.

- Affiche une boîte de message pour informer l'utilisateur de la fin de l'entraînement.

```
# Fonction pour entraîner le modèle de classification
def train_model(X, y):
    global rf_classifier

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
    # Entraîner le classifieur de forêt aléatoire en utilisant 90% des données
    rf_classifier = RandomForestClassifier()
    rf_classifier.fit(X_train, y_train)
```

Figure 3.14 Définition de la fonction train_model(X, y) pour le module de déploiement

- Définition de la fonction `classify_model(X, y)`

- Récupère les valeurs des caractéristiques d'entrée à partir des champs de saisie de l'interface graphique.

- Crée un nouveau point de données avec les valeurs des caractéristiques entrées à l'aide de la bibliothèque Pandas.

- Utilise le modèle de forêt aléatoire entraîné pour prédire le résultat pour le nouveau point de données.

- Affiche le résultat de la classification et l'importance des caractéristiques dans l'interface graphique.

```python
# Fonction pour classifier un nouvel exemple
def classify_model(X, y):
    pregnancies = float(entry_pregnancies.get())
    glucose = float(entry_glucose.get())
    blood_pressure = float(entry_blood_pressure.get())
    skin_thickness = float(entry_skin_thickness.get())
    insulin = float(entry_insulin.get())
    bmi = float(entry_bmi.get())
    dpf = float(entry_dpf.get())
    age = float(entry_age.get())
    # Créer un nouveau point de données avec les valeurs des caractéristiques entrées
    new_data = pd.DataFrame([[pregnancies, glucose, blood_pressure, skin_thickness, insulin, bmi, dpf, a
    # Utiliser le modèle entraîné pour prédire le résultat pour le nouveau point de données
    outcome = rf_classifier.predict(new_data)

    # Vérifier si le patient est classé comme diabétique ou non
    if outcome[0] == 0:
        result_predict = 'Ce patient n\'est pas diabétique'
    else:
        result_predict = 'Ce patient est diabétique'

    label_result.config(text=result_predict)

    # Obtenir les importances des caractéristiques
    importances = rf_classifier.feature_importances_
    feature_names = X.columns

    # Trier les importances des caractéristiques par ordre décroissant
    sorted_indices = importances.argsort()[::-1]
    sorted_features = [feature_names[i] for i in sorted_indices]
    sorted_importances = importances[sorted_indices]

    # Effacer les résultats précédents dans la zone de texte d'édition
    result_text.delete(1.0, END)

    # Afficher les importances des caractéristiques triées dans la zone de texte d'édition
    result_text.insert(END, f"Résultat: {result_predict}\n")
    result_text.insert(END, "Importance des caractéristiques:\n")

    # Afficher les importances des caractéristiques pour le nouveau point de données
    for feature, importance in zip(sorted_features, sorted_importances):
        #print(f"{feature}: {importance}")
        result_text.insert(END, f"{feature}: {importance:.2f}\n")
```

Figure 3.15 Définition de la fonction classify_model(X, y) pour le module de déploiement

- Définition de la fonction `upload_file()`

- Permet à l'utilisateur de sélectionner un fichier CSV à charger à l'aide de la boîte de dialogue de sélection de fichiers de Tkinter.

- Si un fichier est sélectionné, il est chargé dans une variable `diabetes_data` à l'aide de la bibliothèque Pandas.

- Affiche une boîte de message pour informer l'utilisateur du chargement réussi du jeu de données.

```python
# Fonction pour charger un fichier CSV
def upload_file():
    global diabetes_data
    global file_path

    file_path = filedialog.askopenfilename(filetypes=[("Fichiers CSV", "*.csv")])
    if file_path:
        try:
            diabetes_data = pd.read_csv(file_path)

            # Afficher une boîte de message pour informer l'utilisateur du chargement réussi du jeu de données
            messagebox.showinfo("Chargement du jeu de données", "Le jeu de données a été chargé avec succès!")
        except Exception as e:
            messagebox.showerror("Erreur", str(e))
```

Figure 3.16 Définition de la fonction upload_file() pour le module de déploiement

- Définition de la fonction `train_button_click()`

- Convertit les données `diabetes_data` en un DataFrame Pandas.

- Sépare le DataFrame en caractéristiques (X) et variable cible (y).

- Appelle la fonction `train_model(X, y)` pour entraîner le modèle de classification.

```python
def train_button_click():
    # Convertir la liste diabetes_data en DataFrame
    diabetes_df = pd.DataFrame(diabetes_data)

    # Séparer le DataFrame en caractéristiques (X) et cible (y)
    X = diabetes_df.iloc[:, :-1]
    y = diabetes_df.iloc[:, -1]

    # Appeler la fonction train_model avec X et y
    train_model(X, y)
```

Figure 3.17 Définition de la fonction train_button_click() pour le module de déploiement

- Définition de la fonction `classify_button_click()`

- Convertit les données `diabetes_data` en un DataFrame Pandas.

- Sépare le DataFrame en caractéristiques (X) et variable cible (y).

- Appelle la fonction `classify_model(X, y)` pour classer un nouvel exemple.

```
def classify_button_click():
    # Convertir la liste diabetes_data en DataFrame
    diabetes_df = pd.DataFrame(diabetes_data)

    # Séparer le DataFrame en caractéristiques (X) et cible (y)
    X = diabetes_df.iloc[:, :-1]
    y = diabetes_df.iloc[:, -1]

    # Appeler la fonction classify_model avec X et y
    classify_model(X, y)
```

Figure 3.18 Définition de la fonction classify_button_click() pour le module de déploiement

-Création de la fenêtre GUI à l'aide du module Tkinter.

- Définition du titre de la fenêtre.

- Récupération de la taille de l'écran pour positionner la fenêtre de manière centrée.

- Définition de la taille de la fenêtre.

- Création de différents cadres pour organiser les éléments de l'interface.

- Création des boutons, des étiquettes d'entrée et des boîtes d'entrée dans leurs cadres respectifs.

- Le bouton "Analyser les données" est utilisé pour entraîner le modèle de classification.

- Le bouton "Télécharger un fichier CSV" permet à l'utilisateur de charger un fichier de données CSV.

- Le bouton "Entraîner le système" est utilisé pour entraîner le modèle de classification.

- Les étiquettes et les boîtes d'entrée représentent les caractéristiques d'entrée pour la classification du diabète.

```
# Créer la fenêtre GUI
window = tk.Tk()

window.title("Système de classification du diabète - Phase de déploiement")

screen_width = window.winfo_screenwidth()
screen_height = window.winfo_screenheight()

window_width = 600
window_height = 800

x = (screen_width - window_width) // 2
y = (screen_height - window_height) // 2

window.geometry(f"{window_width}x{window_height}+{x}+{y}")

# Créer un cadre pour la phase d'entraînement
train_frame = tk.LabelFrame(window, text="Entraînement du système de classification", padx=10, pady=10)
train_frame.pack(pady=10)

# Créer un cadre pour les étiquettes d'entrée des caractéristiques et les boîtes d'entrée
input_frame = tk.LabelFrame(window, text="Caractéristiques d'entrée", padx=10, pady=10)
input_frame.pack(pady=10)
```

Figure 3.19 Création de la fenêtre GUI pour le module de déploiement

- Définition des fonctions de rappel (callback) pour les boutons "Entraîner le système" et "Classer".

- Ces fonctions extraient les caractéristiques et la variable cible à partir des données `diabetes_data`.

- Elles appellent ensuite les fonctions `train_model(X, y)` et `classify_model(X, y)` respectivement.

```
# Créer le bouton de classification
classify_button = tk.Button(input_frame, text="Classer", command=classify_button_click,
classify_button.pack()
```

Figure 3.20 Création le bouton de la classification pour le module de déploiement

```
button_font = font.Font(family='Arial', size=12, weight='bold')
# Créer le bouton de téléchargement
upload_button = Button(train_frame, text="Télécharger un fichier CSV", command=la
upload_button.pack(pady=10)

# Créer le bouton d'entraînement
train_button = tk.Button(train_frame, text="Entraîner le système", command=train_
train_button.pack()

# Créer les boîtes d'entrée des caractéristiques
label_pregnancies = tk.Label(input_frame, text="Nombre de grossesses: ex: 3")
label_pregnancies.pack()
entry_pregnancies = tk.Entry(input_frame)
entry_pregnancies.pack()

label_glucose = tk.Label(input_frame, text="Glucose: ex: 126")
label_glucose.pack()
entry_glucose = tk.Entry(input_frame)
entry_glucose.pack()

label_blood_pressure = tk.Label(input_frame, text="Pression sanguine: ex: 88")
label_blood_pressure.pack()
entry_blood_pressure = tk.Entry(input_frame)
entry_blood_pressure.pack()

label_skin_thickness = tk.Label(input_frame, text="Épaisseur de la peau: ex: 41")
label_skin_thickness.pack()
entry_skin_thickness = tk.Entry(input_frame)
entry_skin_thickness.pack()

label_insulin = tk.Label(input_frame, text="Insuline: ex: 235")
label_insulin.pack()
entry_insulin = tk.Entry(input_frame)
entry_insulin.pack()

label_bmi = tk.Label(input_frame, text="IMC: ex: 39.3")
label_bmi.pack()
entry_bmi = tk.Entry(input_frame)
entry_bmi.pack()
```

Figure 3.21 Création des boutons, des étiquettes d'entrée et des boîtes d'entrée pour le module de déploiement

- Création d'une zone de texte pour afficher les résultats

```
# Créer la zone de texte pour afficher l'importance des caractéristiques
result_text = tk.Text(window, height=12, width=50)
result_text.pack()
```

Figure 3.22 Création d'une étiquette pour afficher le résultat pour le module de déploiement

3.7 Présentation de l'interface graphique du système de diagnostic du diabète

Le système de diagnostic du diabète se compose de deux modules et une interface principale. Cela comprend une brève description du système et deux buttons pour accéder aux modules du système.

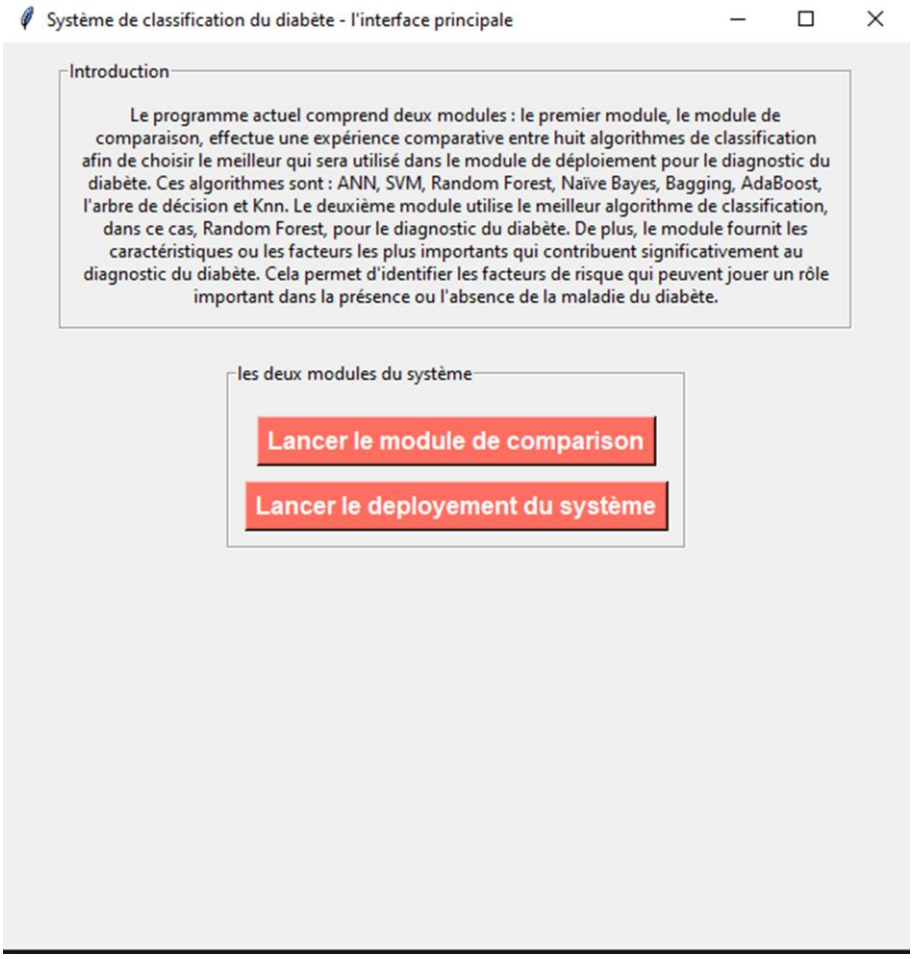

Figure 3.23 l'interface d'accueil pour le système de diagnostic du diabète

3.7.1 Le module de comparaison

Ce module vise à comparer la précision de classification de huit algorithmes. La première étape est importer le fichier « diabetes.csv » qui contient le jeu de données Pima diabètes (figure 3.25). Ensuite dans la deuxième étape, on lance le processus d'évaluation en cliquant sur le boutton « Calculer le taux de précision des modèles de diagnostic » (figure 3.26). Le résultat d'évaluation s'affiche dans la fenêtre de résultats.

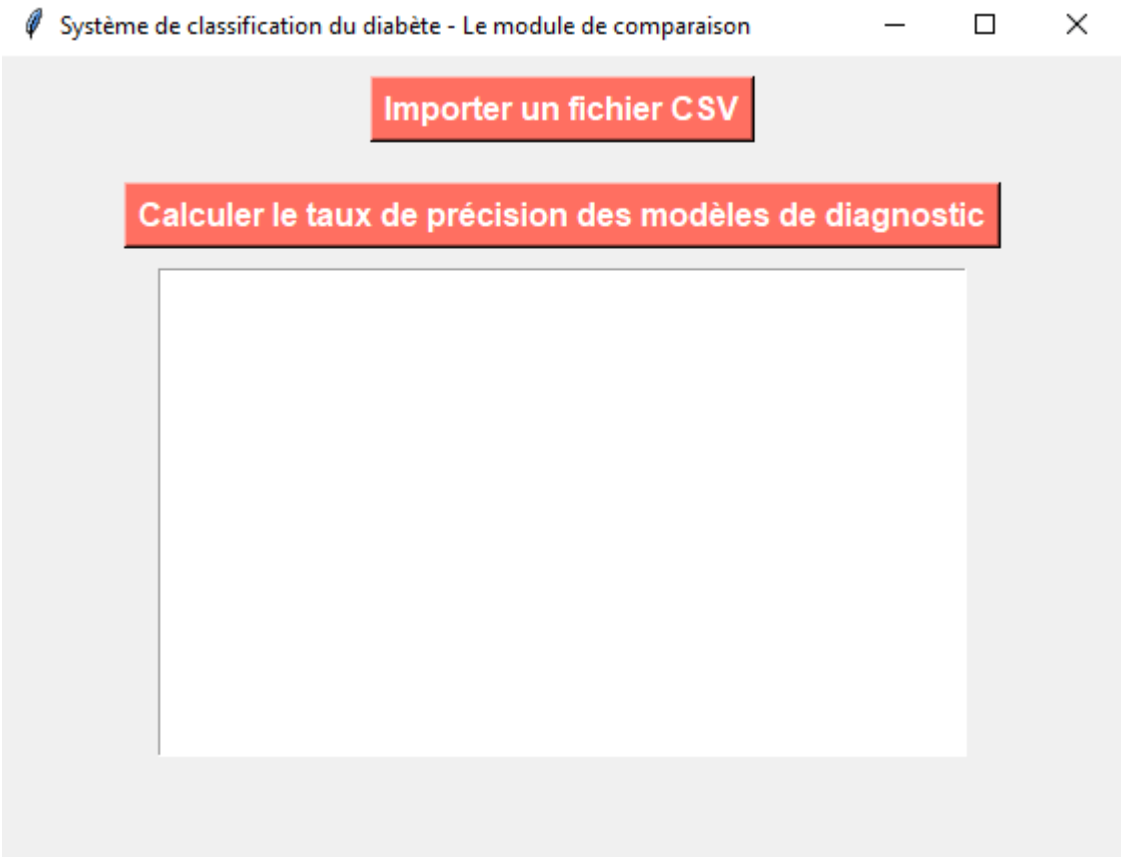

Figure 3.24 Interface du module de comparaison de diagnostic du diabète

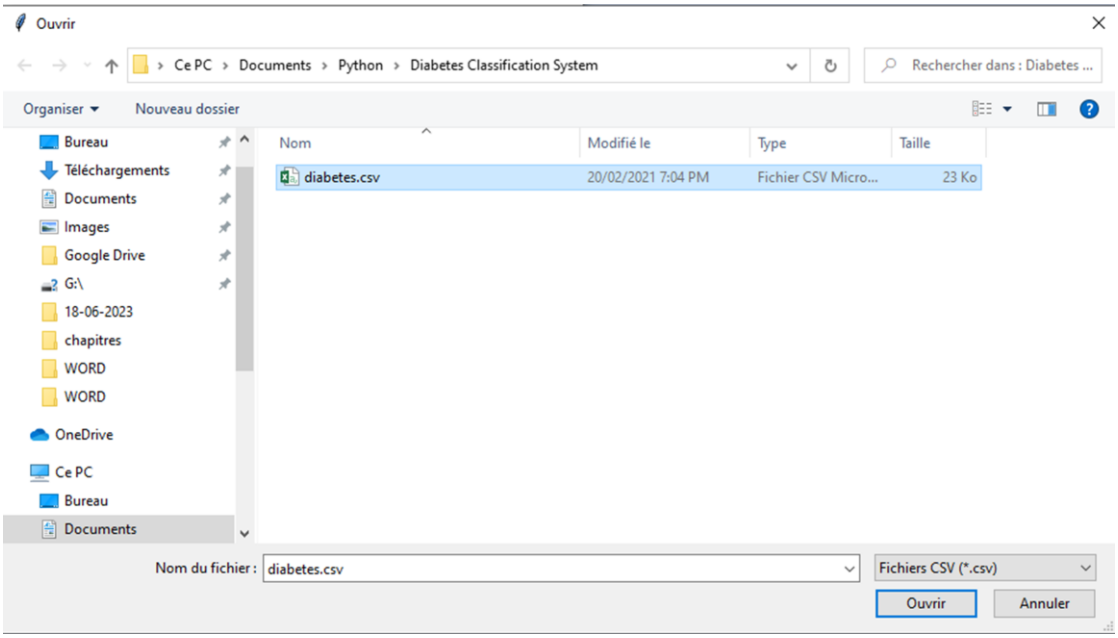

Figure 3.25 Importer le fichier « diabetes.csv » pour lancer la phase d'entrainement

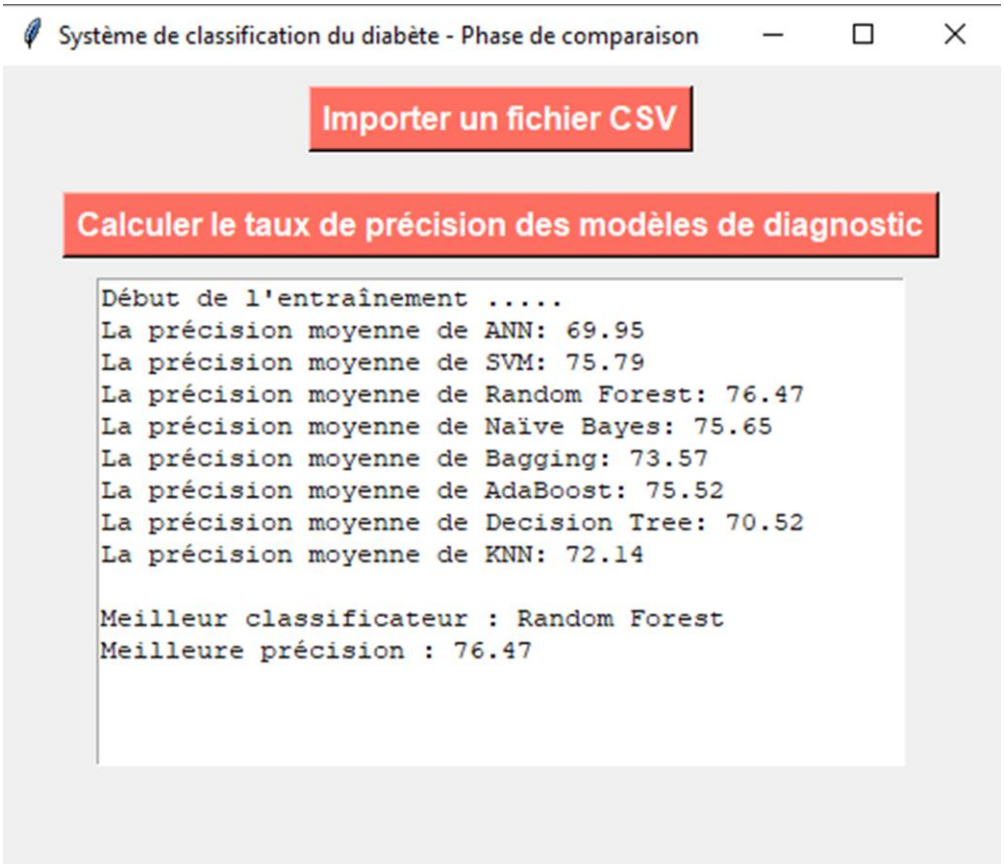

Figure 3.26 Les résultats de l'évaluation sont affichés dans la fenêtre des résultats.

3.7.2 Le module de déploiement

Apres la phase de comparaison, le meilleur algorithme est sélectionné (Random Forest dans notre cas). Cet algorithme est utilisé pour les processus de diagnostic. Pour utiliser ce module, on importe le fichier csv qui contient le jeu de données de diabète en cliquant sur le bouton « importer un fichier CSV » (figure 3.27) et ensuite, sur le bouton « Entrainer le système » pour construire le système de classification en utilisant l'algorithme « Random Forest» (figure 3.29). Après l'entrainement, le système est prêt pour diagnostiquer de nouveaux patients. L'utilisateur peut taper les valeurs des caractéristiques et ensuite clique sur le bouton « classer » pour diagnostique le cas en question (figure 3.30).

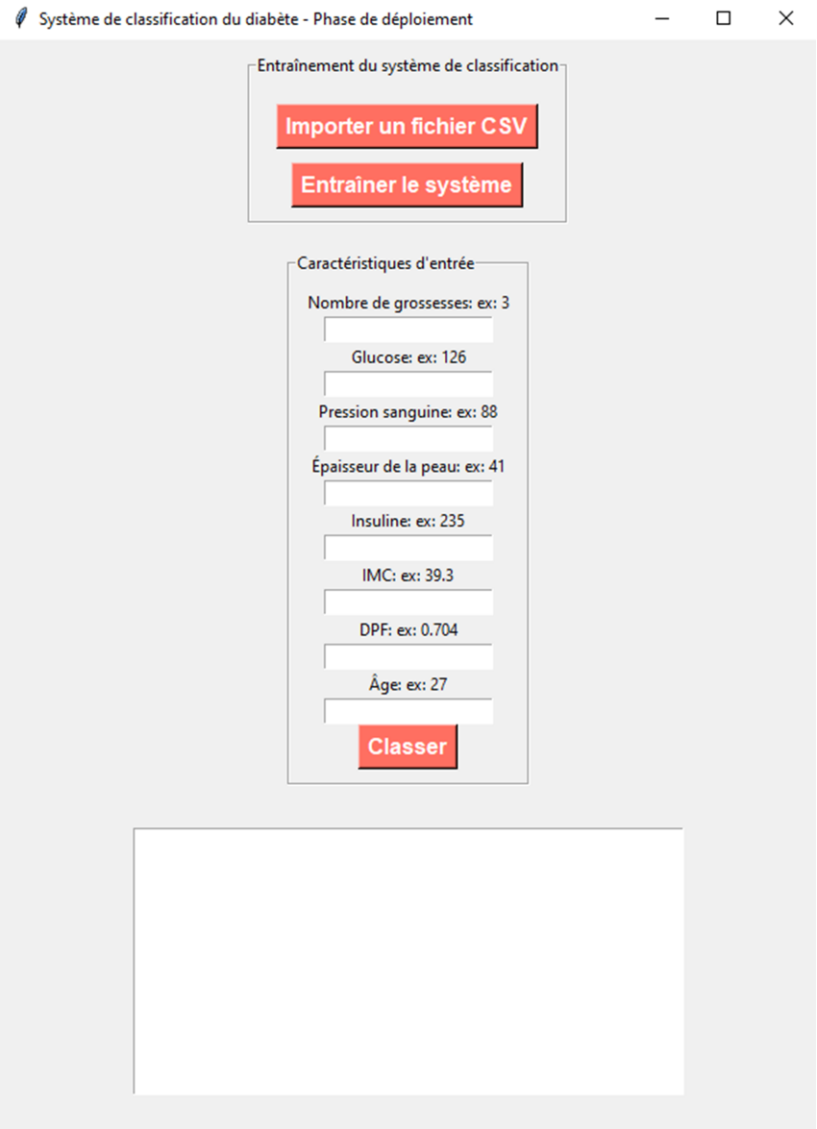

Figure 3.27 Interface du module de déploiement

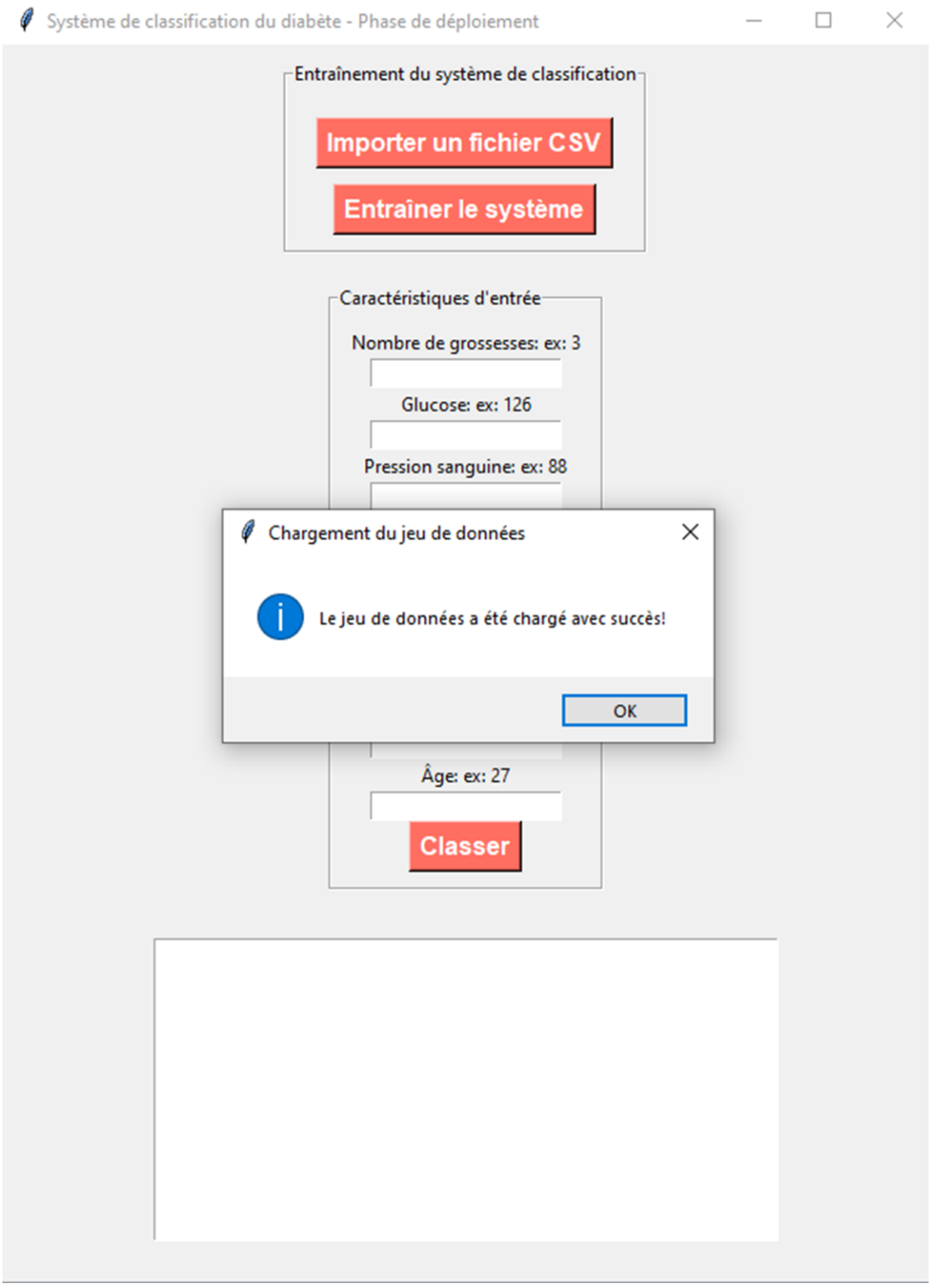

Figure 3.28 Importer le fichier « diabetes.csv »

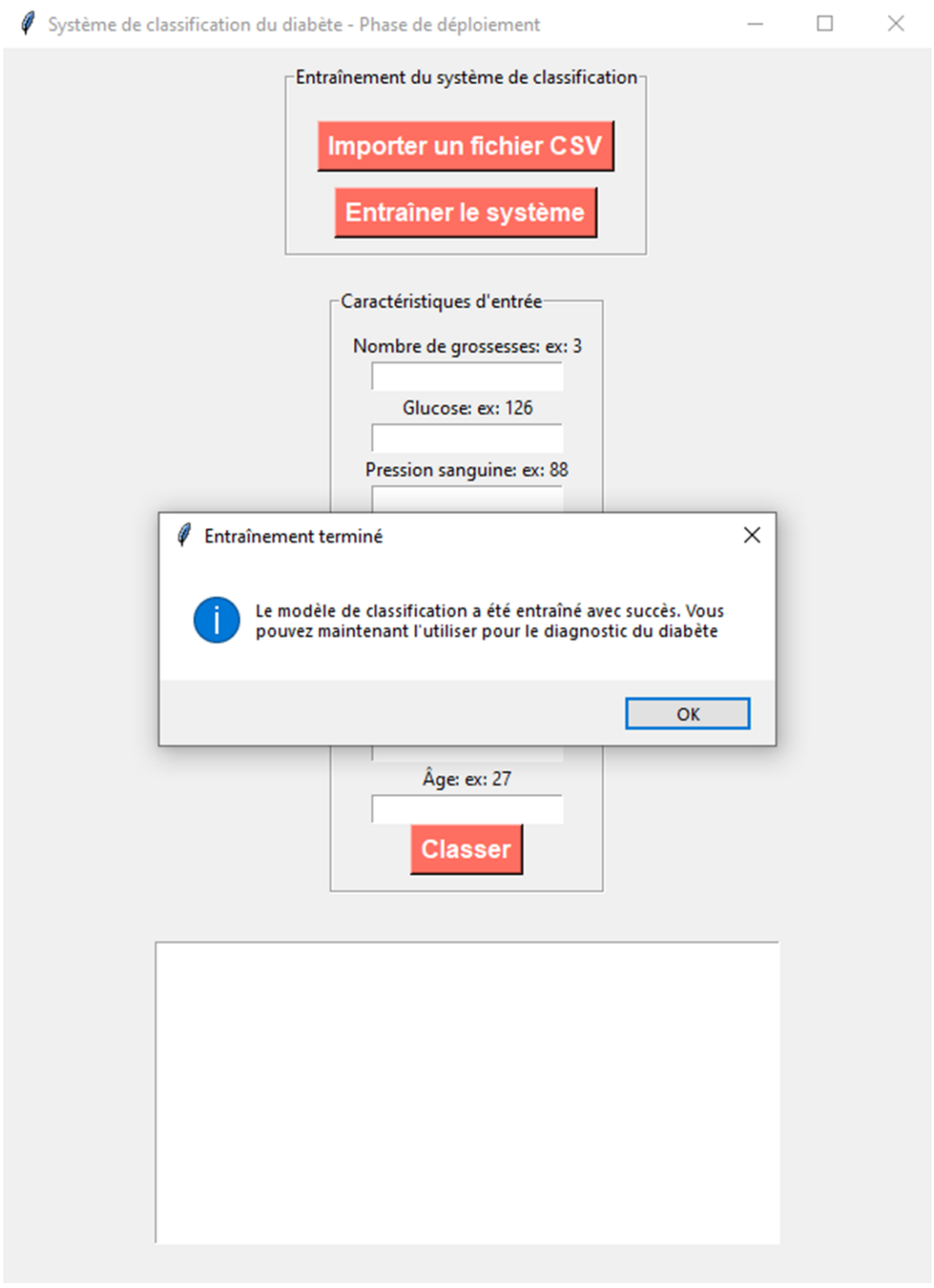

Figure 3.29 Entrainer le système en utilisant Random Forest

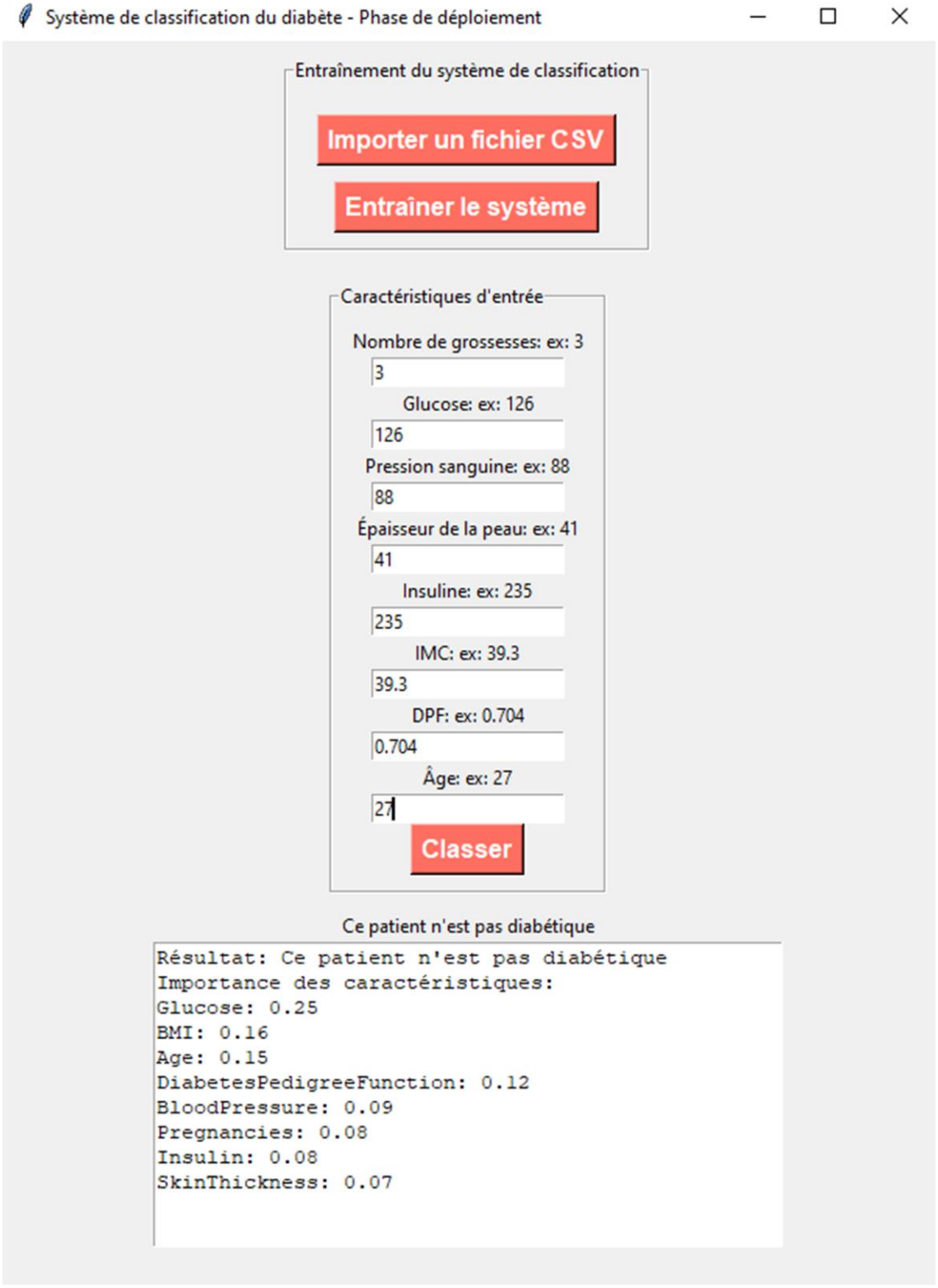

Figure 3.30 Utiliser le système pour diagnostiquer de nouveaux patients

Chapitre 4 : Résultats et Discussions

Les résultats obtenus dans cette étude sont divisés en deux parties.

4.1 La première partie : comparaison des performances de huit algorithmes

Cette partie représente et discute les résultats de l'étude comparative de 08 algorithmes de classification. La performance des algorithmes est évaluée en utilisant la méthode de validation croisée à 10 plis, qui implique de calculer la précision de classification 10 fois et de faire la moyenne des résultats sur les 10 plis. Le tableau 4.1 présente les résultats obtenus par chaque algorithme.

Table 4.1 la précision de classification moyenne de 08 algorithmes utilisant la méthode de validation croisée à 10 plis

	Fold1	**Fold2**	**Fold3**	**Fold4**	**Fold5**	**Fold6**	**Fold7**	**Fold8**	**Fold9**	**Fold 10**	**Average classification accuracy**
ANN	64.94	58.44	68.83	70.13	68.83	74.03	75.32	63.64	67.11	65.79	**67.71**
SVM	74.03	74.03	74.03	71.43	72.73	80.52	75.32	80.52	76.32	78.95	**75.79**
Random Forest	71.43	75.32	76.62	67.53	71.43	80.52	83.12	83.12	72.37	80.26	**76.17**
NB	72.73	75.32	79.22	71.43	71.43	79.22	76.62	80.52	72.37	77.63	**75.65**
Bagging	72.73	80.52	80.52	62.34	67.53	76.62	77.92	81.82	69.74	77.63	**74.74**
AdaBoost	72.73	75.32	75.32	64.94	77.92	70.13	83.12	81.82	71.05	82.89	**75.52**
Decision tree	66.23	74.03	68.83	55.84	70.13	68.83	83.12	80.52	63.16	77.63	**70.83**
Knn	67.53	79.22	71.43	67.53	66.23	74.03	70.13	79.22	71.05	75.00	**72.14**

En examinant le tableau qui représente la précision de classification de huit algorithmes utilisant la méthode de validation croisée à 10 plis, nous pouvons faire les observations suivantes :

1. Précision moyenne de classification : La précision moyenne de classification est indiquée pour chaque algorithme dans la dernière colonne du tableau. Elle nous donne une mesure globale de la performance de chaque algorithme sur l'ensemble des plis. Des valeurs plus élevées

indiquent une meilleure performance. Dans ce cas, les algorithmes varient d'une précision moyenne de 67,71 % (ANN) à 76,17 % (Random Forest).

2. ANN (Réseau de neurones artificiels) : L'algorithme ANN présente la plus faible précision moyenne de classification de 67,71 %. Il a des précisions relativement plus faibles dans certains plis par rapport aux autres algorithmes. Cela est probablement dû à la quantité insuffisante de données utilisées pour l'entraînement ou peut-être au choix des paramètres, étant donné que nous avons utilisé les paramètres par défaut fournis par la bibliothèque scikit-learn de Python.

3. SVM (Machine à vecteurs de support) : L'algorithme SVM affiche une précision moyenne de classification plus élevée de 75,79 %. Il maintient des précisions constantes dans la plupart des plis, ce qui indique une performance stable. Il performe mieux que ANN, mais est surpassé par Random Forest et NB.

4. Random Forest : L'algorithme Random Forest démontre une précision moyenne de classification relativement élevée de 76,17 %. Il atteint des précisions élevées dans plusieurs plis, ce qui témoigne de sa bonne performance et de sa robustesse. Il se classe premier en termes de précision moyenne.

5. NB (Naive Bayes) : L'algorithme NB affiche une précision moyenne de classification de 75,65 %. Il performe de manière cohérente dans les plis et est comparable à Random Forest en termes de précision moyenne.

6. Bagging : L'algorithme Bagging atteint une précision moyenne de classification de 74,74 %. Bien qu'il n'ait pas la plus haute précision parmi les algorithmes, il maintient une performance stable dans les plis.

7. AdaBoost : L'algorithme AdaBoost atteint une précision moyenne de classification de 75,52 %. Il présente des précisions variables dans différents plis, avec certaines valeurs élevées et d'autres faibles. Il n'est pas aussi cohérent que Random Forest ou NB.

8. Arbre de décision : L'algorithme de l'arbre de décision a une précision moyenne de classification de 70,83 %. Il performe relativement moins bien que les autres algorithmes en termes de précision moyenne.

9. KNN (K plus proches voisins) : L'algorithme KNN a une précision moyenne de classification de 72,14 %. Il présente des précisions variables dans différents plis, avec certaines valeurs

élevées et d'autres faibles. Il performe mieux que ANN, mais moins bien que Random Forest et NB.

Dans l'ensemble, Random Forest démontre la précision moyenne de classification la plus élevée, suivis de près par NB, SVM et AdaBoost. ANN et l'arbre de décision ont une performance relativement moins bonne par rapport aux autres algorithmes dans ce jeu de données particulier. Dans ce cas, Random Forest est sélectionné pour la phase de dépoilement.

4.2 La deuxième partie : déploiement du système

Pour la deuxième phase, nous utilisons la méthode de Random Forest pour entraîner le système de diagnostic du diabète sur le même ensemble de données. Une fois l'entraînement terminé, le système sera prêt à être utilisé pour diagnostiquer de nouveaux patients. Pour tester le fonctionnement du système, nous avons sélectionné de manière aléatoire un exemple du jeu de données (voir figure 3.30). Les valeurs des caractéristiques de cet exemple sont les suivantes :

- Nombre de grossesses : 03
- Glucose : 126
- Pression Sanguine : 88
- Epaisseur de la peau : 41
- Insuline : 235
- IMC : 39.3
- PDF : 0.704
- Age : 27

Le système de diagnostic a correctement donné le résultat suivant : ce patient n'est pas diabétique, suivi par un classement des caractéristiques selon leur importance ou leur contribution dans la classification du diagnostic. Voici les résultats obtenus (les caractéristiques sont énumérées avec leur importance respective):

Importance des caractéristiques :

Glucose: 0.25

BMI (Indice de masse corporelle): 0.16

Age: 0.15

DiabetesPedigreeFunction (Fonction de généalogie du diabète): 0.12

BloodPressure (Pression artérielle): 0.09

Pregnancies (Grossesses): 0.09

Insulin: 0.07

SkinThickness (Épaisseur de la peau): 0.07

Ces valeurs représentent l'importance relative de chaque caractéristique dans la classification. Plus la valeur est élevée, plus la caractéristique est considérée comme importante pour le diagnostic. Dans ce cas, la caractéristique la plus importante pour la classification est "Glucose" avec une valeur d'importance de 0.25, suivie de près par "BMI" (0.16) et "Age" (0.15). Cela suggère que la concentration de glucose dans le sang, l'indice de masse corporelle et l'âge sont des facteurs essentiels pour déterminer si un patient est diabétique ou non.

Il est important de noter que ces valeurs d'importance sont spécifiques à ce modèle de classification par Random Forest et peuvent varier d'un modèle à l'autre.

Chapitre 5 : Conclusion

Le diabète est une maladie chronique qui représente un défi majeur pour la santé mondiale. Avec une prévalence croissante, il est essentiel de développer des outils de diagnostic précis pour détecter la maladie et fournir un traitement approprié. Dans cette étude, nous avons exploité les avancées de l'apprentissage automatique pour développer un système de diagnostic du diabète, en utilisant l'algorithme de classification le plus performant.

Notre étude a été guidée par plusieurs objectifs. Tout d'abord, nous avons évalué huit algorithmes de classification couramment utilisés pour sélectionner celui qui offre la plus grande précision de diagnostic du diabète. En utilisant le jeu de données Pima diabetes comme étude de cas, nous avons mesuré la précision de classification de chaque algorithme en utilisant la méthode de validation croisée à 10 plis. Les résultats ont révélé que l'algorithme Random Forest présente la précision moyenne la plus élevée, suivi de près par Naive Bayes, SVM et AdaBoost.

Ensuite, nous avons utilisé l'algorithme Random Forest pour développer le système de diagnostic du diabète. Ce système a été entraîné sur le même jeu de données. Les caractéristiques les plus importantes pour la classification ont également été identifiées, mettant en évidence l'importance du glucose, de l'indice de masse corporelle (BMI) et de l'âge dans la détermination du statut diabétique. Ces informations peuvent être précieuses pour atténuer le risque de diabète chez les personnes non atteintes.

En conclusion, notre étude démontre l'efficacité de quelques algorithmes d'apprentissage automatique pour le diagnostic du diabète, notamment Random Forest, Naive Bayes, SVM et AdaBoost.. Le système de diagnostic développé peut contribuer à la détection du diabète.

Cependant, il convient de noter certaines limites de notre étude. Premièrement, l'efficacité des algorithmes de classification peut varier en fonction du jeu de données utilisé. Par conséquent, il est important de valider les résultats sur des ensembles de données supplémentaires.

Pour les recherches futures, il serait intéressant d'explorer l'application de l'apprentissage automatique à des ensembles de données plus vastes et diversifiés, en incluant des facteurs de risque supplémentaires et des caractéristiques génétiques.

Bibliographie

Acharya, U. (. (2019).). Using deep learning to develop awearable device for diabetes management. *Diabetes Technology & Therapeutics, 21(10),*.

Acharya, U. R. ((2017)). . A novel proposal for deep learning-based diabetes prediction: converting clinical data to image data. . *Diabetes Research and ClinicalPractice, 130,* .

Aggarwal, C. (2017). *Machine Learning for Healthcare Applications.*

Agrawal, R. M. (2013). *Naive Bayes for the diagnosis of diabetes mellitus. International Journal of Computer Applications,.*

Andreas Müller, S. (2007). *Support Vector Machines: The Foundations and Applications.*

Andrew Ng, p. (2004). *An Introduction to Support Vector Machines.*

Awasthi, A. e. (2010). "Support vector machines for classification of diabetes mellitus.". *Artificial Intelligence in Medicine 47.3* .

Balasubramanian, V. &. (2005). Predicting diabetes using artificial neural networks. . *Expert Systems with Applications, 28(3),* .

Bernhard Scholkopf, A. j. (2002). *Learning with Kernels:Support Vector Machines,Regulariyation,Optimiyation and Beyond.*

Bernhard Schölkopf, A. J. (2002). *Support Vector Machines: Theory and Applications.*

Bishop, C. ((1995).). *Réseaux de neurones pour la reconnaissance de formes.* . Presse universitaire d'Oxford.

Bishop, C. (2006). *Pattern Recognition and Machine Learning. Springer.*

Bishop., C. (2006). *Support Vector Machines.*

Breiman, l. (1996.). *bagging predictors,machine learning.*

Breiman, L. (1998). "Overfitting explained.". *The Annals of Statistics 26.5* .

Breiman, L. (2001). *random forests* .

Breiman, L. F. (2001). *Classification and regression trees. Boca Raton, FL: Chapman & Hall/CRC.*

Burkov, A. (2019). *Machine learning for fraud detection* .

Chan, D. (2016). *Rapport Mondial sur le diabète.* Organisation mondiale de la Sante.

Chawla, N. V. (2014). *A k-nearest neighbors classifier for the diagnosis of diabetes.*

Chen W, e. (2020). *for support vector machines in diabetes mellitus.*

Chen, a. (2011). *Bagging for the prediction of diabetic nephropathy.*

chen, a. (2018). *Bagging for the identification of patients with prediabetes.*

Chen, J. e. (2018). "A deep learning approach for the prediction of type 1 diabetes mellitus.". *Diabetes Research and Clinical Practice 143.*

Chen, J. e. (2019). "Deep learning for the development of educational materials for type 1 diabetes mellitus." . *Diabetes Technology & Therapeutics 21.1* .

Chen, L. L. (2012). *Adaptive boosting for diabetes diagnosis.*

Chen, W. a. (2017). Ensemble learning for support vector machines in diabetes mellitus. *Journal of Diabetes Science and Technology 11.1.*

Chen, W. e. (2016). "Outlier detection in diabetes mellitus using support vector machines.". *Journal of Diabetes Science and Technology 10.5.*

Christopher J, C. (2005). *Support Vector Machines for Data Mining.*

Christopher, m. (2006). *Support Vector Machines for Pattern Recognition.*

Christopher, M. (2006). *support vector machines:theory and application* .

construction., I. d. ((2017)). . *Récréation technique : un guide pour les propriétaires et les entrepreneurs.* .

Cormen, T. H. (2009). *Introduction to Algorithms, 3rd Edition. MIT Press,.*

Das, S. (2015). *Naive Bayes classification for diabetes mellitus. International Journal of Computer Applications.*

Das, S. P. (2016). *Naive Bayes for the diagnosis of diabetes mellitus using clinical and laboratory data. International Journal of Computer Applications,.*

Dash, S. S. (2014). . *Naive Bayes for the early detection of diabetes mellitus. International Journal of Computer Applications,.*

Ding, Y. W. ((2017).). Classification of diabetic retinopathy using deep learning. . *IEEE Access, 5,.*

Domingos, p. (2000). *A few useful things to know about machine learning.*

El-Sakka., A. (2019). *Machine Learning for Healthcare Analytics.*

El-Sherif, A. E.-S. (2022). *Naive Bayes classification for the prediction of type 2 diabetes in adults. Journal of Diabetes Research, 2022, 1070782. doi:10.1155/2022/1070782.*

Freund, Y. (1997.). *A decision-theoretic generalization of on-line learning and an application to boosting.*

Freund, Y., & Schapire, R. (1995). *A decision-theoretic generalization of on-line learning and an appliocation to boosting.*

Gareth, J. (2013). *Introduction to Statistical Learning.*

Goodfellow, I. B. (2016). *Neural Networks.*

Goodrich, M. T. (2014). *Data Structures and Algorithms in Java, 4th Edition. Pearson Education.*

Gupta, A. K. ((2018)). . *Big data analytics for smart healthcare applications: A review. IEEE Access, 6.*

Hastie, T. (2009). *Les éléments de l'apprentissage statistique. Springer.*

Haykin, S. (1999). *Réseaux de neurones : une base complète. Macmillan.*

Heidelberg, E. b. (2014). "Decision Tree Algorithm for Diabetes Diagnosis.". *Intelligent Data Analysis with Applications in Healthcare.*

Hongwzi Mao, M. A. (2016). *Resource Managment with Deep Reinforcement Learning.*

James, G. (2012). *Introduction to decision tree* .

James, G. a. (2013). *An introduction to statistical learning* .

Jamkhandikar, S. a. (2017). *"Thyroid disease prediction using decision tree algorithm."*.

jesus cardenas. (2017, 04 12).

Jiang, H. e. (2020). "A review of machine learning methods for diabetes diagnosis.". *Journal of Diabetes Science and Technology 14.1*.

Jordan, R. (2004). *Machine Learning for Medical Diagnosis by Michael I.*

Kaur, J. a. (2017). "Decision tree based diabetes prediction using machine learning algorithms.". *International Journal of Engineering Research & Technology 6.6.*

Kevin, p. (2012). *Machine learning:A probabilistic perspective* .

Kevin, p. (2012.). *Aprobabistic perspective.*

Khan, M. K. (2018). *). Naïve Bayes for the diagnosis of diabetes mellitus. International Journal of Research in Engineering & Technology.*

Khan, S. U. (2017). *A k-nearest neighbors approach for diabetes diagnosis using clinical and laboratory data.* .

Khodabakhsh, M. &. ((2018).). Personalizing insulin therapy using deep learning. *Diabetes Technology & Therapeutics, 20(10),*.

Kirik, D. &. ((2019)). Using deep learning to develop a virtual pancreas for type 1 diabetes. . *Diabetes Technology & Therapeutics, 21(7),* .

Kumar, V. S. (2021). . *A naïve Bayes classifier for the early detection of diabetes. Diabetes & Metabolic Syndrome: Clinical Research & Reviews.*

Kupper, L. &. (1998). *Stratégies pour maximiser le rendement des tests diagnostiques.*

Lee Giles, c. (2018). *Machine Learning for Healthcare: From Research to Practice.*

li, a. (2012). *"Bagging for the diagnosis of diabetic ketoacidosis".*

Li, a. (2016). *8.(ReBagging for the identification of patients with diabetic nephropathy at an early stage.*

Li, J. (2017). *Application of random forest in the classification of diabetic retinopathy.*

Li, J. W. (2020). . *An ensemble learning method for diabetes diagnosis based on AdaBoost and deep learning.*

Li, X. L. (2014). *A novel ensemble learning method for diabetes diagnosis based on AdaBoost.*

Li, Y. (2021). *A hybrid k-nearest neighbors and support vector machine approach for diabetes diagnosis.*

Li, Z. W. (2017). . *An ensemble learning method for diabetes diagnosis based on adaptive boosting and support vector machine. .*

Ltd., T. a. (2016). "Decision Tree Algorithm for Predicting Diabetes." . *Advances in Parallel Computing Algorithms,.*

Michael, N. (2015). *Deep Learning for Healthcare.*

Michael, p. (2017). *Essentials of diabete mellitus :Third Edition.*

Mishra, S. T. (2017). *Application of naïve Bayes for diabetes mellitus diagnosis.*

modèle de validation croisée. (s.d.).

Mohamed, T. H.-D.-R. (2016). *Classification of type 2 diabetes using k-nearest neighbors. .*

Müller., A. (2016). *Machine Learning for Healthcare.*

Murphy, K. P. (2012). *Machine learning: A Probabilistic Perspective. MIT Press.*

Nielsen., M. (2015). *the neural network handbook.*

Palaniappan, S. e. (2012). "Decision tree-based intelligent system for diabetes diagnosis.". *l. Expert Systems with Applications 39.1.*

Peter, j. (1987). *decision tree for elementary arithmetic .*

Qi, Y. (2012). Random forest for bioinformatics. In *Ensemble machine learning: Methods and applications* (pp. 307-323). Boston, MA: Springer US.

Qu, K. L. ((2021)). . *AI-based smart prediction of clinical disease using random forest. Artificial Intelligence in Medicine, 123,.*

Quinlan, J. R. (1986). "Decision trees." . *Machine learning 1 (1).*

Rathi, N. R. (2019). *Naive Bayes for diabetes diagnosis. International Journal of Computer Applications.*

Rish, I. (2001). *An empirical study of the naive bayes classifier.*

Sarwar N, G. P. (2010). Diabetes mellitus, fasting blood glucose concentration, and risk of vascular disease: a collaborative meta-analysis of 102 prospective studies. *Emerging Risk Factors Collaboration.*

Singh, A. A. (2012). . *Naive Bayes for the diagnosis of diabetesmellitus using clinical data. International Journal of Computer Applications.*

Sun, J. e. (2019). "Personalized insulin treatment for type 1 diabetes mellitus using a deep learning-based closed-loop insulin delivery system.". *Diabetes Care 42.12* .

Suthaharan, S. (s.d.). *I machine learning models and algorithme for big data classification thinking with exempel for effective learning 123.*

Tamer, S. P. (2020). *data Analtic in Asset Management* .

Tavasoli, S. (2023). *Top 10machine learning algorithme for Beginners:Supervised,and More* .

Thirugnanam, M. ((2007).). An improved artificial neural network model for effective diabetes prediction. . *Journal of Medical Systems, 31(3),.*

Tom, M. (1997.). *introduction to machine learning* .

Trevor, H. (2009). *the elements of statistical learning* .

Wang, a. (2010). *Bagging for the diagnosis of diabetic retinopathy.*

wang, a. (2014). *Bagging for the identification of patients with impaired fasting glucose.*

Wang, J. L. (2020). *A deep learning k-nearest neighbors approach for diabetes diagnosis.*

Wang, L. Z. (2015). *An improved AdaBoost model for diabetes diagnosis.* .

Wang, W. Z. (2019). *A k-nearest neighbors approach for diabetes diagnosis using machine learning.* .

Wang, X. e. ((2018)). "Retinal image analysis for diabetic retinopathy diagnosis using deep learning." IEEE Transactions on Medical Imaging 37.10 2440-2448. *IEEE Transactions on Medical Imaging 37.10 2440-2448.*

Wang, X. e. (2019). "Deep learning for the informing of policy decisions about type 1 diabetes mellitus." . *Diabetes Technology & Therapeutics 21.1* .

Wang, X. e. (2019). "Deep learning for the discovery of new drug targets for type 2 diabetes mellitus.". *Nature Medicine 25.6* .

Wang, Y. e. (2015). "Support vector clustering for diabetes mellitus." . *BMC Medical Informatics and Decision Making 15.1.*

Wang, Y. W. (2018). *An improved ensemble learning method for diabetes diagnosis based on AdaBoost and decision tree.* .

Wang, Y. W. (2021). *An improved ensemble learning method for diabetes diagnosis based on AdaBoost and deep belief network.* .

Wang, Z. Z. (2015). *A k-nearest neighbors approach for diabetes risk prediction.* .

wang. (2017). *Bagging for the prediction of diabetic retinopathy at an early stage.*

WHO (2023) https://www.who.int/news-room/fact-sheets/detail/diabetes accessed on 26/06/2023

Xindong Wu, V. K. (2008). *top10 algorithms in data mining.*

Zhang L, e. (2014). *Support vector regression for blood glucose level prediction in type 1 diabetes mellitus.*

Zhang L, e. (2018). *"Transfer learning for support vector machines in diabetes mellitus.".*

Zhang, a. (2008). *"Bagging for the prediction of type 2 diabetes".*

Zhang, a. (2013). *Bagging for the prediction of hypoglycemia in patients with type 1 diabetes.*

Zhang, B. (2015). . *Predicting gestational diabetes using random forest. Journal of Diabetes Research, 2015, 820443. doi:10.1155/2015/820443.*

Zhang, C. L. (2013). *Adaboost-based ensemble learning for diabetes diagnosis.*

Zhang, F. Z. (2018). *Improving the performance of k-nearest neighbors for diabetes diagnosis using feature selection.* .

Zhang, H. W. (2016). *An improved AdaBoost model for diabetes diagnosis.*

Zhang, J. W. ((2018)). . Developing a decision support system for diabetes management using deep learning. *Diabetes Technology & Therapeutics, 20(11),* .

Zhang, L. (2016). . *Predicting type 1 diabetes using random forest. Computer Methods and Programs in Biomedicine, 128(1).*

Zhang, L. (2017). support vector machines in diabetes mellitus. *Diabetes Technology & Therapeutics 19.8.*

Zhang, L. a. (2016). *Journal of Diabetes Science and Technology 10.2* .

Zhang, L. a. (2020). *support vector machines in diabetes mellitus Journal of Diabetes Science and Technology 24.3.*

Zhang, L. e. (2018). "Transfer learning for support vector machines in diabetes mellitus.". *Diabetes Technology & Therapeutics 20.10.*

Zhang, L. e. (2019). "Deep learning for the prediction of type 2 diabetes mellitus." . *Diabetes Technology & Therapeutics 21.1* .

Zhang, L. e. (2019). "Deep learning for the raising of public awareness about type 2 diabetes mellitus". *Diabetes Technology & Therapeutics 21.1.*

Zhang, L. e. (2019). "Deep learning for the prediction of diabetic retinopathy progression.". *Diabetes Technology & Therapeutics 21.1.*

Zhang, L. Z. (2012). Random forest for diabetes diagnosis: a systematic review and meta-analysis.

Zhang, L. Z. (2013). *A comparative study of machine learning algorithms for diabetes diagnosis. Journal of Diabetes Research, 2013, 207115. doi:10.1155/2013/207115.*

Zhang, L. Z. (2014). . *Improving the accuracy of diabetes diagnosis using random forest. Journal of Diabetes Research, 2014, 809871. doi:10.1155/2014/809871.*

Zhang, M. Y. (2019). . *An improved AdaBoost model for diabetes diagnosis based on local outlier factor.*

Zhang, X. (2017). *Machine learning methods with noisy, incomplete or small datasets: A review. Artificial Intelligence Review, 47(1).*

Zhang, X. D. ((2017)). Predicting glycemic control in type 1 diabetes using deep learning. . *Diabetes Technology & Therapeutics, 19(11),.*

Zhang, Y. (2021). Diabetes Technology & Therapeutics. *"Deep learning for diabetes diagnosis: A systematic review.*

Zhang, Y. W. (2022). *A novel k-nearest neighbors approach for diabetes diagnosis using machine learning. .*

Zhang, Z. W. (2010). *Application of k-nearest neighbors in the diagnosis of diabetes mellitus. .*

Zou, Q. Q. (2022). *Predicting diabetes mellitus with machine learning: A review. Computers, Materials & Continua.*