Pipeline d'exploration de données de classification - Devoir 2

-Reet Khanchandani

1

Lien du cahier Google Colab

https://colab.research.google.com/drive/1YmWr6KICmwUz7PRQeDTGxBtT0NiLH14v?usp=sharing

Translated to French:

Introduction om/drive/1YmWr6KICmwUz7PRQeDTGxBtT0NiLH14v?usp=sharing

L'objectif de cette tâche de fouille de données est d'effectuer un apprentissage automatique supervisé - en particulier de la

Appliquez des algorithmes de classification traditionnels tels que Random Forest et HistGradientBoosting en utilisant s

Accélérer la formation du modèle et l'inférence en utilisant la trousse d'outils Intel oneAPI AI Analytics et la comparer a

Utilisez NVIDIA RAPIDS (cuML) pour évaluer les performances et l'efficacité accélérées par GPU.

Intégrez un réseau neuronal en utilisant PyTorch et comparez ses performances de classification et sa vitesse d'entraî

Aperçu et sélection de l'ensemble de données

Pour cette tâche, l'ensemble de données sélectionné est l'ensemble de données des indicateurs de santé du diabète du C

Principales caractéristiques

IMC - Indice de masse corporelle

Tabagisme, consommation d'alcool et activité physique

https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators

https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators

Santé générale, santé mentale et physique

· Âge, Niveau d'éducation, Tranche de revenu

· Accès aux soins de santé et barrières liées aux coûts

Antécédents d'accident vasculaire cérébral, de maladie cardiaque ou d'hypertension artérielle.

Variable cible

· Diabète binaire (0 = Pas de diabète, 1 = Prédiabète ou diabète)

Raison de la sélection

Le jeu de données sur le diabète du CDC a été sélectionné en raison de ses :

Pertinence aux défis de santé du monde réel

Mélange équilibré de caractéristiques catégorielles et continues

Taille suffisante pour une évaluation significative du modèle et un étalonnage des performances

Compatibilité avec les bibliothèques d'apprentissage automatique accélérées par CPU et GPU

Tous les enregistrements ont été nettoyés et prétraités dans le cadre du pipeline pour garantir la cohérence et la préparatio

Méthodologie

La méthodologie suivie dans ce projet se compose des étapes clés suivantes, couvrant le flux de travail complet de l'appre

Exploration et analyse des données

· A exploré la distribution de la variable cible Diabète binaire pour évaluer le déséquilibre des classes.

· Visualisation des distributions des caractéristiques à l'aide d'histogrammes pour comprendre la dispersion, l'asymétrie

· Analyse des corrélations par paires entre les caractéristiques numériques en utilisant une matrice de corrélation heatr

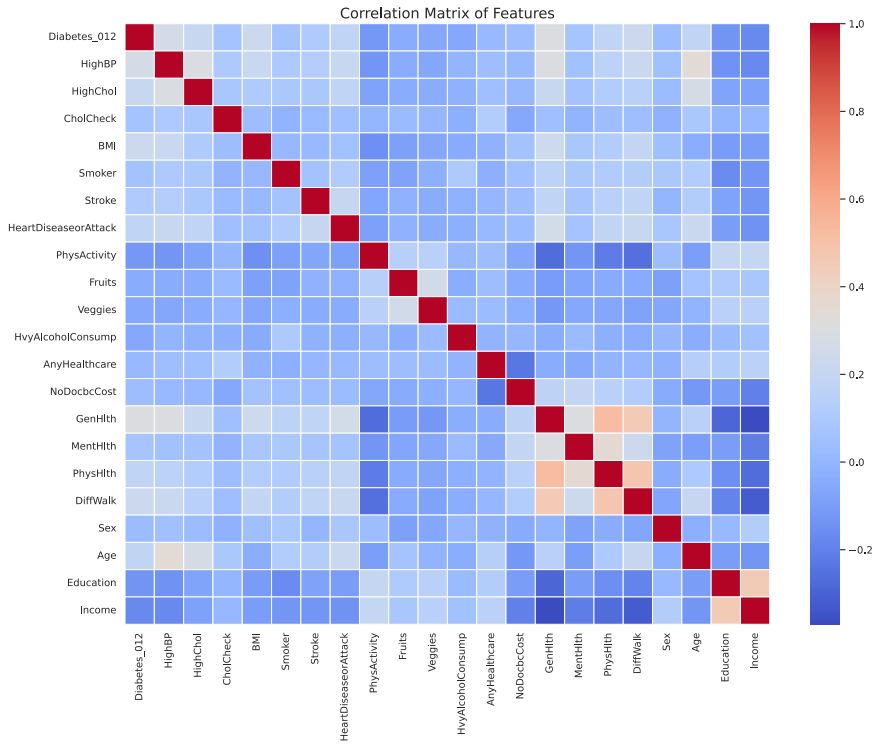

Figure 1: Matrice de confusion des caractéristiques

2. Préparation des données

· Séparé l'ensemble de données en caractéristiques (X) et la variable cible (y).

Divisez les données en ensembles d'entraînement de 80 % et de test de 20 % en utilisant un échantillonnage stratifié p

· Appliqué une mise à l'échelle standard pour la compatibilité du modèle PyTorch et assuré un prétraitement cohérent d

3. Mise en ·uvre et formation du modèle

Les modèles suivants ont été mis en ·uvre pour évaluer les performances de classification dans différents environnements

Modèles standard de Scikit-learn:

Classificateur de forêt aléatoire

Classificateur de renforcement de gradient basé sur l'histogramme

Modèles Scikit-learn optimisés pour Intel:

Forêt aléatoire et augmentation de l'histogramme en utilisant l'extension Intel pour Scikit-learn

NVIDIA RAPIDS:

- Classificateur de forêt aléatoire cuML utilisant l'accélération GPU

Réseau de neurones :

Réseau de neurones à propagation avant à couches multiples implémenté en utilisant PyTorch

Évaluation des performances

Tous les modèles ont été évalués en utilisant des métriques cohérentes et des références computationnelles.

Précision calculée, précision, rappel, score F1 et AUC ROC.

· Temps de formation mesuré et temps d'inférence en secondes.

Facteurs d'accélération calculés entre les implémentations standard et accélérées (Intel vs. Scikit-learn, GPU vs. CPU).

Détails de mise en ·uvre

Implémentation standard de Scikit-learn

Le modèle de base a été implémenté en utilisant le RandomForestClassifier standard de scikit-learn, une méthode d'ensem

Configuration du modèle

La classe weight='balanced' a été utilisée pour gérer un léger déséquilibre de classes.

La graine aléatoire a été fixée à 42 pour garantir la reproductibilité.

L'évaluation comprenait à la fois des étiquettes de classe prédites et des probabilités prédites pour soutenir le calcul de

Métriques d'évaluation

Le modèle a été entraîné et testé en utilisant une division stratifiée de 80/20. Les métriques de performance sont résumées

Forêt aléatoire (Standard)
Temps d'entraînement (s)
31,488
Temps d'inférence (s)
1,575
Précision
0,8383
Précision
0,7891
Rappel
0,8003
Score F1
0,7357
AUC ROC

Tableau 1 : Performance du classificateur de forêt aléatoire standard

Matrice de confusion

La matrice de confusion ci-dessous visualise la performance du modèle sur l'ensemble de test. La plupart des prédictions s

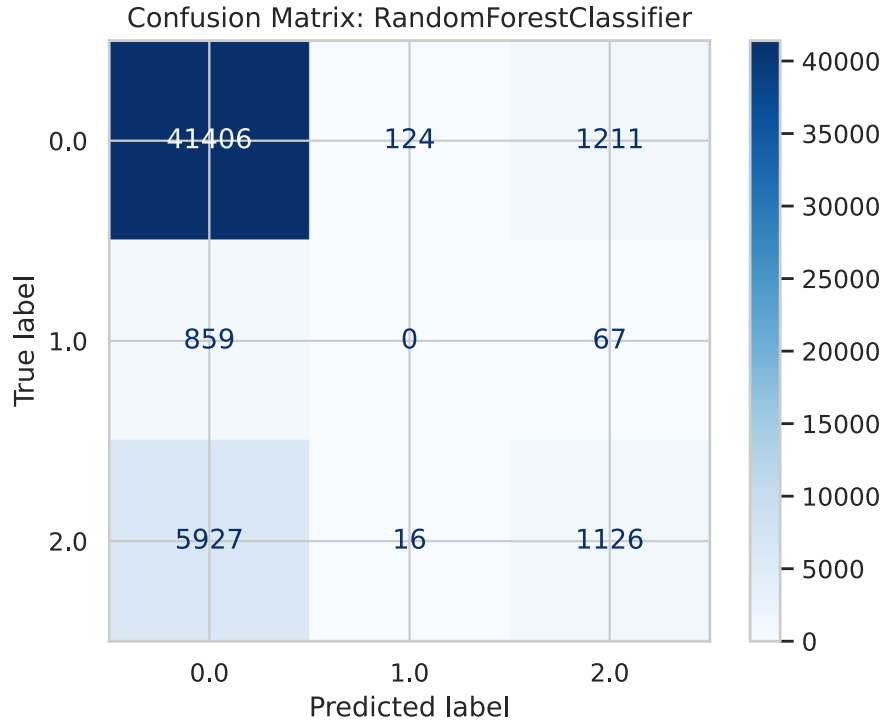

Figure 2 : Matrice de confusion - RandomForestClassifier (scikit-learn standard)

Implémentation de Scikit-learn optimisée pour Intel

Pour accélérer les performances du modèle sur le CPU, l'extension d'Intel pour Scikit-learn a été utilisée. Cette extension p

Mise en ·uvre

Le mécanisme de patching a été appliqué en utilisant la bibliothèque sklearnex.

L'interface du modèle est restée inchangée, garantissant la compatibilité avec les pipelines Scikit-learn existants.

Après le patch, le même RandomForestClassifier a été utilisé sur le même ensemble de données et le même pipeline

Métriques d'évaluation

Le tableau suivant présente les résultats d'évaluation pour le classificateur Random Forest optimisé par Intel :

Forêt aléatoire (optimisée par Intel)
Temps d'entraînement (s)
23,361
Temps d'inférence (s)
1,593
Précision
0,8383
Précision
0,7891
Rappel
0,8003
Score F1
AUC ROC

Tableau 2 : Performance du classificateur de forêt aléatoire optimisé par Intel

Observations

Alors que la précision du modèle et les métriques de classification restent identiques à l'implémentation standard - comme

Google Colab ne fonctionne pas sur les processeurs Intel avec AVX-512 ou backends activés par oneAPI, limitant l'acc

Le gain de performance des optimisations d'Intel est plus apparent sur les systèmes avec des processeurs Intel Xeon d

Malgré la contrainte matérielle, cet exercice a démontré à quel point l'extension d'Intel peut s'intégrer de manière transpare

Mise en ·uvre de NVIDIA RAPIDS

Pour explorer l'accélération GPU pour l'apprentissage automatique, le framework RAPIDS de NVIDIA a été utilisé. RAPIDS

Configuration et dépendances

L'environnement RAPIDS a été mis en place sur Google Colab en utilisant les utilitaires officiels RAPIDS CSP (Cloud Serv

cuDF a été utilisé pour lire et prétraiter l'ensemble de données sur le GPU.

Le RandomForestClassifier de cuml.ensemble a été utilisé pour la classification.

La division train-test a été effectuée via cuml.model_selection.train_test_split.

Configuration du modèle

La forêt aléatoire cuML a été configurée avec :

· Nombre d'estimateurs = 100

· profondeur maximale = 16

· état aléatoire = 42

Métriques d'évaluation

Après la prédiction, les sorties GPU (cuDF.Series) ont été converties en tableaux NumPy basés sur le CPU pour calculer le

Métrique	cuML Random Forest (GPU)
Temps d'entraînement (s)	1.342
Temps d'inférence (s)	0.995
Précision	0.8479
Précision	0.8010
. .	0.8479
Score F1	0.8035
AUC ROC	

Tableau 3 : Performance du classificateur de forêt aléatoire cuML

Matrice de confusion

La matrice de confusion ci-dessous montre les prédictions du classificateur GPU. Tout comme les modèles précédents, la

La méthode predict_proba() n'est pas disponible dans la version actuelle de cuML, donc l'AUC ROC n'a pas pu être calcule

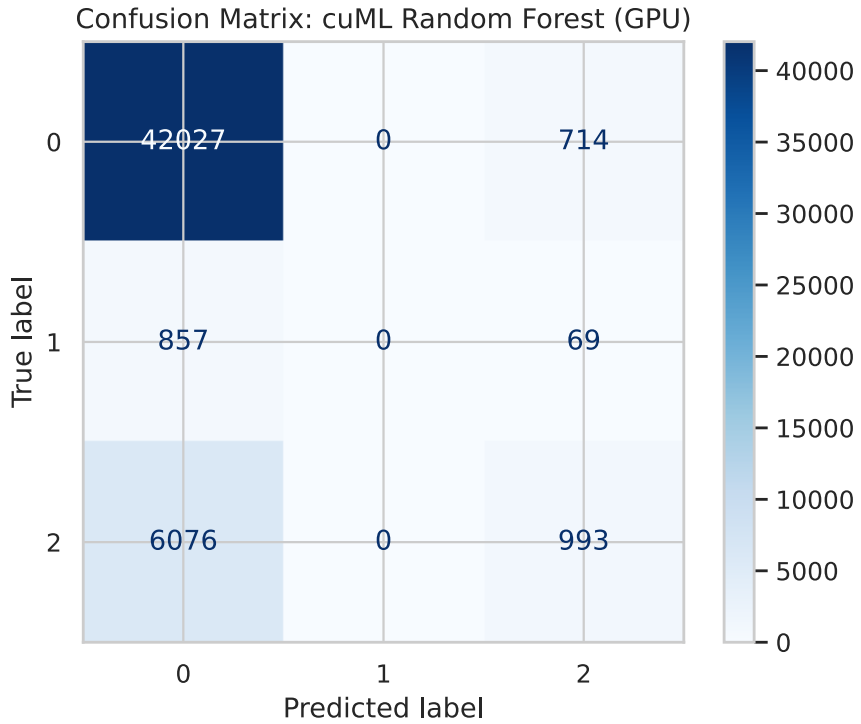

Figure 3 : Matrice de confusion - RandomForestClassifier (scikit-learn standard)

Observations

Le classificateur Random Forest de cuML a démontré des avantages de performance convaincants par rapport à ses hom

Réduction spectaculaire du temps de formation : Le modèle accéléré par GPU a été entraîné près de 22 fois plus rapid

· Inférence évolutive : L'inférence était également plus rapide, se terminant en un peu plus d'une seconde, rendant le m

Efficacité de la mémoire : En utilisant cuDF, toutes les opérations de prétraitement des données et de modélisation sor

Changements de code minimaux : La transition de Scikit-learn à cuML n'a nécessité que des ajustements syntaxiques

Précision constante : Le modèle GPU a atteint une précision de 84,79 %, correspondant étroitement aux performances

Cependant, une limitation clé est l'absence d'une méthode predict_proba(), ce qui limite le calcul des scores ROC AUC. De

L'efficacité computationnelle et l'expérience des développeurs font de RAPIDS un excellent choix pour les tâches d'appren

Implémentation du réseau neuronal PyTorch

Pour explorer une approche d'apprentissage profond, un réseau neuronal feedforward a été implémenté en utilisant PyTor

Architecture du modèle

Le modèle utilisé était un réseau feedforward entièrement connecté avec la structure suivante:

Couche d'entrée : nombre de caractéristiques du jeu de données (numérique)

· Couche cachée : 64 neurones, suivie d'une activation ReLU

· Couche de sortie : 3 neurones correspondant aux trois étiquettes de classe (0, 1, 2)

La perte de cross-entropie a été utilisée comme fonction objectif, et le modèle a été entraîné en utilisant l'optimiseur Adam

Configuration de formation

Taille du lot : 1024

Époques: 20

· Optimiseur : Adam avec un taux d'apprentissage = 0,001

Fonction de perte : CrossEntropyLoss

· Appareil : CUDA (GPU) si disponible, sinon CPU

Métriques d'évaluation

Après l'entraînement, le modèle a été évalué en utilisant des métriques de classification standard. Les prédictions ont été r

Réseau de neurones métrique (PyTorch)
Temps d'entraînement (s)
41,46
Temps d'inférence (s)
0,52
Précision
0,8483
Précision
0,8054
Rappel
Score F1
N/A

Tableau 4 : Performance du classificateur de réseau neuronal (PyTorch)

3ROC AUC n'a pas été calculé en raison de la complexité de l'évaluation basée sur la probabilité multi-classe dans PyTorc

Neuf

Observations

Le réseau neuronal a obtenu la plus grande précision parmi tous les modèles (84,83 %), légèrement supérieur aux variante
Les principaux avantages de l'utilisation de PyTorch dans cette tâche :

Haute précision : Le modèle a efficacement capturé les motifs non linéaires dans les données.

· Personnalisation : L'architecture neuronale, la perte et les routines d'optimisation peuvent être facilement ajustées po

· Scalabilité : PyTorch peut s'adapter à des modèles plus complexes, des couches plus profondes ou des tailles de lot

En résumé, bien que les réseaux neuronaux ne surpassent peut-être pas les méthodes d'ensemble en termes d'efficacité d

Évaluation de la performance

Cette section consolide les résultats de tous les modèles évalués dans le projet. Les performances sont analysées à la fois

6.1
Métriques de précision des prédictions
Tous les modèles ont montré des performances compétitives en termes de métriques de classification. Le réseau neurona

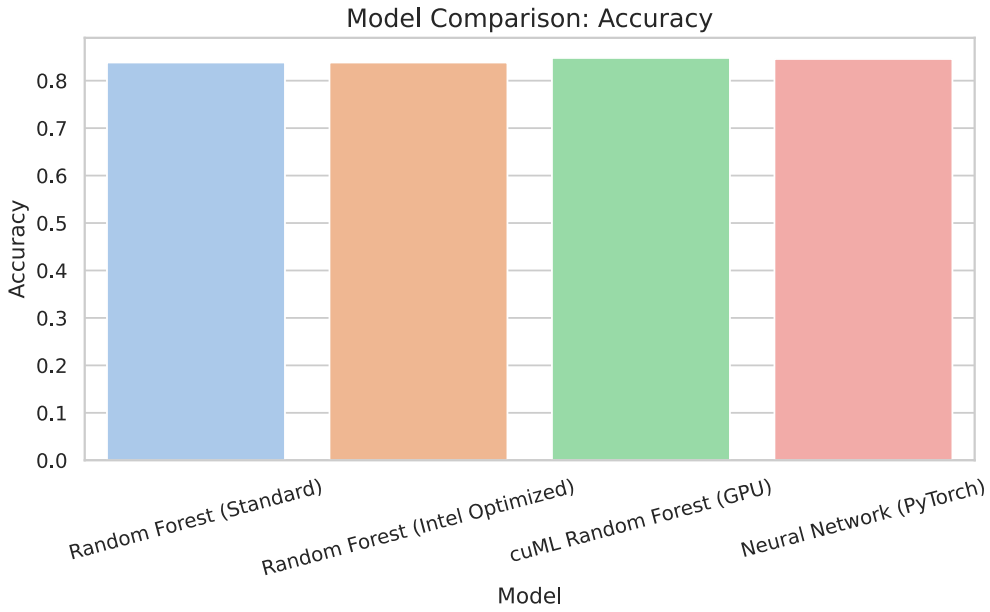

Figure 4: Comparaison des modèles : Précision

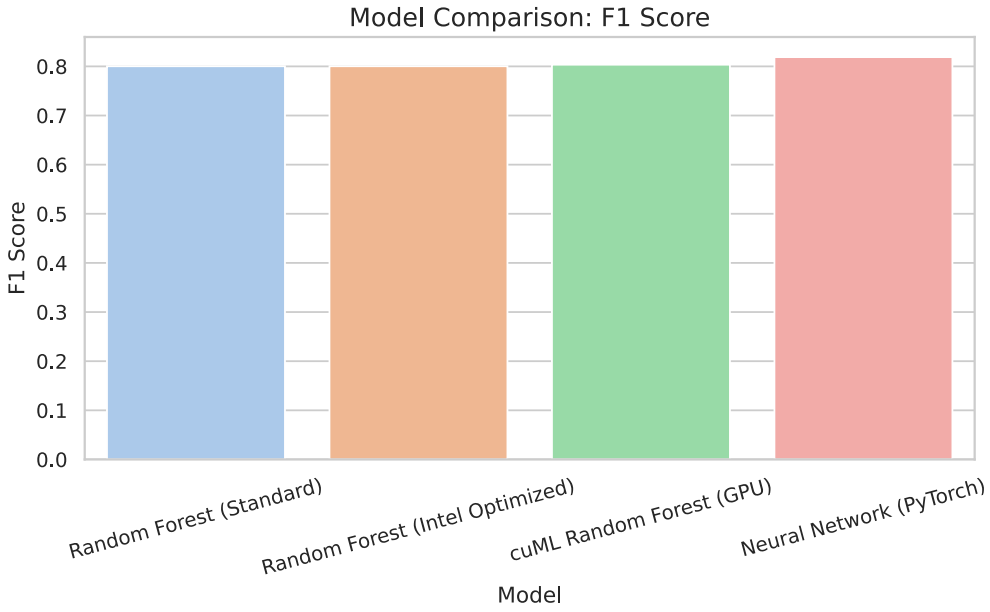

Figure 5: Comparaison des modèles : Score F1

Tous les modèles ont obtenu des scores F1 dans une plage étroite autour de 0,80, suggérant une performance équilibrée.

6.2

Performance informatique

Les gains les plus significatifs ont été observés en termes de performance informatique. Le Random Forest de cuML sur G

Forêt aléatoire de Scikit-learn. Pendant ce temps, le réseau neuronal PyTorch, bien qu'exact, a nécessité beaucoup plus d

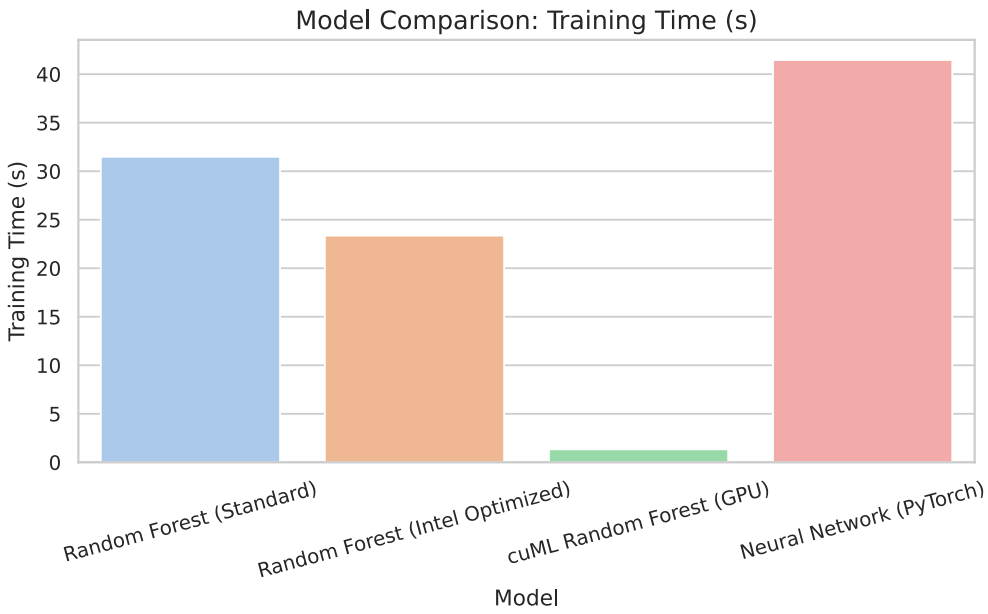

Figure 6: Comparaison des modèles : Temps d'entraînement

En termes d'inférence, le réseau neuronal était le plus rapide, probablement en raison des opérations matricielles efficaces

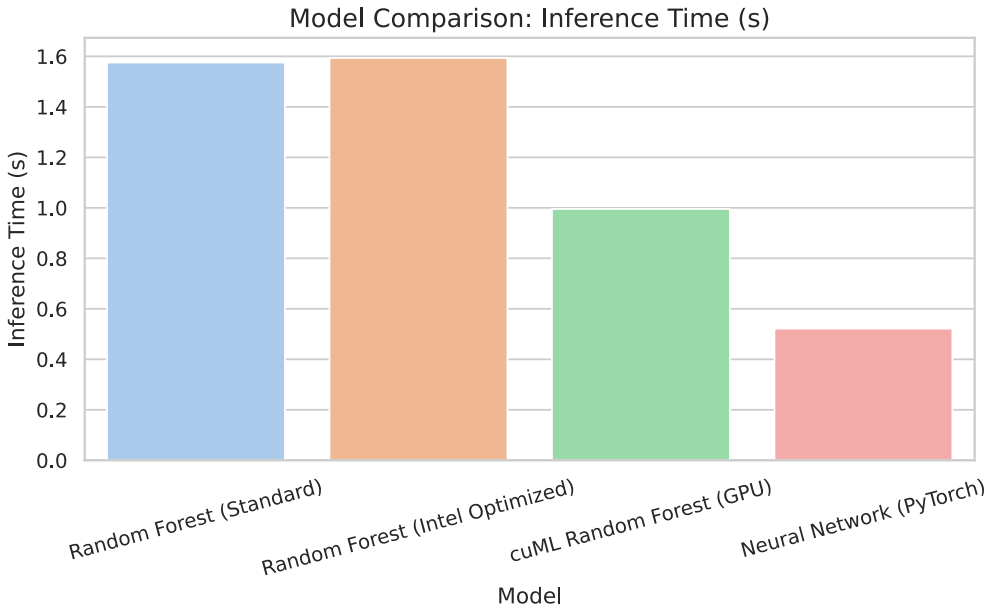

Figure 7: Comparaison des modèles: Temps d'inférence

Analyse de l'accélération

Pour évaluer les avantages de l'accélération, le gain de temps d'entraînement a été calculé par rapport au modèle standar

Random Forest cuML (GPU) a obtenu plus de 22 fois plus rapide dans le temps d'entraînement.

Le Random Forest optimisé par Intel a offert une légère accélération (1,35x), probablement limitée par le matériel de G

Le réseau neuronal a montré l'entraînement le plus lent mais a maintenu la capacité d'inférence en temps réel.

Ces résultats confirment que l'accélération basée sur le GPU offre les améliorations les plus tangibles, notamment lors du

Conclusion

Ce projet a exploré l'application de l'apprentissage automatique supervisé pour la prédiction du diabète en utilisant l'ensem

Forêt aléatoire Scikit-learn standard

Forêt aléatoire optimisée par Intel en utilisant oneAPI

Forêt aléatoire cuML sur GPU en utilisant NVIDIA RAPIDS

Réseau de neurones basé sur PyTorch

Modèle	Temps d'entraînement (s)	Temps d'inférence (s)	Précision	Précision	Rappel
Forêt aléatoire (Standard)	31.48	1.57	0.8388	0.7	0.8
Forêt aléatoire (Intel Opt.)	23	1.5	0.8388	0.7	0.8
Forêt aléatoire (cuML)	1.3	0.9	0.8479	0.8010	0.8479
Réseau neuronal (PyTorch)					

Tableau 5 Tableau de synthèse final: Comparaison des performances de tous les modèles

Tous les modèles ont atteint une précision de classification élevée et des scores F1 élevés, démontrant l'efficacité des tech

Le Random Forest de cuML offrait le meilleur compromis entre précision et vitesse, avec le temps d'entraînement le pl

Le réseau de neurones PyTorch a atteint la plus haute précision, le temps d'inférence le plus court mais a nécessité le

Les modèles optimisés par Intel ont montré des gains minimes dans l'environnement cloud, mais offrent une portabilité

Dans l'ensemble, le projet illustre l'impact de l'optimisation consciente du matériel en apprentissage automatique et met en

8

Références

1. Dépôt de machine learning de l'UCI. Ensemble de données des indicateurs de santé du diabète du CDC. Disponible

Scikit-learn : Apprentissage automatique en Python. Pedregosa et al., Journal of Machine Learning Research, 2011. htt

3. Documentation de la trousse à outils d'analyse AI d'Intel. Disponible sur : https://www.intel.com/content/www/us/en/d

4. Documentation de NVIDIA RAPIDS. Disponible sur : https://docs.rapids.ai/

PyTorch : Une bibliothèque d'apprentissage profond à haute performance de style impératif. Paszke et al., NeurIPS 201